# Java Documentation

1^er mai 2002

## Table des matières

# 1 fungus

## 1.1 Diff

`public class Diff extends fungus.core.infrastructure.SimulationAdapter`

`java.lang.Object`
p. 64 `fungus.core.infrastructure.SimulationAdapter`
p. 13 `fungus.Diff`

Lance le test sur les structure diffusive

### 1.1.1 Attributes

p. 89 `private fungus.core.service.InfoService is` : contient la liste des agents lances. Utilise lors du demarage de ces derniers.

p. 108 `private fungus.core.service.VisualisationService vis` :

`private int nbc` :

`private float dmax` :

p. 125 `private fungus.core.utils.ParametersCollector p` :

### 1.1.2 Constructors

**`public Diff(Infrastructure infra)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure infra` :

### 1.1.3 Methods

**`public void declareParameters(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

**`public void getParameters(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

**`public void buildAgent(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

**`public void startSimulation(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

## 1.2 F

`public class F extends java.lang.Object`

```
java.lang.Object
  fungus.F
```
p. 14

Ceci est la classe qui lance Fungus.

### 1.2.1 Attributes

p. 53

`protected static fungus.core.infrastructure.Infrastructure infra` :

### 1.2.2 Constructors

**`public F()`**

### 1.2.3 Methods

**`public static void main(String [] args) throws InterruptedException, Exception`**

**Parameters**

- `java.lang.String [] args` :

**Exception**

- `java.lang.InterruptedException` :
- `java.lang.Exception` :

**`public static void init() throws InterruptedException, Exception`**

**Exception**

- `java.lang.InterruptedException` :
- `java.lang.Exception` :

**`public static void act() throws InterruptedException`**

**Exception**

- `java.lang.InterruptedException` :

## 1.3 FigTest

`public class FigTest extends java.lang.Object`

```
java.lang.Object
  fungus.FigTest
```
p. 15

Simple Test of Fig Graphics

### 1.3.1 Constructors

**`public FigTest()`**

### 1.3.2 Methods

**`public static void main(String [] argv) throws Exception`**

**Parameters**

`java.lang.String [] argv` :

**Exception**

`java.lang.Exception` :

## 1.4 Fungus

`public class Fungus extends java.lang.Object`

```
java.lang.Object
  fungus.Fungus
```

p. 16

Ceci est la classe qui lance Fungus.

### 1.4.1 Attributes

p. 53 `protected static fungus.core.infrastructure.Infrastructure infra` :

### 1.4.2 Constructors

**`public Fungus()`**

### 1.4.3 Methods

**`public static void main(String [] args) throws InterruptedException, Exception`**

**Parameters**

`java.lang.String [] args` :

**Exception**

`java.lang.InterruptedException` :

`java.lang.Exception` :

## 1.5 PSTest

`public class PSTest extends java.lang.Object`

```
java.lang.Object
  fungus.PSTest
```

p. 17

### 1.5.1 Constructors

**public PSTest()**

### 1.5.2 Methods

**public static void main(String [] argv)**

**Parameters**

`java.lang.String [] argv` :

### 1.6 St

`public class St extends fungus.core.infrastructure.SimulationAdapter`

```
         java.lang.Object
p. 64      fungus.core.infrastructure.SimulationAdapter
p. 18        fungus.St
```

Un exemple de StartUp

#### 1.6.1 Attributes

`private java.util.Vector v :`
contient la liste des agents lances. Utilise lors du demarage de ces derniers.

p. 89 `private fungus.core.service.InfoService is :`

p. 83 `private fungus.core.service.CreateAgentService ca :`

p. 125 `private fungus.core.utils.ParametersCollector p :`

#### 1.6.2 Constructors

**`public St(Infrastructure infra)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure infra :`

#### 1.6.3 Methods

**`public void declareParameters(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i :`

**`public void getParameters(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i :`

**`public void buildAgent(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i :`

# 2 fungus.core

Contient le coeur de Fungus.

## 2.1 ExceptionManager

`public class ExceptionManager extends java.lang.Object`
`implements ExceptionManagerInterface`

`java.lang.Object`
p. 19 `fungus.core.ExceptionManager`

Gestionnaire des Exceptions.
Toutes les Exceptions attrapées dans un `catch` doivent être traité par cette classe. Elle se charge notement d'informer l'utilisateur.

### 2.1.1 Attributes

p. 21 `private static fungus.core.ExceptionManagerInterface em` :

`public static final int warning` :

`public static final int error` :

`public static final int fatal` :

`public static final int noTrace` :
Si ce bit est activé alors on affiche pas la trace.

### 2.1.2 Constructors

**`public ExceptionManager()`**

### 2.1.3 Methods

**`public static void raise(String msg, Throwable e)`**
Une exception à été levée et doit être taitée.

**Parameters**

`java.lang.String msg` : un message à affiché

`java.lang.Throwable e` : l'exception levée

**`public static void raise(Throwable e)`**
Une exception à été levée et doit être taitée.

**Parameters**

`java.lang.Throwable e` : l'exception levée

**`public static void raise(String msg, Throwable e, int type)`**
Une exception à été levée et doit être taitée.

**Parameters**

`java.lang.String msg` :

`java.lang.Throwable e` : l'exception levée

`int type` :

**public static void raise(Throwable e, int type)**
Une exception à été levée et doit être taitée.

**Parameters**

`java.lang.Throwable e` : l'exception levée

`int type` : le type de l'exception

**public static void setExceptionManager(ExceptionManagerInterface em)**
Indique la manière de traiter les Exceptions

**Parameters**

p. 21 `fungus.core.ExceptionManagerInterface em` : l'`ExceptionManagerInterface` du wrapper.

**public void raisei(String msg, Throwable e)**

**Parameters**

`java.lang.String msg` :

`java.lang.Throwable e` :

**public void raisei(String msg, Throwable e, int type)**

**Parameters**

`java.lang.String msg` :

`java.lang.Throwable e` :

`int type` :

## 2.2 `ExceptionManagerInterface`

`public interface class ExceptionManagerInterface extends null`

`fungus.core.ExceptionManagerInterface`

L'interface des gestionnaires d'exceptions.

### 2.2.1 Methods

**`public void raisei(String msg, Throwable e)`**
Traite une exception.

**Parameters**

`java.lang.String msg` : un message à affiché

`java.lang.Throwable e` : l'exception levée

**`public void raisei(String msg, Throwable e, int type)`**
Traite une exception.

**Parameters**

`java.lang.String msg` : un message à affiché

`java.lang.Throwable e` : l'exception levée

`int type` : le type de l'exception

## 2.3 FungusException

`public class FungusException extends java.lang.Exception`

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      fungus.core.FungusException
```
p. 22

### 2.3.1 Constructors

**public FungusException()**

**public FungusException(String s)**

**Parameters**

java.lang.String s :

## 2.4 MethodNotImplementedException

`public class MethodNotImplementedException extends java.lang.RuntimeException`

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        fungus.core.MethodNotImplementedException
```

p. 23

Levé si une méthode n'est pas encore implémenté.

### 2.4.1 Constructors

**`public MethodNotImplementedException()`**

**`public MethodNotImplementedException(String s)`**

**Parameters**

`java.lang.String s` :

# 3 fungus.core.agent

## 3.1 AccessibleAgent

`public abstract class AccessibleAgent extends fungus.core.agent.Agent`
`implements MessageListener`

`java.lang.Object`
p. 26 `fungus.core.agent.Agent`
p. 24 `fungus.core.agent.AccessibleAgent`

Classe qui/dont on peut accéder des ressources partagées.
Les ressources à partager sont enregistrées dans un vecteur.
Lors de la creation d'un canal, on envoie la liste des ressources visibles.

### 3.1.1 Attributes

p. 138 `protected fungus.core.utils.hash.ObjectHash publicProperties` : Propriétés publique (prop -> valeur)

p. 138 `protected fungus.core.utils.hash.ObjectHash propertiesList4Others` : liste des propriétés des autres agents : (Agent -> liste propriété)

p. 138 `protected fungus.core.utils.hash.ObjectHash propertiesOthers` : Propriétés connues des autres agents : [Agent,prop] -> valeur

### 3.1.2 Constructors

**`public AccessibleAgent()`**

### 3.1.3 Methods

**`protected void setProperty(Object prop, Serializable val)`**
Enregistre une de ses propriétés.

**Parameters**

`java.lang.Object prop` : an `Object` value

`java.io.Serializable val` : an `Object` value

**`protected Serializable getProperty(Object prop)`**

**Parameters**

`java.lang.Object prop` :

**return** `java.io.Serializable` :

**`protected void recordProperty(String name)`**
Enregistre une propriété comme étant accessible

**Parameters**

`java.lang.String name` : a `String` value

**`protected void getVisibleProperties(MessageEvent m)`**
Envoie à l'autre agent la liste des ressources disponibles

**Parameters**

p. 47 `fungus.core.canal.MessageEvent m` :

**protected void askProperty(AgentID a, Serializable property)**

Renvoie une ressource

**Parameters**

p. 28 `fungus.core.agent.AgentID a` : a `Agent` value

`java.io.Serializable property` : la cle de la propriete, (souvent une chaine)

**protected void giveProperty(AgentID a, Object prop)**

Donne une property à un autre agent.

**Parameters**

p. 28 `fungus.core.agent.AgentID a` : L'agent à qui on donne

`java.lang.Object prop` : La propriete que l'on donne

**public synchronized void messageRecevied(MessageEvent m)**

Reçoit un message. Dans le cas d'un AccessibleAgent, il parse la liste des messages possibles et agit en conséquence.

**Parameters**

p. 47 `fungus.core.canal.MessageEvent m` : a `MessageEvent` value

**public static Vector makePOKey(AgentID a, Object s)**

Fabrique un clé pour la reception des messages :

**Parameters**

p. 28 `fungus.core.agent.AgentID a` : agent source du message

`java.lang.Object s` : propriete que l'on stocke.

**return** `java.util.Vector` :
le clé

**public void addCanal(Canal c, AgentID other)**

Lors de la creation d'un canal, on demande les proprietes de l'autre.

**Parameters**

p. 41 `fungus.core.canal.Canal c` :

p. 28 `fungus.core.agent.AgentID other` :

## 3.2 Agent

`public abstract class Agent extends java.lang.Object`
`implements Runnable`

`java.lang.Object`
p. 26 `fungus.core.agent.Agent`

Classe abstraite Agent.

### 3.2.1 Attributes

p. 53 `protected fungus.core.infrastructure.Infrastructure infra` : The infrastructure who create this Agent

p. 136 `protected fungus.core.utils.hash.AgentIDHash canal` : liste les canaux (AgentID -> Canal)

p. 28 `protected fungus.core.agent.AgentID id` : ID de cet Agent.

`private boolean go` :

`private boolean stopped` :

### 3.2.2 Constructors

**`public Agent()`**

### 3.2.3 Methods

**`public void setInfrastructure(Infrastructure i)`**
Set a new infrastructure to an agent

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` : an `Infrastructure` value

**`public void run()`**

**`public void start()`**
Autorise l'agent à calculer

**`public void stop()`**
arrete l'agent

**`public AgentID getID()`**
Renvoie l'ID de l'agents

p. 28 **return** `fungus.core.agent.AgentID` : an `AgentID` value

**`public void addCanal(Canal c, AgentID other)`**
Un agent veut communiquer avec moi ?
ne fait qu'un `canal.put(other,c);` Appelé automatiquement lors de la création d'un canal.

**Parameters**

p. 41 `fungus.core.canal.Canal c` : le canal à ajouter
p. 28 `fungus.core.agent.AgentID other` : J'envoie les message à cet agent.

**public void removeCanal(AgentID a)**

Enlève un canal de la liste des canaux disponibles
Ne pas utiliser directement. On préfère utiliser la méthode Canal.remove qui fait le travail.

**Parameters**

p. 28 `fungus.core.agent.AgentID a` : l'agent avec qui on ne communique plus

**protected Canal newCanal(AgentID dest) throws CreateCanalException**

Fabrique un nouveau canal entre cet agent et "dest" si celui n'existe pas.

**Parameters**

p. 28 `fungus.core.agent.AgentID dest` : an `Agent` value

p. 41 **return** `fungus.core.canal.Canal` : le canal.

**Exception**

p. 45 `fungus.core.canal.CreateCanalException` :

**public void setName(AgentID id)**

Enregistre le nom de l'agent.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : an `AgentID` value

**public String toString()**

**return** `java.lang.String` :

**public abstract void init(ObjectHash p)**

Initialise l'agent

**Parameters**

p. 138 `fungus.core.utils.hash.ObjectHash p` :

**public abstract void act()**

Action de l'agent. Ce qu'il fait.

**public void remove()**

Tue moi !
– arrete le thread
– deconnecte tous les canaux

### 3.3 AgentID

```
public class AgentID extends java.lang.Object
   implements Serializable
```

```
java.lang.Object
  fungus.core.agent.AgentID
```

p. 28

Identificateur d'un agent

#### 3.3.1 Attributes

`private java.lang.String name :`

`private java.net.InetAddress host :`

`private int hostkey :`

`private int hashcode :`

#### 3.3.2 Constructors

**`public AgentID(String host, String name)`**

Creates a new `AgentID` instance.

**Parameters**

- `java.lang.String host` : hôte de l'agent
- `java.lang.String name` : nom de l'agent

**`public AgentID(String host, int hostkey, String name)`**

**Parameters**

- `java.lang.String host` :
- `int hostkey` :
- `java.lang.String name` :

**`public AgentID(InetAddress host, int hostkey, String name)`**

**Parameters**

- `java.net.InetAddress host` :
- `int hostkey` :
- `java.lang.String name` :

**`public AgentID(String str)`**

Fabrique un identificateur à partir d'une simple chaine.
La chaîne doit avoir la forme suivante :

`\emph{host}:\emph{name}`

**Parameters**

- `java.lang.String str` : la chaîne représentant l'agent

#### 3.3.3 Methods

**`public String getName()`**

**return** `java.lang.String` :

**`public InetAddress getHost()`**

**return** `java.net.InetAddress` :

**`public int getHostKey()`**

**return** `int` :

**`public boolean equals(AgentID o)`**

Compare deux identificateurs

**Parameters**

p. 28 `fungus.core.agent.AgentID o` : an `AgentID` value

**return** `boolean` :
true si l'ID sont les même.

**`public boolean equals(Object o)`**

**Parameters**

`java.lang.Object o` :

**return** `boolean` :

**`public String toString()`**

**return** `java.lang.String` :

**`public int hashCode()`**

**return** `int` :

## 3.4 AgentsGroup

```
public class AgentsGroup extends fungus.core.agent.Agent
   implements MessageListener
```

java.lang.Object
p. 26 fungus.core.agent.Agent
p. 30 fungus.core.agent.AgentsGroup

Groupe un ensemble d'agents

### 3.4.1 Attributes

p. 89 `private fungus.core.service.InfoService is` :

p. 100 `private fungus.core.service.SchedulerService schd` : Le scheduler qui gere le groupe.

### 3.4.2 Constructors

**public AgentsGroup()**

### 3.4.3 Methods

**public void init(ObjectHash p)**

**Parameters**

p. 138 `fungus.core.utils.hash.ObjectHash p` :

**public void act()**

**public synchronized void messageRecevied(MessageEvent m)**

**Parameters**

p. 47 `fungus.core.canal.MessageEvent m` :

**public void start()**
Autorise l'agent à calculer

**public void stop()**
arrete l'agent

**public void addAgent(AgentID id)**
Ajoute un agent au groupe. L'ajout se fait normalement lors de la création de l'agent.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : L'agent que l'on ajoute

**public void removeAgent(AgentID id)**

**Parameters**

p. 28 `fungus.core.agent.AgentID id` :

**public void setPause(AgentID id, Time millis)**
Programme une pause pour l'agent

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : L'agent qui veut pauser
p. 73 `fungus.core.scheduler.Time millis` : Le temps de pause en unite de temp

## 3.5 AgentsGroupMessage

```
public class AgentsGroupMessage extends java.lang.Object
   implements Message
```

```
java.lang.Object
  fungus.core.agent.AgentsGroupMessage
```
p. 31

Message envoyé aux AgentsGroup

### 3.5.1 Attributes

`public static final byte addAgent` :
On ajoute un agent au groupe.
L'agent à ajouter est dans la source

`public static final byte removeAgent` :
On enlève un agent.
L'agent à enlever est dans la source

`public static final byte setPause` :
Un agent fait une pause.
L'agent qui pause est dans la source.
La durée de la pause est dans le paramètre.

`public static final byte broadcastMessage` :
Envoie une message à tous les agents du groupe.

`public static final int type` :

p. 28 `private fungus.core.agent.AgentID agent` :

`private byte cmd` :

`private java.io.Serializable param` :

### 3.5.2 Constructors

**public AgentsGroupMessage(AgentID source, byte cmd, Serializable param)**

**Parameters**

p. 28 `fungus.core.agent.AgentID source` :
`byte cmd` :
`java.io.Serializable param` :

### 3.5.3 Methods

**public AgentID getSource()**

p. 28 **return** `fungus.core.agent.AgentID` :

**public byte getCmd()**

**return** `byte` :

**public Serializable getParam()**

**return** `java.io.Serializable` :

**public int getType()**

**return** `int` :

**public String toString()**

**return** `java.lang.String` :

## 3.6 EmptyAgent

public class EmptyAgent extends fungus.core.agent.Agent

java.lang.Object
p. 26 fungus.core.agent.Agent
p. 33 fungus.core.agent.EmptyAgent

Agent Vide.

### 3.6.1 Constructors

**public EmptyAgent()**

### 3.6.2 Methods

**public final void init(ObjectHash p)**

**Parameters**

p. 138 fungus.core.utils.hash.ObjectHash p :

**public final void act()**

## 3.7 NotifyAgent

`public abstract class NotifyAgent extends fungus.core.agent.AccessibleAgent`
`implements MessageListener`

java.lang.Object
p. 26 fungus.core.agent.Agent
p. 24 fungus.core.agent.AccessibleAgent
p. 34 fungus.core.agent.NotifyAgent

On peut notifier les nouvelles à des agents lorsqu'on les modifie.

### 3.7.1 Attributes

p. 138 `private fungus.core.utils.hash.ObjectHash notifyList` : prop -> (id1,id2,...)

### 3.7.2 Constructors

**public NotifyAgent()**

### 3.7.3 Methods

**protected void recordNotifcation(AgentID a, Serializable property)**
Enregistre une propriété notifiable.

**Parameters**

p. 28 `fungus.core.agent.AgentID a` : an `Agent` value
`java.io.Serializable property` : an `Object` value

**private void recordANotification(AgentID a, Object p)**
L'agent a veut recevoir le modification de p

**Parameters**

p. 28 `fungus.core.agent.AgentID a` : L'agent qui veut recevoir
`java.lang.Object p` : La propriété que l'on veut recevoir

**protected void setPropertyAndNotify(Object prop, Serializable val)**
Change une propriété et prévient les autres agents

**Parameters**

`java.lang.Object prop` : an `Object` value
`java.io.Serializable val` : an `Object` value

**protected void notify(Object prop)**
prévient les autres agents que la propriété à changé.

**Parameters**

`java.lang.Object prop` : an `Object` value

**public synchronized void messageRecevied(MessageEvent m)**
Reçoit un message.

**Parameters**

p. 47 `fungus.core.canal.MessageEvent m` : a `MessageEvent` value

## 3.8 NullAgentException

`public class NullAgentException extends fungus.core.FungusException`

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      fungus.core.FungusException
        fungus.core.agent.NullAgentException
```

p. 22
p. 35

Throwed if an agent does not exist.

### 3.8.1 Constructors

**`public NullAgentException()`**

**`public NullAgentException(String e)`**

**Parameters**

`java.lang.String e` :

# 4 fungus.core.canal

## 4.1 AbstractCanal

```
public abstract class AbstractCanal extends java.lang.Object
   implements Canal
```

```
java.lang.Object
  fungus.core.canal.AbstractCanal
```
p. 36

Canal Abstrait

### 4.1.1 Attributes

`private java.util.Vector MessageHandlerList :`

### 4.1.2 Constructors

**`public AbstractCanal()`**

### 4.1.3 Methods

**`public void addMessageHandler(MessageHandler m)`**

**Parameters**

p. 48 `fungus.core.canal.MessageHandler m :`

## 4.2 AccessibleMessage

`public class AccessibleMessage extends java.lang.Object`
`implements Message`

`java.lang.Object`
p. 37 `fungus.core.canal.AccessibleMessage`

Message envoyé par les ■AccessibleAgents■

### 4.2.1 Attributes

`public static final byte nop` :
No Operation

`public static final byte askProperties` :
Demande la liste des propriétés

`public static final byte propertiesList` :
Renvoie la liste des propriétés

`public static final byte propertyRequest` :
Demande un certaine propriété

`public static final byte propertyReceived` :
Donne une certaine propriété.

`public static final int type` :

p. 28 `private fungus.core.agent.AgentID agent` :

`private byte cmd` :

`private java.io.Serializable param` :

### 4.2.2 Constructors

**`public AccessibleMessage(AgentID source, byte cmd)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID source` :
`byte cmd` :

**`public AccessibleMessage(AgentID source, byte cmd, Serializable param)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID source` :
`byte cmd` :
`java.io.Serializable param` :

### 4.2.3 Methods

**`public AgentID getSource()`**

p. 28 **return** `fungus.core.agent.AgentID` :

**public byte getCmd()**

**return** `byte` :

**public Serializable getParam()**

**return** `java.io.Serializable` :

**public int getType()**

**return** `int` :

**public String toString()**

**return** `java.lang.String` :

## 4.3 BadCanalException

`public class BadCanalException extends fungus.core.FungusException`

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      fungus.core.FungusException
        fungus.core.canal.BadCanalException
```

p. 22 (fungus.core.FungusException)
p. 39 (fungus.core.canal.BadCanalException)

### 4.3.1 Constructors

**`public BadCanalException()`**

**`public BadCanalException(String s)`**

**Parameters**

`java.lang.String s` :

## 4.4 BadMessageException

public class BadMessageException extends fungus.core.FungusException

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      fungus.core.FungusException
        fungus.core.canal.BadMessageException
```

p. 22
p. 40

### 4.4.1 Constructors

**public BadMessageException()**

**public BadMessageException(String s)**

**Parameters**

java.lang.String s :

## 4.5 Canal

`public interface class Canal extends null`

`fungus.core.canal.Canal`

Canal de communication.

### 4.5.1 Methods

**`public void sendTo(AgentID to, Message msg) throws BadCanalException`**
Envoie un message à un autre agent

**Parameters**

p. 28 `fungus.core.agent.AgentID to` : a qui on envoie le msg

p. 46 `fungus.core.canal.Message msg` : Un message quelconque.

**Exception**

p. 39 `fungus.core.canal.BadCanalException` :

**`public void remove()`**
Supprime un canal

**`public void addMessageHandler(MessageHandler m)`**
Ajoute un traitement sur le signal

**Parameters**

p. 48 `fungus.core.canal.MessageHandler m` : le MessageHandler qui va traiter le signal.

## 4.6 CanalLocal

`public class CanalLocal extends fungus.core.canal.AbstractCanal`

```
java.lang.Object
  fungus.core.canal.AbstractCanal
    fungus.core.canal.CanalLocal
```

p. 36
p. 42

Canal local.

### 4.6.1 Attributes

p. 28 `private fungus.core.agent.AgentID id1` :

p. 28 `private fungus.core.agent.AgentID id2` :

p. 26 `private fungus.core.agent.Agent agent1` :

p. 26 `private fungus.core.agent.Agent agent2` :

p. 53 `private fungus.core.infrastructure.Infrastructure infra` :

p. 89 `private fungus.core.service.InfoService is` :

### 4.6.2 Constructors

**`public CanalLocal(AgentID id1, AgentID id2, Infrastructure infra) throws CreateCanalException`**

Crée un nouveau canal local

**Parameters**

p. 28 `fungus.core.agent.AgentID id1` : ID du premient agent
p. 28 `fungus.core.agent.AgentID id2` : ID du second agent
p. 53 `fungus.core.infrastructure.Infrastructure infra` : infrastructure qui a crée le canal

**Exception**

p. 45 `fungus.core.canal.CreateCanalException` : if an error occurs

### 4.6.3 Methods

**`public void sendTo(AgentID to, Message msg) throws BadCanalException`**
Envoie un message à un autre agent

**Parameters**

p. 28 `fungus.core.agent.AgentID to` : a qui on envoie le msg
p. 46 `fungus.core.canal.Message msg` : Un message quelconque.

**Exception**

p. 39 `fungus.core.canal.BadCanalException` :

**`public void remove()`**
Supprime un canal

### 4.7 CanalNet

`public class CanalNet extends fungus.core.canal.AbstractCanal`
`implements NetMessageListener`

`java.lang.Object`
p. 36 `fungus.core.canal.AbstractCanal`
p. 43 `fungus.core.canal.CanalNet`

Canal reseau.
TODO : Penser a compresser le flux...

#### 4.7.1 Attributes

p. 28 `private fungus.core.agent.AgentID id1` :

p. 28 `private fungus.core.agent.AgentID id2` :

p. 26 `private fungus.core.agent.Agent agent1` :

p. 26 `private fungus.core.agent.Agent agent2` :

p. 56 `private fungus.core.infrastructure.NetConnection c1` :

p. 56 `private fungus.core.infrastructure.NetConnection c2` :

p. 53 `private fungus.core.infrastructure.Infrastructure infra` :

p. 89 `private fungus.core.service.InfoService is` :

`public static final java.lang.String message` :
On envoie un message

#### 4.7.2 Constructors

**`public CanalNet(Infrastructure infra)`**
Constructeur vide pour pouvoir invoquer performMessage()

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure infra` :

**`public CanalNet(AgentID id1, AgentID id2, Infrastructure infra, boolean createOtherPart) throws CreateCanalException`**

Creates a new `CanalNet` instance.

**Parameters**

p. 28 `fungus.core.agent.AgentID id1` : le premier agent
p. 28 `fungus.core.agent.AgentID id2` : le second agent
p. 53 `fungus.core.infrastructure.Infrastructure infra` : l'infrastructure creatrice
`boolean createOtherPart` : si `true` on demende la creation de la seconde partie du canal. (sur la machine distante)

**Exception**

p. 45 `fungus.core.canal.CreateCanalException` : if an error occurs

### 4.7.3 Methods

**`public void sendTo(AgentID to, Message msg) throws BadCanalException`**

Envoie un message à un autre agent

**Parameters**

p. 28 `fungus.core.agent.AgentID to` : a qui on envoie le msg

p. 46 `fungus.core.canal.Message msg` : Un message quelconque.

**Exception**

p. 39 `fungus.core.canal.BadCanalException` :

**`public void remove()`**

Supprime un canal.
pas sur que ca marche!!!

**`private void createOtherPart(NetConnection c)`**

Demande la création du canal sur l'hôte distant

**Parameters**

p. 56 `fungus.core.infrastructure.NetConnection c` : a `NetConnection` value

**`public boolean performMessage(String type, ObjectInputStream in, NetConnection nc)`**

**Parameters**

`java.lang.String type` :

`java.io.ObjectInputStream in` :

p. 56 `fungus.core.infrastructure.NetConnection nc` :

**return** `boolean` :

## 4.8 CreateCanalException

public class CreateCanalException extends fungus.core.FungusException

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      fungus.core.FungusException
        fungus.core.canal.CreateCanalException
```

p. 22
p. 45

### 4.8.1 Constructors

**public CreateCanalException()**

**public CreateCanalException(String s)**

**Parameters**

java.lang.String s :

## 4.9 Message

`public interface class Message extends null`
`implements Serializable`

`fungus.core.canal.Message`

Message pouvant transite par les canal

### 4.9.1 Methods

**`public AgentID getSource()`**

p. 28 **return** `fungus.core.agent.AgentID` :

**`public byte getCmd()`**

**return** `byte` :

**`public Serializable getParam()`**

**return** `java.io.Serializable` :

**`public int getType()`**
Renvoie le type du message

**return** `int` :
an `int` value

## 4.10 MessageEvent

public class MessageEvent extends java.util.EventObject

```
java.lang.Object
  java.util.EventObject
    fungus.core.canal.MessageEvent
```

p. 47

### 4.10.1 Attributes

java.lang.Object mesg :

### 4.10.2 Constructors

**public MessageEvent(Object source, Object mesg)**

**Parameters**

java.lang.Object source :

java.lang.Object mesg :

### 4.10.3 Methods

**public Object getMessage()**

**return** java.lang.Object :

## 4.11 `MessageHandler`

`public interface class MessageHandler extends null`

`fungus.core.canal.MessageHandler`

Interface de modifcation d'un message.

### 4.11.1 Methods

**`public Message messageModification(Message message)`**
Modifie une message

**Parameters**

p. 46 `fungus.core.canal.Message message` : le Message a modifier

p. 46 **return** `fungus.core.canal.Message` : le message modifier

## 4.12 MessageListener

`public interface class MessageListener extends null`

`fungus.core.canal.MessageListener`

### 4.12.1 Methods

**`public void messageRecevied(MessageEvent m)`**

**Parameters**

p. 47 `fungus.core.canal.MessageEvent m :`

### 4.13 NotifyMessage

`public class NotifyMessage extends fungus.core.canal.AccessibleMessage`

`java.lang.Object`
p. 37 `fungus.core.canal.AccessibleMessage`
p. 50 `fungus.core.canal.NotifyMessage`

Message envoyé par les ■NotifyAgents■

#### 4.13.1 Attributes

`public static final byte recordNotifcation` :
Donne une certaine propriété.

#### 4.13.2 Constructors

**`public NotifyMessage(AgentID source, byte cmd)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID source` :
`byte cmd` :

**`public NotifyMessage(AgentID source, byte cmd, Serializable param)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID source` :
`byte cmd` :
`java.io.Serializable param` :

# 5 fungus.core.infrastructure

## 5.1 CreateAgentException

`public class CreateAgentException extends fungus.core.FungusException`

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      fungus.core.FungusException
        fungus.core.infrastructure.CreateAgentException
```

p. 22
p. 51

Throws if we cannot create an agent.

### 5.1.1 Constructors

**public CreateAgentException()**

**public CreateAgentException(String e)**

**Parameters**

java.lang.String e :

## 5.2 FungusProtocolException

public class FungusProtocolException extends fungus.core.FungusException

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      fungus.core.FungusException
        fungus.core.infrastructure.FungusProtocolException
```

p. 22
p. 52

On ne parle pas à une infrastructure Fungus

### 5.2.1 Constructors

**public FungusProtocolException()**

**public FungusProtocolException(String e)**

**Parameters**

java.lang.String e :

## 5.3 Infrastructure

`public class Infrastructure extends java.lang.Object`
`implements NetMessageListener`

`java.lang.Object`
p. 53 `fungus.core.infrastructure.Infrastructure`

Cette classe est la base du système

### 5.3.1 Attributes

p. 89 `private fungus.core.service.InfoService is` :

`private java.lang.String simulationClass` :

p. 63 `private fungus.core.infrastructure.Simulation simulation` :

p. 62 `private fungus.core.infrastructure.ServiceFactory sf` :

p. 138 `private fungus.core.utils.hash.ObjectHash services` :

`public static final java.lang.String startSimulation` :
Signale que l'on veut démarer la simulation. Message reseau

`public static final java.lang.String createCanal` :
Signale que l'on veut créer un canal

`private boolean stopped` :

### 5.3.2 Constructors

**`public Infrastructure()`**
Creates a new `Infrastructure` instance.

### 5.3.3 Methods

**`public void init() throws FungusException`**
Initialise la simulation.
L'initialisation a été coupe en deux pour pouvoir être integre proprement dans la GUI.

**Exception**

p. 22 `fungus.core.FungusException` : if an error occurs

**`public Simulation init1() throws FungusException`**
Premiere étape de l'initialisation de l'infrastructure.

p. 63 **return** `fungus.core.infrastructure.Simulation` : a `Simulation` value

**Exception**

p. 22 `fungus.core.FungusException` :

**`public void init2(Simulation st) throws FungusException`**

**Parameters**

p. 63 `fungus.core.infrastructure.Simulation st` :

**Exception**

p. 22 `fungus.core.FungusException` :

**`public void stopAll() throws FungusException`**

**Exception**

p. 22 `fungus.core.FungusException` :

**`public void setSimulationClass(String c)`**

Fixe le nom de la classe de démarage

**Parameters**

`java.lang.String c` : la classe de démrage

**`public void startService(String service)`**

Démare un service

**Parameters**

`java.lang.String service` : le service à démarer

**`public Service service(String service) throws ServiceException`**

Renvoie un service stocké dans la listes des services

**Parameters**

`java.lang.String service` : le nom du service à renvoyer

p. 102 **return** `fungus.core.service.Service` : le service

**Exception**

p. 103 `fungus.core.service.ServiceException` : le service n'est pas trouvé

**`public void removeAgent(AgentID id)`**

tue un agent

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : an `AgentID` value

**`public Canal newCanal(AgentID id1, AgentID id2) throws CreateCanalException`**

Fabrique un nouveau canal

**Parameters**

p. 28 `fungus.core.agent.AgentID id1` : an `AgentID` value

p. 28 `fungus.core.agent.AgentID id2` : an `AgentID` value

p. 41 **return** `fungus.core.canal.Canal` : a `Canal` value

**Exception**

p. 45 `fungus.core.canal.CreateCanalException` : if an error occurs

**`public Canal newCanal(AgentID id1, AgentID id2, boolean flag) throws CreateCanalException`**

Fabrique un nouveau canal (que distant ici)

**Parameters**

p. 28 `fungus.core.agent.AgentID id1` : an `AgentID` value

p. 28 `fungus.core.agent.AgentID id2` : an `AgentID` value

`boolean flag` : Si true alors le canal distant n'est pas crée...

p. 41 **return** `fungus.core.canal.Canal` : a `Canal` value

**Exception**

p. 45 `fungus.core.canal.CreateCanalException` : if an error occurs

**`public boolean performMessage(String type, ObjectInputStream in, NetConnection out)`**

**Parameters**

`java.lang.String type` :

`java.io.ObjectInputStream in` :

p. 56 `fungus.core.infrastructure.NetConnection out` :

**return** `boolean` :

## 5.4 NetConnection

`public class NetConnection extends java.lang.Thread`

```
java.lang.Object
  java.lang.Thread
    fungus.core.infrastructure.NetConnection
```

p. 56

Connection réseaux.
supporte un socket et suis le protocole

### 5.4.1 Attributes

`public static final java.lang.String eom` :
End of message

`private static final java.lang.String name` :

`private java.net.Socket socket` :

p. 98 `private fungus.core.service.NetworkService ns` :

p. 89 `private fungus.core.service.InfoService is` :

p. 53 `private fungus.core.infrastructure.Infrastructure infra` :

`private java.io.BufferedOutputStream out` :

`private java.io.ObjectOutputStream oos` :

`private java.io.ObjectInputStream in` :

`private boolean isRunning` :

`private int jeton` :

`private boolean lock` :

### 5.4.2 Constructors

**`public NetConnection(Socket s, NetworkService n, Infrastructure i)`**
Créer une nouvelle connection entrante.

**Parameters**

- `java.net.Socket s` : La socket qui rentre
- p. 98 `fungus.core.service.NetworkService n` : Le server reseau de Fungus
- p. 53 `fungus.core.infrastructure.Infrastructure i` :

**`public NetConnection(Socket s, NetworkService n, Infrastructure i, boolean flag)`**

**Parameters**

- `java.net.Socket s` :
- p. 98 `fungus.core.service.NetworkService n` :
- p. 53 `fungus.core.infrastructure.Infrastructure i` :
- `boolean flag` :

### 5.4.3 Methods

**public void doConnection(Socket s, NetworkService n, Infrastructure i, boolean flag)**
Créer une nouvelle connection entrante.

**Parameters**

`java.net.Socket s` : La sockect qui rentre

p. 98 `fungus.core.service.NetworkService n` : Le server reseau de Fungus

p. 53 `fungus.core.infrastructure.Infrastructure i` :

`boolean flag` : si true alors connection sortante.

**private void sendAgents() throws IOException**
Envoie la listes des agents locaux à l'autre.

**Exception**

`java.io.IOException` :

**private void receiveAgents()**
Reçoit la liste des agents.

**private void sendKnowHosts() throws IOException**
Envoie la liste des hôtes connue à l'autre bout...

**Exception**

`java.io.IOException` :

**private void receiveHosts()**
Recoit la liste des hôtes connus

**public InetAddress getHost()**
Renvoie le nom de l'hôte local
Utiliser par NetworkService lors de la création d'une connection

**return** `java.net.InetAddress` :
a `String` value

**public void run()**
Attend des requêtes entrantes et les traitent

**public synchronized int startSend(Serializable msg)**
prend un jeton et envoie le debut message

**Parameters**

`java.io.Serializable msg` : a `Serializable` value

**return** `int` :
le jeton

**public void send(int jeton, Serializable msg)**
Envoie un message.

**Parameters**

`int jeton` : message number

`java.io.Serializable msg` : a `String` value

**`public void endSend(int jeton)`**

End of message. Remove the lock.

**Parameters**

`int jeton` : the message number

**`public String toString()`**

**return** `java.lang.String` :

## 5.5 NetMessageListener

`public interface class NetMessageListener extends null`

`fungus.core.infrastructure.NetMessageListener`

Interface pour les classes qui doivent recovoir des messages du reseau

### 5.5.1 Methods

**`public boolean performMessage(String type, ObjectInputStream in, NetConnection nc)`**
Lit et traite un message du reseau

**Parameters**

- `java.lang.String type` : le type du message
- `java.io.ObjectInputStream in` : le flux d'entree
- p. 56 `fungus.core.infrastructure.NetConnection nc` :

**return** `boolean` :
`true` si le message est traite ici

## 5.6 NetMessageSelector

`public class NetMessageSelector extends java.lang.Object`

`java.lang.Object`
p. 60 `fungus.core.infrastructure.NetMessageSelector`

Redirige les messages entrant du reseau. les associations <*type*> -> <*classe qui traite le message*> sont réalise dans la méthode `init`. Si possible changer cette méthode...

On utilise cette méthode simplement car les

### 5.6.1 Attributes

p. 138 `private static fungus.core.utils.hash.ObjectHash map` :

### 5.6.2 Constructors

**`public NetMessageSelector()`**

### 5.6.3 Methods

**`public static void init(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

**`public static void record(String type, NetMessageListener inst)`**
enregistre un type de messge à la classe qui la traitera.

**Parameters**

`java.lang.String type` : le type du message

p. 59 `fungus.core.infrastructure.NetMessageListener inst` : une instance de la classe qui traitera le message.

**`public static boolean perform(String type, ObjectInputStream in, NetConnection nc)`**

**Parameters**

`java.lang.String type` :

`java.io.ObjectInputStream in` :

p. 56 `fungus.core.infrastructure.NetConnection nc` :

**return** `boolean` :

## 5.7 NetworkedCreationAgentException

`public class NetworkedCreationAgentException extends fungus.core.infrastructure.CreateAgentException`

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      fungus.core.FungusException
        fungus.core.infrastructure.CreateAgentException
          fungus.core.infrastructure.NetworkedCreationAgentException
```

p. 22
p. 51
p. 61

Throws if a agent is created on another host.

### 5.7.1 Constructors

**`public NetworkedCreationAgentException()`**

**`public NetworkedCreationAgentException(String e)`**

**Parameters**

`java.lang.String e` :

## 5.8 `ServiceFactory`

`public class ServiceFactory extends java.lang.Object`

`java.lang.Object`
p. 62 `fungus.core.infrastructure.ServiceFactory`

Fabrique des services

### 5.8.1 Attributes

p. 53 `private fungus.core.infrastructure.Infrastructure infra` :

### 5.8.2 Constructors

**`public ServiceFactory(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

### 5.8.3 Methods

**`public Service getService(String service) throws ServiceException`**
Donne un nouveau service.
**Attention** on peut lancer plusieurs fois le meme service.

**Parameters**

`java.lang.String service` : nom du service

p. 102 **return** `fungus.core.service.Service` : Renvoie le service

**Exception**

p. 103 `fungus.core.service.ServiceException` : Si le service ne peut pas se lancer

### 5.9 Simulation

`public interface class Simulation extends null`

`fungus.core.infrastructure.Simulation`

Interface pour le démarage de Fungus.
Une classe implémentant cette interface pourra être passé en paramètre à Fungus (-f simulation)

#### 5.9.1 Methods

**`public void getParameters(Infrastructure i)`**
Collecte les paramètres de la simulation.

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` : infrastructure de la simulation

**`public void declareParameters(Infrastructure i)`**
Declare les paramètres à recuperer.

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` : an `Infrastructure` value

**`public void buildAgent(Infrastructure i)`**
Construit les agents de la simulation.

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` : infrastructure de la simulation

**`public void startSimulation(Infrastructure i)`**
démare les agents

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` : infrastructure de la simulation

**`public void stopSimulation(Infrastructure i)`**
Arrête les agents

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

**`public void removeSimulation(Infrastructure i)`**
Enlèvent les agents

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

## 5.10 SimulationAdapter

`public class SimulationAdapter extends java.lang.Object`
`   implements Simulation`

`java.lang.Object`
p. 64 `  fungus.core.infrastructure.SimulationAdapter`

Adapter pour la simualtion.
les démarage, arrêt, et nettoyage de la simulation sont implémentés.

### 5.10.1 Attributes

p. 89 `private fungus.core.service.InfoService is` :

### 5.10.2 Constructors

**`public SimulationAdapter(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

### 5.10.3 Methods

**`public void getParameters(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

**`public void declareParameters(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

**`public void buildAgent(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

**`public void startSimulation(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

**`public void stopSimulation(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

**`public void removeSimulation(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

## 5.11 Synchronization

```
public class Synchronization extends fungus.core.service.AbstractService
   implements SynchronizationService
```

```
java.lang.Object
  fungus.core.service.AbstractService
    fungus.core.infrastructure.Synchronization
```

p. 77
p. 65

Contient un semaphore.
**Attention** ; Contient qu'un semaphore.

### 5.11.1 Attributes

`private int sem` :

### 5.11.2 Constructors

**`public Synchronization()`**

### 5.11.3 Methods

**`public void init()`**

**`public int getNewSem()`**

**return** `int` :

**`public int getValue(int i)`**

**Parameters**

`int i` :

**return** `int` :

**`public synchronized int get(int i)`**

**Parameters**

`int i` :

**return** `int` :

**`public synchronized int revoke(int i)`**

**Parameters**

`int i` :

**return** `int` :

**`public synchronized void waitUntilZero(int i)`**

**Parameters**

`int i` :

# 6 fungus.core.scheduler

Implémentation des services *scheduler*.
Contient aussi les modèles temporels et de gestion du temps.

## 6.1 EventDrivenModel

```
public class EventDrivenModel extends java.lang.Object
   implements TimeModel
```

```
java.lang.Object
  fungus.core.scheduler.EventDrivenModel
```
p. 66

Modèle dirigé par les evenements.
L'increments temporel n'est fixé.

### 6.1.1 Attributes

p. 73 `private fungus.core.scheduler.Time time` : l'heure courante

### 6.1.2 Constructors

**`public EventDrivenModel()`**
Fabrique un nouveau `EventDrivenModel` d'increment fixé à 1.

### 6.1.3 Methods

**`public Time getCurrentTime()`**

p. 73 **return** `fungus.core.scheduler.Time` :

**`public void advanceInTime()`**

**`public Time nextTic()`**

p. 73 **return** `fungus.core.scheduler.Time` :

### 6.2 Pool

`public abstract class Pool extends java.lang.Object`
`implements TimeModel`

`java.lang.Object`
p. 67 `fungus.core.scheduler.Pool`

Un pool d'agent.
Les pools d'agents sont des ensembles d'agents qui seront geres par un Scheduler.

#### 6.2.1 Attributes

p. 136 `protected fungus.core.utils.hash.AgentIDHash pool` : Le pool d'agent à être gere. AgentID -> date reprise activité

p. 73 `protected fungus.core.scheduler.Time time` : l'heure courante

#### 6.2.2 Constructors

**`public Pool()`**

#### 6.2.3 Methods

**`public void addAgent(AgentID agent)`**
Ajoute un agent au pool.
L'heure de demarage est fixe a '0'.

**Parameters**

p. 28 `fungus.core.agent.AgentID agent` : an `AgentID` value

**`public void addAgent(AgentID agent, Time t)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID agent` :
p. 73 `fungus.core.scheduler.Time t` :

**`public boolean removeAgent(AgentID agent)`**
Enleve un agent du pool

**Parameters**

p. 28 `fungus.core.agent.AgentID agent` : l'agent à enlever

**return** `boolean` :
`true` si l'agnet à bien était enlevé

**`public Collection getSchedulableAgent(Time t)`**
Renvoie l'ensemble des agents qui peuvent etre traiter.

**Parameters**

p. 73 `fungus.core.scheduler.Time t` : l'heure limite de declenchement des agents.

**return** `java.util.Collection` :
un emsemble d'agent qui peuvent être traité par le scheduler.

**public void put(AgentID id, Time t)**

Ajoute un agent au pool

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : an `AgentID` value

p. 73 `fungus.core.scheduler.Time t` : a `Time` value

**public void setWakeupTime(AgentID id, Time time)**

Fixe l'heure de redémarage d'un agent

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : l'agent à traiter

p. 73 `fungus.core.scheduler.Time time` : l'heure de redémarage

**public Collection keySet()**

Renvoie l'ensemble des agents

**return** `java.util.Collection` :
a `Set` value

**public Time getCurrentTime()**

p. 73 **return** `fungus.core.scheduler.Time` :

**public abstract void advanceInTime()**

**public abstract Time nextTic()**

p. 73 **return** `fungus.core.scheduler.Time` :

## 6.3 Pool.SchedulableAgentFilter

`private class Pool.SchedulableAgentFilter extends java.lang.Object`
`implements AgentIDTimeProcedure`

`java.lang.Object`
p. 69 `fungus.core.scheduler.Pool.SchedulableAgentFilter`

Filtre qui donne les agents à scheduler

### 6.3.1 Attributes

`java.util.List list` :

p. 73 `fungus.core.scheduler.Time upperbound` :

### 6.3.2 Constructors

**public Pool.SchedulableAgentFilter(Pool this$0, Time t)**

**Parameters**

p. 67 `fungus.core.scheduler.Pool this$0` :
p. 73 `fungus.core.scheduler.Time t` :

### 6.3.3 Methods

**public boolean execute(Object a, Object b)**

**Parameters**

`java.lang.Object a` :
`java.lang.Object b` :

**return** `boolean` :

**public Collection getFilteredAgent()**
Donne la liste des agents à scheduler

**return** `java.util.Collection` :
a `Collection` value

## 6.4 Scheduler

`public class Scheduler extends fungus.core.service.AbstractService`
`implements SchedulerService, Runnable`

```
java.lang.Object
```
p. 77 `fungus.core.service.AbstractService`
p. 70 `fungus.core.scheduler.Scheduler`

Une simple implémentation de Scheduler.
En fait l'implémentation de base... Oui, mais elle gère quand même le reseau...

### 6.4.1 Attributes

p. 67 `private fungus.core.scheduler.Pool pool` : Le premier pool d'agent à être gere.
A terme une liste de pool

`private java.lang.Thread my` :

p. 89 `private fungus.core.service.InfoService is` :

p. 98 `private fungus.core.service.NetworkService nt` :

`public static final java.lang.String scheduleAgent` :
Ajouter un agent dans un scheduler

`public static final java.lang.String startScheduler` :
Demare le scheduler

`public static final java.lang.String endRun` :
Enoyer quand le scheduler a fini un "run"

`private boolean go` :
le thread continue de tourner si `true`

`private boolean stopped` :
on fait une pause si `true`

### 6.4.2 Constructors

**`public Scheduler()`**

### 6.4.3 Methods

**`public void init()`**

**`public void addAgent(AgentID id)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID id` :

**`public void addAgent(AgentID id, int pool)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID id` :
`int pool` :

**public void removeAgent(AgentID id)**

**Parameters**

p. 28 fungus.core.agent.AgentID id :

**public void start()**

**public void start(AgentID id)**

**Parameters**

p. 28 fungus.core.agent.AgentID id :

**public void start(int pool)**

**Parameters**

int pool :

**public void sleepUntil(AgentID id, Time date)**

**Parameters**

p. 28 fungus.core.agent.AgentID id :

p. 73 fungus.core.scheduler.Time date :

**public void sleepDuring(AgentID id, Time time)**

**Parameters**

p. 28 fungus.core.agent.AgentID id :

p. 73 fungus.core.scheduler.Time time :

**public void stop()**

**public void stop(AgentID id)**

**Parameters**

p. 28 fungus.core.agent.AgentID id :

**public void stop(int pool)**

**Parameters**

int pool :

**public void run()**

**private NetworkService network()**

p. 98 **return** fungus.core.service.NetworkService :

## 6.5 Scheduler.MessageListener

public static class Scheduler.MessageListener extends java.lang.Object
implements NetMessageListener

java.lang.Object
p. 72 fungus.core.scheduler.Scheduler.MessageListener

repond au message de scheduling par reseau

### 6.5.1 Attributes

p. 53 private fungus.core.infrastructure.Infrastructure infra :

p. 100 private fungus.core.service.SchedulerService sc :

public static int numberOfHosts :

### 6.5.2 Constructors

**public Scheduler.MessageListener(Infrastructure i)**

**Parameters**

p. 53 fungus.core.infrastructure.Infrastructure i :

### 6.5.3 Methods

**private SchedulerService scheduler()**

p. 100 **return** fungus.core.service.SchedulerService :

**public boolean performMessage(String type, ObjectInputStream in, NetConnection out)**

**Parameters**

java.lang.String type :

java.io.ObjectInputStream in :

p. 56 fungus.core.infrastructure.NetConnection out :

**return** boolean :

## 6.6 Time

`public class Time extends java.util.Date`

```
java.lang.Object
  java.util.Date
    fungus.core.scheduler.Time
```
p. 73

Représentation du temps.
Utilise entre autre par les schedulers.

### 6.6.1 Constructors

**public Time()**
Nouvelle heure.
L'heure est mise à 0.

**public Time(long init)**
Nouvelle heure.

**Parameters**

`long init` : valeur initiale de l'heure

**public Time(Time t)**

**Parameters**

p. 73 `fungus.core.scheduler.Time t` :

### 6.6.2 Methods

**public void advance(Time t)**
Avance l'heure du *t* unité

**Parameters**

p. 73 `fungus.core.scheduler.Time t` : le temps que l'on ajoute.

**public Time add(Time t)**
ajoute *t* unités à ce temps

**Parameters**

p. 73 `fungus.core.scheduler.Time t` : le nombre d'unité à ajouter

p. 73 **return** `fungus.core.scheduler.Time` : le nouveau temps

**public String toString()**

**return** `java.lang.String` :

## 6.7 `TimeDrivenModel`

`public class TimeDrivenModel extends java.lang.Object`
`implements TimeModel`

`java.lang.Object`
p. 74 `fungus.core.scheduler.TimeDrivenModel`

Modèle dirigé par le temps.
L'increments temporel est fixé un fois pour toute.

### 6.7.1 Attributes

p. 73 `private fungus.core.scheduler.Time time` : l'heure courante

p. 73 `private fungus.core.scheduler.Time inc` : l'incrément

### 6.7.2 Constructors

**`public TimeDrivenModel(Time inc)`**
Fabrique un nouveau `TimeDrivenModel` d'increment fixé.

**Parameters**

p. 73 `fungus.core.scheduler.Time inc` : l'incrément temporel

**`public TimeDrivenModel()`**
Fabrique un nouveau `TimeDrivenModel` d'increment fixé à 1.

### 6.7.3 Methods

**`public Time getCurrentTime()`**

p. 73 **return** `fungus.core.scheduler.Time` :

**`public void advanceInTime()`**

**`public Time nextTic()`**

p. 73 **return** `fungus.core.scheduler.Time` :

## 6.8 TimeDrivenPool

`public class TimeDrivenPool extends fungus.core.scheduler.Pool`

java.lang.Object
p. 67 fungus.core.scheduler.Pool
p. 75 fungus.core.scheduler.TimeDrivenPool

Modèle dirigé par le temps.
L'increments temporel est fixé un fois pour toute.

### 6.8.1 Attributes

p. 73 `private fungus.core.scheduler.Time inc` : l'incrément

p. 73 `private fungus.core.scheduler.Time nt` :

### 6.8.2 Constructors

**`public TimeDrivenPool(Time inc)`**
Fabrique un nouveau `TimeDrivenPool` d'increment fixé.

**Parameters**

p. 73 `fungus.core.scheduler.Time inc` : l'incrément temporel

**`public TimeDrivenPool()`**
Fabrique un nouveau `TimeDrivenPool` d'increment fixé à 1.

### 6.8.3 Methods

**`public void advanceInTime()`**

**`public Time nextTic()`**

p. 73 **return** `fungus.core.scheduler.Time` :

## 6.9 `TimeModel`

`public interface class TimeModel extends null`

`fungus.core.scheduler.TimeModel`

Modèle temporel.

### 6.9.1 Methods

**`public Time getCurrentTime()`**
Donne l'heure courante

p. 73 **return** `fungus.core.scheduler.Time` : l'heure courante

**`public void advanceInTime()`**
Avance le temps.
L'incrément temporel n'est pas forcement fixe. cela depend du modèle utilisé.

**`public Time nextTic()`**
Renvoie l'heure du prochain nullticnull d'horloge.
Si l'heure ne peut pas être connue renvoie '0'

p. 73 **return** `fungus.core.scheduler.Time` : l'heure du prochain nullticnull

# 7 fungus.core.service

Definit les services et contient la plupart des interfaces des services standards.

## 7.1 AbstractService

`public abstract class AbstractService extends java.lang.Object`
`implements Service, ServiceInformation`

`java.lang.Object`
p. 77 `fungus.core.service.AbstractService`

Classe abstraite pour les services
En général les servoces dérivent de cette classe

### 7.1.1 Attributes

`private java.lang.String type :`

`private java.lang.String name :`

`private java.lang.String author :`

`private java.lang.String version :`

`private java.util.Date date :`

p. 53 `protected fungus.core.infrastructure.Infrastructure infra :`

### 7.1.2 Constructors

**`public AbstractService(String type)`**
Fabrique un nouveau service

**Parameters**

`java.lang.String type` : type du service

### 7.1.3 Methods

**`public String getType()`**
Renvoie le type du service ie sa ressource

**return** `java.lang.String` :
le type du service

**`public void setInfrastructure(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i :`

**`public void init()`**
initialise le nouveau service

**`protected void setInformation(String name, String author, String version, Date date)`**
Renseigne les méta informations du service.

**Parameters**

`java.lang.String name` : le nom du service

`java.lang.String author` : l'auteur du service

`java.lang.String version` : la version du service

`java.util.Date date` : la date de dernière modification

**`public String getName()`**

Donne le nom du service

**return** `java.lang.String` :

**`public String getAuthor()`**

Donne l'auteur du service

**return** `java.lang.String` :

**`public String getVersion()`**

Donne la version du service

**return** `java.lang.String` :

**`public Date getDate()`**

Donne la date du service

**return** `java.util.Date` :

## 7.2 AgentInstanceMaker

`public interface class AgentInstanceMaker extends null`

`fungus.core.service.AgentInstanceMaker`

Instancie un agent.
Permet de delegue la creation soit a un Class.forName ou au (make ) de Kawa.

### 7.2.1 Methods

**`public Agent newInstance(String name) throws Exception`**
Renvoie une nouvelle instance de Agent.

**Parameters**

`java.lang.String name` : nom de la classe de l'agent

p. 26 **return** `fungus.core.agent.Agent` :

**Exception**

`java.lang.Exception` : if an error occurs

### 7.3 CreateAgent

`public class CreateAgent extends fungus.core.service.AbstractService`
`implements CreateAgentService, AgentInstanceMaker`

```
java.lang.Object
```
p. 77 `fungus.core.service.AbstractService`
p. 80 `fungus.core.service.CreateAgent`

Fabrique des agents. Ces agents sont toujours locau.

#### 7.3.1 Attributes

p. 89 `private fungus.core.service.InfoService is` :

p. 79 `private fungus.core.service.AgentInstanceMaker am` :

#### 7.3.2 Constructors

**`public CreateAgent()`**

#### 7.3.3 Methods

**`public void init()`**

**`public AgentID newAgent(String classname) throws CreateAgentException`**
Fabrique un nouvel agent.

**Parameters**

`java.lang.String classname` : nom de la classe de l'agent que l'on générera.

p. 28 **return** `fungus.core.agent.AgentID` : L'ID du nouvel agent

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` : if an error occurs

**`public AgentID newAgent(String classname, AgentID parent) throws CreateAgentException`**

**Parameters**

`java.lang.String classname` :
p. 28 `fungus.core.agent.AgentID parent` :

p. 28 **return** `fungus.core.agent.AgentID` :

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` :

**`public AgentID newAgent(String classname, ObjectHash p) throws CreateAgentException`**
Fabrique un nouvel agent

**Parameters**

`java.lang.String classname` : le nom de la classe de l'agent
p. 138 `fungus.core.utils.hash.ObjectHash p` : ObjectHash qui contient les possibles paramètres de l'agent utilise lors de son initialisation

p. 28 **return** `fungus.core.agent.AgentID` : L'ID du nouvel agent

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` : if an error occurs

**`public AgentID newAgent(String classname, ObjectHash p, AgentID parent) throws CreateAgentException`**

Fabrique un nouvel agent

**Parameters**

`java.lang.String classname` : le nom de la classe de l'agent

p. 138 `fungus.core.utils.hash.ObjectHash p` : ObjectHash qui contient les possibles paramètres de l'agent utiliser lors de son initialisation

p. 28 `fungus.core.agent.AgentID parent` : ID d'un eventuel parent de cet agent. Le parent devra être un AgentGroup.

p. 28 **return** `fungus.core.agent.AgentID` : L'ID du nouvel agent

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` : if an error occurs

**`public AgentID newAgent(String classname, ObjectHash p, AgentID parent, boolean local) throws CreateAgentException`**

**Parameters**

`java.lang.String classname` :

p. 138 `fungus.core.utils.hash.ObjectHash p` :

p. 28 `fungus.core.agent.AgentID parent` :

`boolean local` :

p. 28 **return** `fungus.core.agent.AgentID` :

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` :

**`public void setAgentMaker(AgentInstanceMaker a, boolean flag)`**

**Parameters**

p. 79 `fungus.core.service.AgentInstanceMaker a` :

`boolean flag` :

**`public Agent newInstance(String name) throws Exception`**

**Parameters**

`java.lang.String name` :

p. 26 **return** `fungus.core.agent.Agent` :

**Exception**

`java.lang.Exception` :

## 7.4 CreateAgentException

`public class CreateAgentException extends fungus.core.FungusException`

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      fungus.core.FungusException
        fungus.core.service.CreateAgentException
```
p. 22
p. 82

Throws if we cannot create an agent.

### 7.4.1 Constructors

**public CreateAgentException()**

**public CreateAgentException(String e)**

**Parameters**

`java.lang.String e` :

## 7.5 CreateAgentService

`public interface class CreateAgentService extends null`
`implements Service`

`fungus.core.service.CreateAgentService`

**Service :**

]ressource

createAgent ]role

Fabrique un nouvel agent.

### 7.5.1 Attributes

`public static final java.lang.String ressource` :

### 7.5.2 Methods

**`public AgentID newAgent(String classname) throws CreateAgentException`**
Fabrique un nouvel agent.

**Parameters**

`java.lang.String classname` : nom de la classe de l'agent que l'on générera.

p. 28 **return** `fungus.core.agent.AgentID` : L'ID du nouvel agent

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` : if an error occurs

**`public AgentID newAgent(String classname, AgentID parent) throws CreateAgentException`**

**Parameters**

`java.lang.String classname` :

p. 28 `fungus.core.agent.AgentID parent` :

p. 28 **return** `fungus.core.agent.AgentID` :

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` :

**`public AgentID newAgent(String classname, ObjectHash p) throws CreateAgentException`**
Fabrique un nouvel agent

**Parameters**

`java.lang.String classname` : le nom de la classe de l'agent

p. 138 `fungus.core.utils.hash.ObjectHash p` : ObjectHash qui contient les possibles paramètres de l'agent utilise lors de son initialisation

p. 28 **return** `fungus.core.agent.AgentID` : L'ID du nouvel agent

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` : if an error occurs

**public AgentID newAgent(String classname, ObjectHash p, AgentID parent) throws CreateAgentException**

Fabrique un nouvel agent

**Parameters**

`java.lang.String classname` : le nom de la classe de l'agent

p. 138 `fungus.core.utils.hash.ObjectHash p` : ObjectHash qui contient les possibles paramètres de l'agent utiliser lors de son initialisation

p. 28 `fungus.core.agent.AgentID parent` : ID d'un eventuel parent de cet agent. Le parent devra être un AgentGroup.

p. 28 **return** `fungus.core.agent.AgentID` : L'ID du nouvel agent

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` : if an error occurs

**public AgentID newAgent(String classname, ObjectHash p, AgentID parent, boolean local) throws CreateAgentException**

Fabrique un nouvel agent

**Parameters**

`java.lang.String classname` : le nom de la classe de l'agent

p. 138 `fungus.core.utils.hash.ObjectHash p` : ObjectHash qui contient les possibles paramètres de l'agent utiliser lors de son initialisation

p. 28 `fungus.core.agent.AgentID parent` : ID d'un eventuel parent de cet agent. Le parent devra être un AgentGroup.

`boolean local` : si `true` l'agent est local

p. 28 **return** `fungus.core.agent.AgentID` : L'ID du nouvel agent

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` : if an error occurs

**public void setAgentMaker(AgentInstanceMaker a, boolean flag)**

**Parameters**

p. 79 `fungus.core.service.AgentInstanceMaker a` :

`boolean flag` :

## 7.6 Info

public class Info extends fungus.core.service.AbstractService
   implements InfoService

java.lang.Object
p. 77 fungus.core.service.AbstractService
p. 85 fungus.core.service.Info

Serveur d'information.
Contient des informations sur l'ensembles des agents

### 7.6.1 Attributes

p. 136 private fungus.core.utils.hash.AgentIDHash **iDtoAgent** : AgentID -> Agent

p. 138 private fungus.core.utils.hash.ObjectHash **agentToID** : Agent -> AgentID

p. 138 private fungus.core.utils.hash.ObjectHash **iDIDtoCanal** : (AgentID,AgentID) -> Canal

public static final java.lang.String **newAgent** :
Signale qu'un nouvel agent à été crée. Message reseau

private long **agentCounter** :

private java.net.InetAddress **host** :

private int **hostkey** :

p. 26 private static fungus.core.agent.Agent **empty** :

### 7.6.2 Constructors

**public Info()**

### 7.6.3 Methods

**public AgentID recordAgent(Agent agent)**
Enregistre un agent. Accessoirement lui donne aussi un nom

**Parameters**

p. 26 fungus.core.agent.Agent agent : L'agent à enregistrer

p. 28 **return** fungus.core.agent.AgentID : le nom de l'agent

**public AgentID recordAgent(AgentID id)**
Enregistre un agent à partir de son ID.
Considère que l'agent est distant

**Parameters**

p. 28 fungus.core.agent.AgentID id : l'ID de l'agent

p. 28 **return** fungus.core.agent.AgentID : Le nouvel Agent ID

**public AgentID recordAgent(String id)**
Enregistre un agent à partir de la représentation textuelle de son ID.
Considère que l'agent est distant

**Parameters**

`java.lang.String id` : l'ID de l'agent

p. 28 **return** `fungus.core.agent.AgentID` : Le nouvel Agent ID

**`public void removeAgent(Agent agent)`**
efface un agent des tables

**Parameters**

p. 26 `fungus.core.agent.Agent agent` : an `Agent` value

**`public Agent getAgent(AgentID id) throws NullAgentException`**
Renvoie l'agent qui correspond à cet ID.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : a `String` value

p. 26 **return** `fungus.core.agent.Agent` : an `Agent` value

**Exception**

p. 35 `fungus.core.agent.NullAgentException` : si l'agent n'existe pas

**`public Collection getAgents()`**
Renvoie la liste des ID des agents

**return** `java.util.Collection` :
a `Vector` d'AgentID

**`public Collection getLocalAgents()`**
Donne la liste des agents locaux.

**return** `java.util.Collection` :
un `Vector` d'`AgentID`.

**`public InetAddress getHost()`**
Renvoie le nom de l'hôte local

**return** `java.net.InetAddress` :
a `String` value

**`public boolean isLocal(AgentID id)`**
Dit si un agent est local

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : L'agent à tester

**return** `boolean` :
true si il est local

**`public synchronized void recordCanal(AgentID a1, AgentID a2, Canal c)`**
Enregistre un canal
l'ordre des agents n'est pas importantes...

**Parameters**

p. 28 `fungus.core.agent.AgentID a1` : un premier agent du canal

p. 28 `fungus.core.agent.AgentID a2` : un second agent du canal

p. 41 `fungus.core.canal.Canal c` : le canal

**`public void removeCanal(AgentID a1, AgentID a2)`**

Enlève un canal

**Parameters**

p. 28 `fungus.core.agent.AgentID a1` : an `AgentID` value

p. 28 `fungus.core.agent.AgentID a2` : an `AgentID` value

**`public Canal getCanal(AgentID a1, AgentID a2)`**

donne le canal

**Parameters**

p. 28 `fungus.core.agent.AgentID a1` : an `AgentID` value

p. 28 `fungus.core.agent.AgentID a2` : an `AgentID` value

p. 41 **return** `fungus.core.canal.Canal` : a `Canal` value

**`public static NetMessageListener getNetMessageListener(Infrastructure i)`**

donne une instance de `NetMessageListener`

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` : an `Infrastructure` value

p. 59 **return** `fungus.core.infrastructure.NetMessageListener` : a `staTic` value

## 7.7 Info.MessageListener

private static class Info.MessageListener extends java.lang.Object
implements NetMessageListener

java.lang.Object
p. 88 fungus.core.service.Info.MessageListener

repond au message de creation d'agents par le reseau

### 7.7.1 Attributes

p. 53 private fungus.core.infrastructure.Infrastructure infra :

### 7.7.2 Constructors

**Info.MessageListener(Infrastructure i)**

**Parameters**

p. 53 fungus.core.infrastructure.Infrastructure i :

### 7.7.3 Methods

**public boolean performMessage(String type, ObjectInputStream in, NetConnection out)**

**Parameters**

java.lang.String type :

java.io.ObjectInputStream in :

p. 56 fungus.core.infrastructure.NetConnection out :

**return** boolean :

### 7.8 InfoService

`public interface class InfoService extends null`
`implements Service`

`fungus.core.service.InfoService`

**Service :**

]ressource

info ]role

Donne des informations sur l'ensemble du systeme.

#### 7.8.1 Attributes

`public static final java.lang.String ressource` :

#### 7.8.2 Methods

**`public AgentID recordAgent(Agent agent)`**
Enregistre un agent. Accessoirement lui donne aussi un nom

**Parameters**

p. 26 `fungus.core.agent.Agent agent` : L'agent à enregistrer

p. 28 **return** `fungus.core.agent.AgentID` : le nom de l'agent

**`public AgentID recordAgent(AgentID id)`**
Enregistre un agent à partir de son ID.
Considère que l'agent est distant

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : l'ID de l'agent

p. 28 **return** `fungus.core.agent.AgentID` : Le nouvel Agent ID

**`public AgentID recordAgent(String id)`**
Enregistre un agent à partir de la représentation textuelle de son ID.
Considère que l'agent est distant

**Parameters**

`java.lang.String id` : l'ID de l'agent

p. 28 **return** `fungus.core.agent.AgentID` : Le nouvel Agent ID

**`public void removeAgent(Agent agent)`**
efface un agent des tables

**Parameters**

p. 26 `fungus.core.agent.Agent agent` : an `Agent` value

**`public Agent getAgent(AgentID id) throws NullAgentException`**
Renvoie l'agent qui correspond à cet ID.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` :

p. 26 **return** `fungus.core.agent.Agent` : an `Agent` value

**Exception**

p. 35 `fungus.core.agent.NullAgentException` : si l'agent n'existe pas

**`public Collection getAgents()`**
Renvoie la liste des ID des agents

**return** `java.util.Collection` :
a `Collection` d'AgentID

**`public Collection getLocalAgents()`**
Donne la liste des agents locaux.

**return** `java.util.Collection` :
un `Collection` d'`AgentID`.

**`public InetAddress getHost()`**
Renvoie le nom de l'hôte local

**return** `java.net.InetAddress` :
a `String` value

**`public boolean isLocal(AgentID id)`**
Dit si un agent est local

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : L'agent à tester

**return** `boolean` :
true si il est local

**`public void recordCanal(AgentID a1, AgentID a2, Canal c)`**
Enregistre un canal
l'ordre des agents n'est pas importantes...

**Parameters**

p. 28 `fungus.core.agent.AgentID a1` : un premier agent du canal
p. 28 `fungus.core.agent.AgentID a2` : un second agent du canal
p. 41 `fungus.core.canal.Canal c` : le canal

**`public void removeCanal(AgentID a1, AgentID a2)`**
Enlève un canal

**Parameters**

p. 28 `fungus.core.agent.AgentID a1` : an `AgentID` value
p. 28 `fungus.core.agent.AgentID a2` : an `AgentID` value

**`public Canal getCanal(AgentID a1, AgentID a2)`**
donne le canal

**Parameters**

p. 28 `fungus.core.agent.AgentID a1` : an `AgentID` value

p. 28 `fungus.core.agent.AgentID a2` : an `AgentID` value

p. 41 **return** `fungus.core.canal.Canal` : a `Canal` value

## 7.9 `Network`

`public class Network extends fungus.core.service.AbstractService`
`implements NetworkService, Runnable`

`java.lang.Object`
p. 77 `fungus.core.service.AbstractService`
p. 92 `fungus.core.service.Network`

Serveur reseaux. Écoute le bon port et attend sagement les connections.
**!**\ la classe `NetMessageSelector` crée des dépendances non souhaitées

### 7.9.1 Attributes

`protected static int port` :

`private java.net.ServerSocket serverSocket` :

`private java.lang.Thread myThread` :

p. 138 `private fungus.core.utils.hash.ObjectHash connections` : Table des connections en cours. (InetAdress, NetConnection)

`private boolean again` :

### 7.9.2 Constructors

**`public Network()`**

### 7.9.3 Methods

**`public void init()`**

**`public void start()`**

**`public void run()`**

**`public void connect(Socket to)`**
Établie une connection vers un autre hôte

**Parameters**

`java.net.Socket to` : La socket vers l'hôte distant.

**`public NetConnection getConnection(InetAddress host)`**

**Parameters**

`java.net.InetAddress host` :

p. 56 **return** `fungus.core.infrastructure.NetConnection` :

**`public Collection getConnections()`**

**return** `java.util.Collection` :

**`public void removeConnection(String c)`**
Enlève une connection de la liste des connections

**Parameters**

`java.lang.String c` : la conenction a enlever.

**`public Set knowHosts()`**

Renvoie la liste des hôtes connus.

**return** `java.util.Set` :
Ennumeration de InetAddress qui contient les adresses IP des hôtes

**`public int numberOfHosts()`**

**return** `int` :

**`public Iterator getLocalAgents()`**

Renvoie la liste des AgentID.
Utiliser par NetConnection lors de l'initialisation de celui-ci..

**return** `java.util.Iterator` :
a `Vector` value

**`public Infrastructure getInfrastrusture()`**

p. 53 **return** `fungus.core.infrastructure.Infrastructure` :

**`public int getPort()`**

**return** `int` :

### 7.10 NetworkCreateAgent

public class NetworkCreateAgent extends fungus.core.service.AbstractService
implements CreateAgentService, AgentInstanceMaker

java.lang.Object
p. 77 fungus.core.service.AbstractService
p. 94 fungus.core.service.NetworkCreateAgent

Construit des agents a travers le reseau.

#### 7.10.1 Attributes

p. 89 private fungus.core.service.InfoService is :

p. 98 private fungus.core.service.NetworkService ns :

p. 105 private static fungus.core.service.SynchronizationService syn :

p. 109 private fungus.core.utils.CircularList hosts :

private java.util.Iterator hostIterator :

private static int synId :

p. 79 private fungus.core.service.AgentInstanceMaker am :

p. 28 private fungus.core.agent.AgentID netID :

public static final java.lang.String createAgent :
Signale que l'on veut créer un agent

public static final java.lang.String agentCreated :
Signale que l'on veut créer un agent

public static final java.lang.String agentFactory :
Envoie le nom de la factory qui instancie les agents.

#### 7.10.2 Constructors

**public NetworkCreateAgent()**

#### 7.10.3 Methods

**public void init()**

**public AgentID newAgent(String classname) throws CreateAgentException**
Fabrique un nouvel agent.

**Parameters**

java.lang.String classname : nom de la classe de l'agent que l'on générera.

p. 28 **return** fungus.core.agent.AgentID : L'ID du nouvel agent

**Exception**

p. 51 fungus.core.infrastructure.CreateAgentException : if an error occurs

**public AgentID newAgent(String classname, AgentID parent) throws CreateAgentException**

**Parameters**

`java.lang.String classname` :

p. 28 `fungus.core.agent.AgentID parent` :

p. 28 **return** `fungus.core.agent.AgentID` :

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` :

**public AgentID newAgent(String classname, ObjectHash p) throws CreateAgentException**

Fabrique un nouvel agent

**Parameters**

`java.lang.String classname` : le nom de la classe de l'agent

p. 138 `fungus.core.utils.hash.ObjectHash p` : ObjectHash qui contient les possibles paramètres de l'agent utilise lors de son initialisation

p. 28 **return** `fungus.core.agent.AgentID` : L'ID du nouvel agent

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` : if an error occurs

**public AgentID newAgent(String classname, ObjectHash p, AgentID parent) throws CreateAgentException**

Fabrique un nouvel agent

**Parameters**

`java.lang.String classname` : le nom de la classe de l'agent

p. 138 `fungus.core.utils.hash.ObjectHash p` : ObjectHash qui contient les possibles paramètres de l'agent utiliser lors de son initialisation

p. 28 `fungus.core.agent.AgentID parent` : ID d'un eventuel parent de cet agent. Le parent devra être un AgentGroup.

p. 28 **return** `fungus.core.agent.AgentID` : L'ID du nouvel agent

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` : if an error occurs

**public AgentID newAgent(String classname, ObjectHash p, AgentID parent, boolean local) throws CreateAgentException**

**Parameters**

`java.lang.String classname` :

p. 138 `fungus.core.utils.hash.ObjectHash p` :

p. 28 `fungus.core.agent.AgentID parent` :

`boolean local` :

p. 28 **return** `fungus.core.agent.AgentID` :

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` :

**`private AgentID newLocalAgent(String classname, ObjectHash p, AgentID parent) throws CreateAgentException`**

**Parameters**

`java.lang.String classname` :

p. 138 `fungus.core.utils.hash.ObjectHash p` :

p. 28 `fungus.core.agent.AgentID parent` :

p. 28 **return** `fungus.core.agent.AgentID` :

**Exception**

p. 51 `fungus.core.infrastructure.CreateAgentException` :

**`public void setAgentMaker(AgentInstanceMaker a, boolean flag)`**

**Parameters**

p. 79 `fungus.core.service.AgentInstanceMaker a` :

`boolean flag` :

**`public Agent newInstance(String name) throws Exception`**

**Parameters**

`java.lang.String name` :

p. 26 **return** `fungus.core.agent.Agent` :

**Exception**

`java.lang.Exception` :

**`private void setID(AgentID id)`**

Ruse pour passer le nom de l'agent creer sur un autre machine.
on fixe un champ qui sera renvoyer...

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : id de l'agent crée ailleurs

**`public static NetMessageListener getNetMessageListener(Infrastructure i)`**

donne une instance de `NetMessageListener`

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` : an `Infrastructure` value

p. 59 **return** `fungus.core.infrastructure.NetMessageListener` : a `staTic` value

### 7.11 NetworkCreateAgent.MessageListener

private static class NetworkCreateAgent.MessageListener extends java.lang.Object
implements NetMessageListener

java.lang.Object
p. 97 fungus.core.service.NetworkCreateAgent.MessageListener

repond au message de creation d'agents par le reseau

#### 7.11.1 Attributes

p. 53 private fungus.core.infrastructure.Infrastructure infra :

p. 83 private fungus.core.service.CreateAgentService ca :

#### 7.11.2 Constructors

**NetworkCreateAgent.MessageListener(Infrastructure i)**

**Parameters**

p. 53 fungus.core.infrastructure.Infrastructure i :

#### 7.11.3 Methods

**private void setCreateAgent()**

**public boolean performMessage(String type, ObjectInputStream in, NetConnection out)**

**Parameters**

java.lang.String type :

java.io.ObjectInputStream in :

p. 56 fungus.core.infrastructure.NetConnection out :

**return** boolean :

### 7.12 NetworkService

`public interface class NetworkService extends null`
`implements Service`

`fungus.core.service.NetworkService`

**Service :**

]ressource

network ]role

Attend et gere des connections reseaux

#### 7.12.1 Attributes

`public static final java.lang.String ressource` :

#### 7.12.2 Methods

**`public void start()`**
Demare le serveur

**`public int getPort()`**
Renvoie le port ecoute par ce server

**return** `int` :
le numero du port

**`public void connect(Socket to)`**
Établie une connection vers un autre hôte

**Parameters**

`java.net.Socket to` : La socket vers l'hôte distant.

**`public NetConnection getConnection(InetAddress host)`**
Renvoie la connection vers un hote donne.

**Parameters**

`java.net.InetAddress host` : Addresse de l'hote

p. 56 **return** `fungus.core.infrastructure.NetConnection` : Connection vers celui-ci ou `null` si elle n'existe pas.

**`public Collection getConnections()`**
Renvoie toutes les connections.

**return** `java.util.Collection` :
Une collection de NetConnection.

**`public void removeConnection(String c)`**
Enlève une connection de la liste des connections

**Parameters**

`java.lang.String c` : la conenction a enlever.

**`public int numberOfHosts()`**
Renvoie le nombre d'hote auquel on est connecte

**return** `int` :
nombre d'hote.

**`public Set knowHosts()`**
Renvoie la liste des hôtes connus.

**return** `java.util.Set` :
Ennumeration de InetAddress qui contient les adresses IP des hôtes

### 7.13 SchedulerService

`public interface class SchedulerService extends null`
`implements Service`

`fungus.core.service.SchedulerService`

**Service :**

]ressource

scheduler ]role

Gère les agents "leger"

Service Ordonnanceur.

#### 7.13.1 Attributes

`public static final java.lang.String ressource` :

#### 7.13.2 Methods

**`public void addAgent(AgentID id)`**
Enregistre un agent qui devra être scheduler.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : l'agent que l'on ajoute

**`public void addAgent(AgentID id, int pool)`**
Enregistre un agent qui devra être scheduler par un certain "pool".
Le pool est une "file d'attente". Cela permet de grouper des agents au niveau du schedeleur.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : l'agent que l'on ajoute

`int pool` : la file ou on place l'agent

**`public void removeAgent(AgentID id)`**
enleve un agent du scheduler.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : l'agent à enlever

**`public void start()`**
demare le scheduler.

**`public void start(AgentID id)`**
Démare un agent.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : l'agent à démarer

**`public void start(int pool)`**
Démare un pool d'agent.

**Parameters**

`int pool` : le pool à démarer

**public void sleepUntil(AgentID id, Time date)**

Met un agent en sommeil en spécifiant l'heure de reveil.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : l'agent qui dors

p. 73 `fungus.core.scheduler.Time date` : l'heure du reveil

**public void sleepDuring(AgentID id, Time time)**

Met un agent en sommeil pendant *time* unité de temps

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : l'agent qui va dormir

p. 73 `fungus.core.scheduler.Time time` : le temps de repos

**public void stop()**

arrete le schedeleur.

**public void stop(AgentID id)**

Arrete un agent.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : l'agent à arréter

**public void stop(int pool)**

Arrete un pool agent.

**Parameters**

`int pool` : l'agent à arréter

### 7.14 Service

`public interface class Service extends null`

`fungus.core.service.Service`

Un service.
Les services sont des éléments modulaires de l'infrastructure. (je sais ca veut rien dire)

#### 7.14.1 Methods

**`public String getType()`**
Renvoie la clé du service

**return** `java.lang.String` :
le type de service

**`public void setInfrastructure(Infrastructure i)`**
Spécifie l'infrastructure auquel se reporte le service

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` : l'infrastructure

**`public void init()`**
initialise le service.

### 7.15 ServiceException

public class ServiceException extends fungus.core.FungusException

```
        java.lang.Object
          java.lang.Throwable
            java.lang.Exception
p. 22         fungus.core.FungusException
p. 103          fungus.core.service.ServiceException
```

Throws if exception occur during Services

#### 7.15.1 Constructors

**public ServiceException()**

**public ServiceException(String e)**

**Parameters**

java.lang.String e :

## 7.16 `ServiceInformation`

`public interface class ServiceInformation extends null`

`fungus.core.service.ServiceInformation`

Permet d'avoir des "meta" informations sur un service.

### 7.16.1 Methods

**`public String getName()`**
Donne le nom du service

**return** `java.lang.String` :

**`public String getAuthor()`**
Donne l'auteur du service

**return** `java.lang.String` :

**`public String getVersion()`**
Donne la version du service

**return** `java.lang.String` :

**`public Date getDate()`**
Donne la date du service

**return** `java.util.Date` :

### 7.17 `SynchronizationService`

`public interface class SynchronizationService extends null implements Service`

`fungus.core.service.SynchronizationService`

**Service :**

]ressource

synchronization ]role

Offre des sémaphores utilisable comme on le souhaite.

#### 7.17.1 Attributes

`public static final java.lang.String ressource` :

#### 7.17.2 Methods

**`public int getNewSem()`**
Reserve un nouveau sémaphore

**return** `int` :
l'identifiant du semaphore que l'on vient d'allouer

**`public int getValue(int sem)`**
regarde la valeur du demaphore

**Parameters**

`int sem` : l'identifiant du semaphore

**return** `int` :

**`public int get(int sem)`**
prend une ressource

**Parameters**

`int sem` : l'identifiant du semaphore

**return** `int` :
la nouvelle valeur

**`public int revoke(int sem)`**
rend une ressource

**Parameters**

`int sem` : l'identifiant du semaphore

**return** `int` :
la nouvelle valeur

**`public void waitUntilZero(int sem)`**
Attend que toutes les ressources de ce semaphore soit rendues.

**Parameters**

`int sem` : l'identifiant du semaphore

## 7.18 Test

```
public class Test extends fungus.core.service.AbstractService
   implements TestService
```

```
java.lang.Object
  fungus.core.service.AbstractService
    fungus.core.service.Test
```

p. 77
p. 106

### 7.18.1 Constructors

**public Test()**

### 7.18.2 Methods

**public void test()**

## 7.19 TestService

```
public interface class TestService extends null
   implements Service
```

```
fungus.core.service.TestService
```

### 7.19.1 Methods

**public void test()**

## 7.20 VisualisationService

`public interface class VisualisationService extends null`
`implements Service`

`fungus.core.service.VisualisationService`

**Service :**

]ressource

visualisation ]role

Recoit ou va chercher les informations pour visualiser les simulations.

### 7.20.1 Attributes

`public static final java.lang.String ressource` :

### 7.20.2 Methods

**`public void addVisualisation(AgentID ag, Object type)`**
Ajoute une visualisation.

**Parameters**

p. 28 `fungus.core.agent.AgentID ag` : l'agent que l'on souhaite surveiller

`java.lang.Object type` : le type de visualisation.
**!**\ Pour l'instant, ce n'est absolument pas specifié.

**`public void addVisualisation(AgentID ag, Object type, Object arg)`**
Ajoute une visualisation.

**Parameters**

p. 28 `fungus.core.agent.AgentID ag` : l'agent que l'on souhaite surveiller

`java.lang.Object type` : le type de visualisation.
**!**\ Pour l'instant, ce n'est absolument pas specifié.

`java.lang.Object arg` : un argument passe à la visualisation. Par exemple, la proprieté que l'on veut visualiser dans l'agent.

# 8 fungus.core.utils

## 8.1 CircularList

```
public class CircularList extends java.util.ArrayList
```

```
java.lang.Object
  java.util.AbstractCollection
    java.util.AbstractList
      java.util.ArrayList
        fungus.core.utils.CircularList
```

p. 109

Une liste circulaire. Quand l' iterateur arrive a la fin de la liste, il repart au debut.

### 8.1.1 Constructors

**public CircularList()**

**public CircularList(Collection c)**

**Parameters**

`java.util.Collection c` :

**public CircularList(int initialCapacity)**

**Parameters**

`int initialCapacity` :

### 8.1.2 Methods

**public Iterator iterator()**

**return** `java.util.Iterator` :

### 8.2 CircularList.CircularIterator

`private class CircularList.CircularIterator extends java.lang.Object implements Iterator`

`java.lang.Object`
p. 110 `fungus.core.utils.CircularList.CircularIterator`

Describe class `CircularIterator` here.

#### 8.2.1 Attributes

p. 109 `private fungus.core.utils.CircularList l` :

`private int i` :

#### 8.2.2 Constructors

**`CircularList.CircularIterator(CircularList this$0, CircularList c)`**

**Parameters**

p. 109 `fungus.core.utils.CircularList this$0` :
p. 109 `fungus.core.utils.CircularList c` :

#### 8.2.3 Methods

**`public boolean hasNext()`**

**return** `boolean` :
always `true`

**`public Object next()`**
Describe `next` method here.

**return** `java.lang.Object` :
an `Object` value

**`public void remove()`**
Describe `remove` method here.

## 8.3 FigGraphics

`public class FigGraphics extends java.awt.Graphics`

```
java.lang.Object
  java.awt.Graphics
    fungus.core.utils.FigGraphics
```
p. 111

FIG Graphics
This is a graphics subclass that produce a FIG 3.2.
Usage :

```
import fungus.core.util.FigGraphics;
...
Graphics fig = new FigGraphics();

fig.setColor(Color.red);
fig.fillRect(10, 10, 30, 30);
```

– 1 pixel = 0.02222 mm
– 45 px = 1mm
page layout : A4 landscape

### 8.3.1 Attributes

`public static final boolean DEBUG` :

`protected static final int PAGEHEIGHT` :

`protected static final int PAGEWIDTH` :

`protected int xOffset` :

`protected int yOffset` :

`protected java.io.PrintWriter os` :

`public static final int cDefault` :

`public static final int cBlack` :

`public static final int cBlue` :

`public static final int cGreen` :

`public static final int cCyan` :

`public static final int cRed` :

`public static final int cMagenta` :

`public static final int cYellow` :

`public static final int cWhite` :

`protected java.awt.Color clr` :
The current color

protected int pen_color :

protected int fill_color :

protected java.awt.Font font :
The current font

protected java.awt.Rectangle clippingRect :

protected int line_style :

public static final int lSDefault :

public static final int lSSolid :

public static final int lSDashed :

public static final int lSDotted :

public static final int lSDash_dotted :

public static final int lSDash_double_dotted :

public static final int lSDash_triple_dotted :

public static final int afNotFilled :

public static final int afFullSaturation :

protected int depth :

protected int area_fill :

protected float style_val :

protected int join_style :

public static final int jMiter :

public static final int jBevel :

public static final int jRound :

protected int cap_style :

public static final int csButt :

public static final int csRound :

public static final int csPrjecting :

protected int radius :

protected int forward_arrow :

`protected int backward_arrow :`

`public static final int aOff :`

`public static final int aOn :`

#### 8.3.2 Constructors

**`public FigGraphics()`**

**`public FigGraphics(Writer w)`**

**Parameters**

`java.io.Writer w` :

**`public FigGraphics(Writer w, boolean emitProlog)`**

**Parameters**

`java.io.Writer w` :

`boolean emitProlog` :

#### 8.3.3 Methods

**`public Graphics create()`**

**return** `java.awt.Graphics` :

**`public void setOutput(Writer w)`**
Change the Writer this context's output goes to.

**Parameters**

`java.io.Writer w` :

**`private void preparePolyline(int sub_type)`**

**Parameters**

`int sub_type` :

**`private void prepareOval(int sub_type)`**

**Parameters**

`int sub_type` :

**`private int offx(int x)`**

**Parameters**

`int x` :

**return** `int` :

**`private int offy(int y)`**

**Parameters**

`int y` :

**return** `int` :

**`private int colorToColor(Color c)`**

**Parameters**

`java.awt.Color c` :

**return** `int` :

**`public void translate(int x, int y)`**

**Parameters**

`int x` :

`int y` :

**`public Color getColor()`**

**return** `java.awt.Color` :

**`public void setBackground(Color c)`**

**Parameters**

`java.awt.Color c` :

**`public void setColor(Color c)`**

**Parameters**

`java.awt.Color c` :

**`public void setPaintMode()`**

**`public void setXORMode(Color c1)`**

**Parameters**

`java.awt.Color c1` :

**`public Font getFont()`**

**return** `java.awt.Font` :

**`public void setFont(Font f)`**

**Parameters**

`java.awt.Font f` :

**`public FontMetrics getFontMetrics()`**

**return** `java.awt.FontMetrics` :

**public FontMetrics getFontMetrics(Font f)**

**Parameters**

java.awt.Font f :

**return** java.awt.FontMetrics :

**public Rectangle getClipRect()**

**return** java.awt.Rectangle :

**public Shape getClip()**

**return** java.awt.Shape :

**public void setClip(Shape s)**

**Parameters**

java.awt.Shape s :

**public Rectangle getClipBounds()**

**return** java.awt.Rectangle :

**public void setClip(int x, int y, int width, int height)**

**Parameters**

int x :

int y :

int width :

int height :

**public void clipRect(int x, int y, int width, int height)**

**Parameters**

int x :

int y :

int width :

int height :

**public void copyArea(int x, int y, int width, int height, int dx, int dy)**

**Parameters**

int x :

int y :

int width :

int height :

int dx :

int dy :

**public void drawLine(int x1, int y1, int x2, int y2)**

**Parameters**

int x1 :

int y1 :

int x2 :

int y2 :

**private void rectangle(int x, int y, int width, int height)**

**Parameters**

int x :

int y :

int width :

int height :

**private void rectangle(int x, int y, int width, int height, int s)**

**Parameters**

int x :

int y :

int width :

int height :

int s :

**public void fillRect(int x, int y, int width, int height)**

**Parameters**

int x :

int y :

int width :

int height :

**public void drawRect(int x, int y, int width, int height)**

**Parameters**

int x :

int y :

int width :

int height :

**public void clearRect(int x, int y, int width, int height)**

**Parameters**

int x :

int y :

int width :

int height :

**public void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)**

**Parameters**

int x :

int y :

int width :

int height :

int arcWidth :

int arcHeight :

**public void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)**

**Parameters**

int x :

int y :

int width :

int height :

int arcWidth :

int arcHeight :

**public void draw3DRect(int x, int y, int width, int height, boolean raised)**

**Parameters**

int x :

int y :

int width :

int height :

boolean raised :

**public void fill3DRect(int x, int y, int width, int height, boolean raised)**

**Parameters**

int x :

int y :

int width :

int height :

boolean raised :

**public void drawOval(int x, int y, int width, int height)**

**Parameters**

int x :

int y :

int width :

int height :

**public void fillOval(int x, int y, int width, int height)**

**Parameters**

int x :

int y :

int width :

int height :

**protected void doArc(int x, int y, int width, int height, int startAngle, int arcAngle, boolean fill)**

**Parameters**

int x :

int y :

int width :

int height :

int startAngle :

int arcAngle :

boolean fill :

**public void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)**

**Parameters**

int x :

int y :

int width :

int height :

int startAngle :

int arcAngle :

**public void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)**

**Parameters**

int x :

int y :

int width :

int height :

int startAngle :

int arcAngle :

**public void drawPolyline(int [] xPoints, int [] yPoints, int nPoints)**

**Parameters**

int [] xPoints :

int [] yPoints :

int nPoints :

**public void drawPolygon(int [] xPoints, int [] yPoints, int nPoints)**

**Parameters**

`int [] xPoints` :

`int [] yPoints` :

`int nPoints` :

**`public void drawPolygon(Polygon p)`**

**Parameters**

`java.awt.Polygon p` :

**`public void fillPolygon(int [] xPoints, int [] yPoints, int nPoints)`**

**Parameters**

`int [] xPoints` :

`int [] yPoints` :

`int nPoints` :

**`public void fillPolygon(Polygon p)`**

**Parameters**

`java.awt.Polygon p` :

**`public void drawString(AttributedCharacterIterator ai, int x, int y)`**

**Parameters**

`java.text.AttributedCharacterIterator ai` :

`int x` :

`int y` :

**`public void drawString(String str, int x, int y)`**

**Parameters**

`java.lang.String str` :

`int x` :

`int y` :

**`public void drawChars(char [] data, int offset, int length, int x, int y)`**

**Parameters**

`char [] data` :

`int offset` :

`int length` :

`int x` :

`int y` :

**`public void drawBytes(byte [] data, int offset, int length, int x, int y)`**

**Parameters**

`byte [] data` :

`int offset` :

`int length` :

`int x` :

`int y` :

**protected boolean doImage(Image img, int x, int y, int width, int height, int sx, int sy, int sw, int sh, ImageObserver observer, Color bgcolor)**

**Parameters**

`java.awt.Image img` :

`int x` :

`int y` :

`int width` :

`int height` :

`int sx` :

`int sy` :

`int sw` :

`int sh` :

`java.awt.image.ImageObserver observer` :

`java.awt.Color bgcolor` :

**return** `boolean` :

**public boolean drawImage(Image img, int x, int y, ImageObserver observer)**

**Parameters**

`java.awt.Image img` :

`int x` :

`int y` :

`java.awt.image.ImageObserver observer` :

**return** `boolean` :

**public boolean drawImage(Image img, int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4, ImageObserver observer)**

**Parameters**

`java.awt.Image img` :

`int x1` :

`int y1` :

`int x2` :

`int y2` :

`int x3` :

`int y3` :

`int x4` :

`int y4` :

`java.awt.image.ImageObserver observer` :

**return** boolean :

**public boolean drawImage(Image img, int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4, Color c, ImageObserver observer)**

**Parameters**

java.awt.Image img :

int x1 :

int y1 :

int x2 :

int y2 :

int x3 :

int y3 :

int x4 :

int y4 :

java.awt.Color c :

java.awt.image.ImageObserver observer :

**return** boolean :

**public boolean drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)**

**Parameters**

java.awt.Image img :

int x :

int y :

int width :

int height :

java.awt.image.ImageObserver observer :

**return** boolean :

**public boolean drawImage(Image img, int x, int y, Color bgcolor, ImageObserver observer)**

**Parameters**

java.awt.Image img :

int x :

int y :

java.awt.Color bgcolor :

java.awt.image.ImageObserver observer :

**return** boolean :

**public boolean drawImage(Image img, int x, int y, int width, int height, Color bgcolor, ImageObserver observer)**

**Parameters**

java.awt.Image img :

int x :

int y :

int width :

int height :

java.awt.Color bgcolor :

java.awt.image.ImageObserver observer :

**return** boolean :

**public void dispose()**

**public void finalize()**

**public String toString()**

**return** java.lang.String :

**protected void emitProlog()**

**protected void diagnostic(String s)**

**Parameters**

java.lang.String s :

## 8.4 IntAccumulator

`public class IntAccumulator extends java.lang.Object`

```
java.lang.Object
  fungus.core.utils.IntAccumulator
```

p. 123

Acculuiateur d'entier. Cela se comporte plus ou moins comme un ArrayList mais en plus repide. Dispose d'un mécanisme similaire aux iterateurs pour parcourir un intervalle.

### 8.4.1 Attributes

`private int capacity` :
Capacité d'une cellule.

`private java.util.ArrayList li` :

`private int [] current` :

`private int currentIndex` :

`private int size` :

`private int min` :

`private int max` :

`private int index` :

`private int endIter` :

`private int [] currentIter` :

`private int currentIndexIter` :

### 8.4.2 Constructors

**`public IntAccumulator()`**

### 8.4.3 Methods

**`public void add(int x)`**
Ajoute un entier a l'accumulateur.

**Parameters**

`int x` :

**`public int getSize()`**
Donne la taille de l'accumulateur

**return** `int` :

**`public int getMin()`**
Renvoie la valeur minimale accumulée

**return** `int` :

**public int getMax()**
Renvoie la valeur maximale accumulée

**return** `int` :

**public int get(int index)**
Donne un élément de l'accumulateur

**Parameters**

`int index` : la position de l'élément dans l'accumulateur

**return** `int` :

**public IntAccumulator getRangeIterator(int start, int end)**
Donne un iterateur sur l'intervalle passer en argument

**Parameters**

`int start` : le début de l'intervalle

`int end` : la fin de l'intervalle

p. 123 **return** `fungus.core.utils.IntAccumulator` :

**public boolean again()**
Il reste des éléments dans l'iterateur ?

**return** `boolean` :
`true` si il reste des elements

**public int next()**
renvoie le prochain éléments de l'iterateur

**return** `int` :

### 8.5 ParametersCollector

`public class ParametersCollector extends java.lang.Object`

`java.lang.Object`
p. 125 `fungus.core.utils.ParametersCollector`

Classe qui sert à collecter les paramètres de la simulations.
Wrapper vers l'interface la plus apprie (AWT standalone ou JPanel pour la GUI)

#### 8.5.1 Attributes

`private static java.lang.Class paramsClass` :

p. 130 `private fungus.core.utils.ParametersCollectorInterface params` :

#### 8.5.2 Constructors

**`public ParametersCollector(String title)`**
Creates a new `ParametersCollector` instance.

**Parameters**

`java.lang.String title` : Titre de la fenêtre.

#### 8.5.3 Methods

**`public static void setParametersCollectorMethod(Class m)`**

**Parameters**

`java.lang.Class m` :

**`public void addParam(String title)`**
Ajoute un champ paramètre à la fenêtre

**Parameters**

`java.lang.String title` : le nom du champ

**`public void addParam(String title, String defaut)`**
Ajoute un champ paramètre à la fenêtre

**Parameters**

`java.lang.String title` : le nom du champ
`java.lang.String defaut` :

**`public void collect()`**
Attend que l'on ai remplit les champs.

**`public boolean getBoolean(String title)`**
Renvoie la valeur de ce champ sous la forme d'un boolean

**Parameters**

`java.lang.String title` : le nom du champ

**return** `boolean` :
la valeur du champ (true, autre)

**`public int getInt(String title)`**

Renvoie la valeur de ce champ sous la forme d'un int

**Parameters**

`java.lang.String title` : le nom du champ

**return** `int` :
la valeur du champ

**`public long getLong(String title)`**

Renvoie la valeur de ce champ sous la forme d'un long

**Parameters**

`java.lang.String title` : le nom du champ

**return** `long` :
la valeur du champ

**`public float getFloat(String title)`**

Renvoie la valeur de ce champ sous la forme d'un float

**Parameters**

`java.lang.String title` : le nom du champ

**return** `float` :
la valeur du champ

**`public double getDouble(String title)`**

Renvoie la valeur de ce champ sous la forme d'un double

**Parameters**

`java.lang.String title` : le nom du champ

**return** `double` :
la valeur du champ

**`public String getString(String title)`**

Renvoie la valeur de ce champ sous la forme d'un String

**Parameters**

`java.lang.String title` : le nom du champ

**return** `java.lang.String` :
la valeur du champ

### 8.6 ParametersCollectorAWT

```
public class ParametersCollectorAWT extends java.awt.Frame
   implements ActionListener, ParametersCollectorInterface
```

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          fungus.core.utils.ParametersCollectorAWT
```
p. 127

Classe qui sert à collecter les paramètres de la simulations.
Ouvre une fenêtre AWT qui propose de remplir les champs de la simulation.

#### 8.6.1 Attributes

`private java.awt.Panel par` :

p. 138
`private fungus.core.utils.hash.ObjectHash t` :

`private int nbparam` :

`private boolean lock` :

#### 8.6.2 Constructors

**`public ParametersCollectorAWT()`**
Creates a new `ParametersCollector` instance.

#### 8.6.3 Methods

**`public void setTitle(String title)`**

**Parameters**

`java.lang.String title` :

**`public void addParam(String title)`**
Ajoute un champ paramètre à la fenêtre

**Parameters**

`java.lang.String title` : le nom du champ

**`public void addParam(String title, String defaut)`**
Ajoute un champ paramètre à la fenêtre

**Parameters**

`java.lang.String title` : le nom du champ

`java.lang.String defaut` :

**`public void collect()`**
Attend que l'on ai remplit les champs.

**`public boolean getBoolean(String title)`**
Renvoie la valeur de ce champ sous la forme d'un boolean

**Parameters**

`java.lang.String title` : le nom du champ

**return** `boolean` :
la valeur du champ (true, autre)

**`public int getInt(String title)`**
Renvoie la valeur de ce champ sous la forme d'un int

**Parameters**

`java.lang.String title` : le nom du champ

**return** `int` :
la valeur du champ

**`public long getLong(String title)`**
Renvoie la valeur de ce champ sous la forme d'un long

**Parameters**

`java.lang.String title` : le nom du champ

**return** `long` :
la valeur du champ

**`public float getFloat(String title)`**
Renvoie la valeur de ce champ sous la forme d'un float

**Parameters**

`java.lang.String title` : le nom du champ

**return** `float` :
la valeur du champ

**`public double getDouble(String title)`**
Renvoie la valeur de ce champ sous la forme d'un double

**Parameters**

`java.lang.String title` : le nom du champ

**return** `double` :
la valeur du champ

**`public String getString(String title)`**
Renvoie la valeur de ce champ sous la forme d'un String

**Parameters**

`java.lang.String title` : le nom du champ

**return** `java.lang.String` :
la valeur du champ

**`public void actionPerformed(ActionEvent e)`**

**Parameters**

`java.awt.event.ActionEvent e :`

## 8.7 `ParametersCollectorInterface`

`public interface class ParametersCollectorInterface extends null`

`fungus.core.utils.ParametersCollectorInterface`

Interface qui sert à collecter les paramètres de la simulations.

### 8.7.1 Methods

**`public void setTitle(String title)`**
Donne un tire à la fenetre.

**Parameters**

`java.lang.String title` : a `String` value

**`public void addParam(String title)`**
Ajoute un champ paramètre à la fenêtre

**Parameters**

`java.lang.String title` : le nom du champ

**`public void addParam(String title, String defaut)`**
Ajoute un champ paramètre à la fenêtre

**Parameters**

`java.lang.String title` : le nom du champ

`java.lang.String defaut` :

**`public void collect()`**
Attend que l'on ai remplit les champs.

**`public boolean getBoolean(String title)`**
Renvoie la valeur de ce champ sous la forme d'un boolean

**Parameters**

`java.lang.String title` : le nom du champ

**return** `boolean` :
la valeur du champ (true, autre)

**`public int getInt(String title)`**
Renvoie la valeur de ce champ sous la forme d'un int

**Parameters**

`java.lang.String title` : le nom du champ

**return** `int` :
la valeur du champ

**`public long getLong(String title)`**
Renvoie la valeur de ce champ sous la forme d'un long

**Parameters**

`java.lang.String title` : le nom du champ

**return** `long` :
la valeur du champ

**`public float getFloat(String title)`**
Renvoie la valeur de ce champ sous la forme d'un float

**Parameters**

`java.lang.String title` : le nom du champ

**return** `float` :
la valeur du champ

**`public double getDouble(String title)`**
Renvoie la valeur de ce champ sous la forme d'un double

**Parameters**

`java.lang.String title` : le nom du champ

**return** `double` :
la valeur du champ

**`public String getString(String title)`**
Renvoie la valeur de ce champ sous la forme d'un String

**Parameters**

`java.lang.String title` : le nom du champ

**return** `java.lang.String` :
la valeur du champ

## 8.8 Shell

public class Shell extends java.lang.Thread

```
java.lang.Object
  java.lang.Thread
    fungus.core.utils.Shell
```
p. 132

Un shell de contrôle pour Fungus

### 8.8.1 Attributes

p. 53 private fungus.core.infrastructure.Infrastructure infra :

private java.lang.String prompt :

private java.lang.String welcome :

### 8.8.2 Constructors

**public Shell(Infrastructure i)**

**Parameters**

p. 53 fungus.core.infrastructure.Infrastructure i :

### 8.8.3 Methods

**public void run()**

## 8.9 Util

`public class Util extends java.lang.Object`

`java.lang.Object`
p. 133 `fungus.core.utils.Util`

Classe d'utilitaire

### 8.9.1 Attributes

`public static final int information` :
Message de debuggage : Information

`public static final int service` :
Message de debuggage : Message des services

`public static final int protocol` :
Message de debuggage : Message de communication

`public static final int debug_` :
Message de debuggage : Message réellement de debug

`public static final int mycelium` :
Message de debuggage : Message mycelium

`public static final int tmp` :
Message de debuggage : Message temporaire...

`protected static boolean debug` :
si 'true' alors on affiche les messages de debug

`protected static int debugLevel` :
Masque pour l'affichage des niveau de debuggage

`private static java.util.Properties prop` :

`private static java.lang.String opts` :
Contient les options acceptables

`private static java.lang.String help` :
Contient l'aide des options

p. 138 `private static fungus.core.utils.hash.ObjectHash params` : Contient les paramètres.

`public static java.lang.String [] args` :

### 8.9.2 Constructors

**`public Util()`**

### 8.9.3 Methods

**`public static void loadProperties(String rsc)`**
Charge les propriétés

**Parameters**

`java.lang.String rsc` : le nom de la ressource à charger.

**`public static String getProperty(String classs, String proper)`**

Renvoie une propriété.

**Parameters**

`java.lang.String classs` : classe de la propriété

`java.lang.String proper` : propriété

**return** `java.lang.String` :
valeur de la propriété.

**`public static void printDebug(String mesg)`**

Affiche un message de debugage

**Parameters**

`java.lang.String mesg` : a `String` value

**`public static void printDebug(String mesg, int level)`**

Affiche un message de debugage d'un type donnée

**Parameters**

`java.lang.String mesg` : le message à afficher

`int level` : an `int` value

**`public static String [] parseCommandLine(String [] args, Infrastructure infra)`**

Parse la ligne de commande

**Parameters**

`java.lang.String [] args` : la ligne d'arguments

p. 53 `fungus.core.infrastructure.Infrastructure infra` : l'infrastructure du système.

**return** `java.lang.String` :
les arguments

**`public static void addOption(char letter, boolean haveOption, String help)`**

Ajoute une option à la ligne de commande. `!\` Cette méthode ne fait pas ce qu'on veut...

**Parameters**

`char letter` : a `char` value

`boolean haveOption` : a `boolean` value

`java.lang.String help` : a `String` value

**`public static boolean optionSet(char option)`**

Dit si l'option à été passe en argument

**Parameters**

`char option` : a `char` value

**return** `boolean` :

**public static String getOptionArg(char option)**

Renvoie la valeur d'une option de la ligne de commande

**Parameters**

`char option` : l'option a vérifier

**return** `java.lang.String` :

**public static void setOptionArg(char option, String args)**

Modifie une option de la ligne de commande.

**Parameters**

`char option` : l'option a modifier

`java.lang.String args` : la nouvelle valeur.

# 9 fungus.core.utils.hash

Contient des tables de hachages pour Fungus.
Ces tables sont rapides que celles proposées en standard par Java.

## 9.1 AgentIDHash

`public class AgentIDHash extends gnu.trove.THashMap`

```
java.lang.Object
  gnu.trove.TObjectHash
    gnu.trove.THashMap
      fungus.core.utils.hash.AgentIDHash
```
p. 136

Table de hash pour les AgentID

### 9.1.1 Constructors

**public AgentIDHash()**

### 9.1.2 Methods

**public Object put(AgentID key, Object value)**

**Parameters**

p. 28 `fungus.core.agent.AgentID key` :

`java.lang.Object value` :

**return** `java.lang.Object` :

**public Object get(AgentID key)**

**Parameters**

p. 28 `fungus.core.agent.AgentID key` :

**return** `java.lang.Object` :

**public Object remove(AgentID key)**

**Parameters**

p. 28 `fungus.core.agent.AgentID key` :

**return** `java.lang.Object` :

## 9.2 AgentIDTimeProcedure

```
public interface class AgentIDTimeProcedure extends null
   implements TObjectObjectProcedure
```

```
fungus.core.utils.hash.AgentIDTimeProcedure
```

Simple extension de l'interface de Trove.
Cela nous permet de garder une certaine indépendance vis à vis de Trove.

## 9.3 ObjectHash

```
public class ObjectHash extends gnu.trove.THashMap
   implements Serializable, TObjectObjectProcedure
```

```
java.lang.Object
  gnu.trove.TObjectHash
    gnu.trove.THashMap
      fungus.core.utils.hash.ObjectHash
```
p. 138

Table de hash pour des objets quelquonques

### 9.3.1 Attributes

`private java.io.ObjectOutputStream out` :

### 9.3.2 Constructors

**`public ObjectHash()`**

### 9.3.3 Methods

**`private void writeObject(ObjectOutputStream out) throws IOException`**

**Parameters**

`java.io.ObjectOutputStream out` :

**Exception**

`java.io.IOException` :

**`private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException`**

**Parameters**

`java.io.ObjectInputStream in` :

**Exception**

`java.io.IOException` :

`java.lang.ClassNotFoundException` :

**`public String toString()`**

**return** `java.lang.String` :

**`public boolean execute(Object a, Object b)`**

**Parameters**

`java.lang.Object a` :

`java.lang.Object b` :

**return** `boolean` :

# 10 `fungus.gui`

L'interface utilisateur de Fungus.

## 10.1 `About`

`public class About extends javax.swing.JDialog`

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Dialog
          javax.swing.JDialog
            fungus.gui.About
```
p. 139

### 10.1.1 Attributes

`private javax.swing.JButton jButton1 :`

`private javax.swing.JLabel jLabel1 :`

`private javax.swing.JPanel jPanel1 :`

### 10.1.2 Constructors

**`public About(Frame parent, boolean modal)`**
Creates new form About

**Parameters**

`java.awt.Frame parent` :

`boolean modal` :

### 10.1.3 Methods

**`private void initComponents()`**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**`private void jButton1ActionPerformed(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt` :

**`private void closeDialog(WindowEvent evt)`**
Closes the dialog

**Parameters**

`java.awt.event.WindowEvent evt` :

## 10.2 ExceptionManager

`public class ExceptionManager extends java.lang.Object`
`implements ExceptionManagerInterface`

`java.lang.Object`
p. 140 `fungus.gui.ExceptionManager`

Gestionnaire d'Exceptions de l'interface grpahique

### 10.2.1 Attributes

`private static java.awt.Frame parent` :

### 10.2.2 Constructors

**`public ExceptionManager(Frame root)`**
Creates a new `ExceptionManager` instance.

**Parameters**

`java.awt.Frame root` : a `Frame` value

### 10.2.3 Methods

**`public void raisei(String msg, Throwable e)`**

**Parameters**

`java.lang.String msg` :

`java.lang.Throwable e` :

**`public void raisei(String msg, Throwable e, int type)`**

**Parameters**

`java.lang.String msg` :

`java.lang.Throwable e` :

`int type` :

## 10.3 ExceptionPanel

public class ExceptionPanel extends javax.swing.JDialog

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Dialog
          javax.swing.JDialog
            fungus.gui.ExceptionPanel
```

p. 141

### 10.3.1 Attributes

private javax.swing.JButton jButton2 :

private javax.swing.JButton jButton1 :

private javax.swing.JScrollPane jScrollPane1 :

private javax.swing.JLabel jLabel2 :

private javax.swing.JLabel jLabel1 :

private javax.swing.JTextArea jTextArea2 :

private javax.swing.JPanel jPanel2 :

private javax.swing.JPanel jPanel1 :

### 10.3.2 Constructors

**public ExceptionPanel(Frame parent, boolean modal, String msg, String emsg)**
Creates new form ExceptionPanel

**Parameters**

java.awt.Frame parent :

boolean modal :

java.lang.String msg :

java.lang.String emsg :

### 10.3.3 Methods

**private void initComponents()**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**private void showTrace(ActionEvent evt)**

**Parameters**

java.awt.event.ActionEvent evt :

**private void okb(ActionEvent evt)**

**Parameters**

`java.awt.event.ActionEvent evt :`

**`private void closeDialog(WindowEvent evt)`**

Closes the dialog

**Parameters**

`java.awt.event.WindowEvent evt :`

## 10.4 Fungus

`public class Fungus extends javax.swing.JFrame`

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          javax.swing.JFrame
            fungus.gui.Fungus
```
p. 143

### 10.4.1 Attributes

p. 53 `protected static fungus.core.infrastructure.Infrastructure infra` :

p. 160 `private static fungus.gui.SharingSpace sspc` :

### 10.4.2 Constructors

**`public Fungus()`**
Creates new form Gui

### 10.4.3 Methods

**`public static void main(String [] args)`**

**Parameters**

`java.lang.String [] args` : the command line arguments

**`public static SharingSpace getSharingSpace()`**
Renvoie un espace d'échange.

p. 160 **return** `fungus.gui.SharingSpace` : a `SharingSpace` value

## 10.5 Help

`public class Help extends javax.swing.JFrame`

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          javax.swing.JFrame
            fungus.gui.Help
```
p. 144

### 10.5.1 Attributes

p. 151 `private fungus.gui.ModuleManager mm :`

`private javax.swing.tree.DefaultMutableTreeNode rootNode :`

`private javax.swing.tree.DefaultTreeModel treeModel :`

`private javax.swing.JTree jTree1 :`

`private javax.swing.JLabel help :`

`private javax.swing.JButton jButton1 :`

`private javax.swing.JScrollPane jScrollPane3 :`

`private javax.swing.JScrollPane jScrollPane1 :`

`private javax.swing.JPanel jPanel1 :`

`private javax.swing.JSplitPane jSplitPane2 :`

### 10.5.2 Constructors

**`public Help(ModuleManager mm)`**
Creates new form Preferences

**Parameters**

p. 151 `fungus.gui.ModuleManager mm :`

### 10.5.3 Methods

**`private void initComponents()`**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**`private void okAction(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**`private void exitForm(WindowEvent evt)`**
Exit the Application

**Parameters**

`java.awt.event.WindowEvent evt` :

**`private TreeModel getTreeModel()`**

Renvoie le tree model du JTree situe a gauche de paneau

**return** `javax.swing.tree.TreeModel` :
a `TreeModel` value

**`public void addTreeNodeText(String s)`**

Ajoute une ligne au JTree

**Parameters**

`java.lang.String s` : a `String` value

**`public void setModule(Module module)`**

Donne le scrollpane dans lequles on va faire nos preference

**Parameters**

p. 149 `fungus.gui.Module module` :

## 10.6 MainFrame

public class MainFrame extends javax.swing.JFrame

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          javax.swing.JFrame
            fungus.gui.MainFrame
```
p. 146

### 10.6.1 Attributes

p. 151
private fungus.gui.ModuleManager mm :

private javax.swing.JMenuItem aboutMenuItem :

private javax.swing.JMenuItem openMenuItem :

private javax.swing.JMenu fileMenu :

private javax.swing.JButton loadButton :

private javax.swing.JButton jButton3 :

private javax.swing.JSeparator jSeparator2 :

private javax.swing.JButton saveAllButton :

private javax.swing.JSeparator jSeparator1 :

private javax.swing.JButton prefButton :

private javax.swing.JMenuItem cutMenuItem :

private javax.swing.JMenuBar menuBar :

private javax.swing.JScrollPane jScrollPane1 :

private javax.swing.JMenuItem saveMenuItem :

private javax.swing.JMenu moduleMenu :

private javax.swing.JMenuItem copyMenuItem :

private javax.swing.JMenuItem deleteMenuItem :

private javax.swing.JMenuItem saveAllMenuItem :

private javax.swing.JMenuItem newMenuItem :

private javax.swing.JMenuItem exitMenuItem :

private javax.swing.JButton saveButton :

`private javax.swing.JMenuItem preferenceMenuItem :`

`private javax.swing.JMenuItem contentsMenuItem :`

`private javax.swing.JMenu editMenu :`

`private javax.swing.JLabel jLabel2 :`

`private javax.swing.JMenuItem pasteMenuItem :`

`private javax.swing.JToolBar jToolBar1 :`

`private javax.swing.JMenu helpMenu :`

### 10.6.2 Constructors

**`public MainFrame()`**
Creates new form Gui

### 10.6.3 Methods

**`private void initComponents()`**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**`private void loadState(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**`private void saveAllState(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**`private void saveState(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**`private void preferenceMenuItemActionPerformed(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**`private void aboutMenuItemActionPerformed(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**`private void exitMenuItemActionPerformed(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**private void exitForm(WindowEvent evt)**

Exit the Application

**Parameters**

`java.awt.event.WindowEvent evt` :

**public ModuleManager getModuleManager()**

p. 151 **return** `fungus.gui.ModuleManager` :

## 10.7 `Module`

`public interface class Module extends null`

`fungus.gui.Module`

interface pour les modules de l'interface utilisateur.

### 10.7.1 Methods

**`public void init(SharingSpace s)`**
Initialise le module avec tt

**Parameters**

p. 160 `fungus.gui.SharingSpace s` : Espace d'echange

**`public String getName()`**
le nom du module

**return** `java.lang.String` :
le nom du module

**`public JMenuItem getMenuItem()`**
l'item qui est inséré dans le menu

**return** `javax.swing.JMenuItem` :
a `JMenuItem` value

**`public ImageIcon getIcon()`**
l'icône affiché dans la barre d'outils.

**return** `javax.swing.ImageIcon` :
an `ImageIcon` value

**`public JPanel getPanel()`**
ce qui doit être affiche dans la fenêtre

**return** `javax.swing.JPanel` :
a `JPanel` value

**`public JPanel getPreferencePanel()`**
le panel de preférence.

**return** `javax.swing.JPanel` :
a `JPanel` value

**`public String getHelp()`**
Renvoie de l'aide. L'aide est une page HTML.

**return** `java.lang.String` :

**`public void getPreferences(Properties p)`**
Prend les préférence.

**Parameters**

`java.util.Properties p` : les préférences du système.

**public void setPreferences(Properties p)**

Fixe les preferences. On se debrouille pour le faire...

**Parameters**

`java.util.Properties p` : l'ensemble des propriétés du système graphique.

**public void saveState(Properties p)**

Sauvegarde l'état d'un module Appelé lors d'un 'save' du menu.

**Parameters**

`java.util.Properties p` : contientra l'état du module

**public void loadState(Properties p)**

Charge l'état du module

**Parameters**

`java.util.Properties p` : contient l'état du module

## 10.8 ModuleManager

`public class ModuleManager extends java.lang.Object`

`java.lang.Object`
p. 151 `fungus.gui.ModuleManager`

Manager de modules

### 10.8.1 Attributes

`private javax.swing.JMenu menu` :

`private javax.swing.JMenuItem contentsMenu` :

`private javax.swing.JToolBar toolBar` :

`private javax.swing.JScrollPane panel` :

`private javax.swing.JScrollPane prefPanel` :

`private javax.swing.JButton prefbutton` :

p. 160 `private fungus.gui.SharingSpace sspc` :

p. 149 `private fungus.gui.Module current` :

p. 156 `private fungus.gui.Preferences p` :

p. 144 `private fungus.gui.Help h` :

`private static java.util.HashMap modules` :

`private java.util.Properties prop` :

`public final java.lang.String preferencesFileName` :

### 10.8.2 Constructors

**`public ModuleManager(JMenu m, JMenuItem c, JToolBar tb, JScrollPane p, JButton b, SharingSpace s)`**

Creates new ModuleManager

**Parameters**

- `javax.swing.JMenu m` :
- `javax.swing.JMenuItem c` :
- `javax.swing.JToolBar tb` :
- `javax.swing.JScrollPane p` :
- `javax.swing.JButton b` :
- p. 160 `fungus.gui.SharingSpace s` :

### 10.8.3 Methods

**`public void addModule(Module m)`**

**Parameters**

p. 149 `fungus.gui.Module m` :

**`public static Module getModule(String name)`**

**Parameters**

`java.lang.String name` :

p. 149 **return** `fungus.gui.Module` :

**`public static Iterator getModulesIterator()`**

Donne tous les modules enregistrés.

**return** `java.util.Iterator` :
an `Iterator` value

**`public void setPreferences()`**

**`public Properties getPreferences()`**

**return** `java.util.Properties` :

**`public void saveCurrent(File f)`**

Sauve l'état du module courant.

**Parameters**

`java.io.File f` : Le fichier de sauvegarde.

**`public void saveAll(File f)`**

Sauvegarde l'état de tous les modules

**Parameters**

`java.io.File f` : a `File` value

**`public void loadAll(File f)`**

Charge l'état des module

**Parameters**

`java.io.File f` : a `File` value

## 10.9 ModuleManager.ButtonAL

```
class ModuleManager.ButtonAL extends java.lang.Object
  implements ActionListener
```

```
java.lang.Object
  fungus.gui.ModuleManager.ButtonAL
```
p. 153

Action listener des bouton des modules.

### 10.9.1 Attributes

p. 149 `private fungus.gui.Module module` :

`private boolean prefSet_p` :

### 10.9.2 Constructors

**`public ModuleManager.ButtonAL(ModuleManager this$0, Module m)`**

**Parameters**

p. 151 `fungus.gui.ModuleManager this$0` :

p. 149 `fungus.gui.Module m` :

### 10.9.3 Methods

**`public void actionPerformed(ActionEvent actionEvent)`**

**Parameters**

`java.awt.event.ActionEvent actionEvent` :

## 10.10 ModuleManager.ContAL

`private class ModuleManager.ContAL extends java.lang.Object`
`implements ActionListener`

`java.lang.Object`
p. 154 `fungus.gui.ModuleManager.ContAL`

Action listener du menu Halp.

### 10.10.1 Attributes

p. 149 `private fungus.gui.Module module` :

### 10.10.2 Constructors

**`public ModuleManager.ContAL(ModuleManager this$0)`**

**Parameters**

p. 151 `fungus.gui.ModuleManager this$0` :

### 10.10.3 Methods

**`public void actionPerformed(ActionEvent actionEvent)`**

**Parameters**

`java.awt.event.ActionEvent actionEvent` :

## 10.11 ModuleManager.PrefAL

private class ModuleManager.PrefAL extends java.lang.Object
implements ActionListener

java.lang.Object
p. 155 fungus.gui.ModuleManager.PrefAL

Action listener du bouton preference.

### 10.11.1 Attributes

p. 149 private fungus.gui.Module module :

### 10.11.2 Constructors

**public ModuleManager.PrefAL(ModuleManager this$0)**

**Parameters**

p. 151 fungus.gui.ModuleManager this$0 :

### 10.11.3 Methods

**public void actionPerformed(ActionEvent actionEvent)**

**Parameters**

java.awt.event.ActionEvent actionEvent :

### 10.12 Preferences

public class Preferences extends javax.swing.JFrame

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          javax.swing.JFrame
            fungus.gui.Preferences
```
p. 156

#### 10.12.1 Attributes

p. 151 private fungus.gui.ModuleManager mm :

private javax.swing.tree.DefaultMutableTreeNode rootNode :

private javax.swing.tree.DefaultTreeModel treeModel :

private javax.swing.JTree jTree1 :

private javax.swing.JButton jButton3 :

private javax.swing.JButton jButton2 :

private javax.swing.JButton jButton1 :

private javax.swing.JScrollPane jScrollPane3 :

private javax.swing.JScrollPane jScrollPane1 :

private javax.swing.JLabel jLabel1 :

private javax.swing.JPanel jPanel1 :

private javax.swing.JSplitPane jSplitPane2 :

#### 10.12.2 Constructors

**public Preferences(ModuleManager mm)**
Creates new form Preferences

**Parameters**

p. 151 fungus.gui.ModuleManager mm :

#### 10.12.3 Methods

**private void initComponents()**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**private void applyAction(ActionEvent evt)**

**Parameters**

java.awt.event.ActionEvent evt :

**`private void okAction(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt` :

**`private void exitForm(WindowEvent evt)`**

Exit the Application

**Parameters**

`java.awt.event.WindowEvent evt` :

**`private TreeModel getTreeModel()`**

Renvoie le tree model du JTree situe a gauche de paneau

**return** `javax.swing.tree.TreeModel` :
a `TreeModel` value

**`public void addTreeNodeText(String s)`**

Ajoute une ligne au JTree

**Parameters**

`java.lang.String s` : a `String` value

**`public JScrollPane getScrollPane()`**

Donne le scrollpane dans lequles on va faire nos preference

**return** `javax.swing.JScrollPane` :
a `javax.swing.JScrollPane` value

**`public void setModule(Module module)`**

**Parameters**

p. 149 `fungus.gui.Module module` :

## 10.13 SaveAllStateFileChooser

`public class SaveAllStateFileChooser extends javax.swing.JFileChooser`

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JFileChooser
          fungus.gui.SaveAllStateFileChooser
```
p. 158

File Chooser pour les SaveAll. On propose de choisir les modules à sauver.

### 10.13.1 Constructors

**`public SaveAllStateFileChooser()`**

Creates a new `SaveAllStateFileChooser` instance.

## 10.14 SaveAllStateFileChooser.Accessory

private class SaveAllStateFileChooser.Accessory extends javax.swing.JPanel

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          fungus.gui.SaveAllStateFileChooser.Accessory
```
p. 159

### 10.14.1 Constructors

**public SaveAllStateFileChooser.Accessory(SaveAllStateFileChooser this$0, SaveAllStateFileChooser fc)**

**Parameters**

p. 158 fungus.gui.SaveAllStateFileChooser this$0 :

p. 158 fungus.gui.SaveAllStateFileChooser fc :

## 10.15 SharingSpace

`public class SharingSpace extends java.util.HashMap`

```
java.lang.Object
  java.util.AbstractMap
    java.util.HashMap
      fungus.gui.SharingSpace
```
p. 160

Espace d'echange entre les modules.
C'est une simple extension de l'interface java.util.Map.

### 10.15.1 Constructors

**public SharingSpace()**

## 10.16 Splash

public class Splash extends javax.swing.JFrame

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          javax.swing.JFrame
            fungus.gui.Splash
```
p. 161

### 10.16.1 Attributes

private boolean threadgo :

private int value :

private javax.swing.JProgressBar jProgressBar1 :

private javax.swing.JLabel jLabel2 :

private javax.swing.JLabel jLabel1 :

private javax.swing.JPanel jPanel1 :

### 10.16.2 Constructors

**public Splash()**
Creates new form Splash

### 10.16.3 Methods

**private void initComponents()**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**private void exitForm(WindowEvent evt)**
Exit the Application

**Parameters**

java.awt.event.WindowEvent evt :

**public void setVisible(boolean flag)**

**Parameters**

boolean flag :

**public void dispose()**

**public void setMax(int n)**

**Parameters**

int n :

**public void incValue()**

# 11 fungus.gui.module

Les modules de l'interface utilisateur.

## 11.1 AgentsMonitorModule

```
public class AgentsMonitorModule extends java.lang.Object
   implements Module
```

```
java.lang.Object
  fungus.gui.module.AgentsMonitorModule
```
p. 162

Donne des informations sur les agents de la suimualtion en cours.

### 11.1.1 Attributes

p. 160 `private fungus.gui.SharingSpace sspc` :

p. 164 `private fungus.gui.module.AgentsMonitorPanel p` :

### 11.1.2 Constructors

**`public AgentsMonitorModule()`**

### 11.1.3 Methods

**`public void init(SharingSpace s)`**

**Parameters**

p. 160 `fungus.gui.SharingSpace s` :

**`public ImageIcon getIcon()`**

**return** `javax.swing.ImageIcon` :

**`public JMenuItem getMenuItem()`**

**return** `javax.swing.JMenuItem` :

**`public String getName()`**

**return** `java.lang.String` :

**`public JPanel getPanel()`**

**return** `javax.swing.JPanel` :

**`public JPanel getPreferencePanel()`**

**return** `javax.swing.JPanel` :

**`public String getHelp()`**

**return** `java.lang.String` :

**`public void getPreferences(Properties p)`**

**Parameters**

`java.util.Properties p` :

**`public void setPreferences(Properties p)`**

**Parameters**

`java.util.Properties p` :

**`public void saveState(Properties states)`**

**Parameters**

`java.util.Properties states` :

**`public void loadState(Properties states)`**

**Parameters**

`java.util.Properties states` :

**`public SharingSpace getSharingSpace()`**

p. 160 **return** `fungus.gui.SharingSpace` :

## 11.2 AgentsMonitorPanel

public class AgentsMonitorPanel extends javax.swing.JPanel

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          fungus.gui.module.AgentsMonitorPanel
```

p. 164

### 11.2.1 Attributes

p. 162 private fungus.gui.module.AgentsMonitorModule amm :

private javax.swing.JScrollPane jScrollPane16 :

private javax.swing.JPanel popertieList :

private javax.swing.JLabel agentInfo :

private javax.swing.JList properties :

private javax.swing.JScrollPane jScrollPane27 :

private javax.swing.JScrollPane jScrollPane26 :

private javax.swing.JList agents :

private javax.swing.JButton jButton1 :

private javax.swing.JButton jButton11 :

private javax.swing.JScrollPane jScrollPane1 :

private javax.swing.JPanel jPanel5 :

private javax.swing.JPanel agentList :

private javax.swing.JLabel jLabel16 :

private javax.swing.JPanel jPanel1 :

private javax.swing.JSplitPane jSplitPane2 :

private javax.swing.JSplitPane jSplitPane1 :

### 11.2.2 Constructors

**public AgentsMonitorPanel(AgentsMonitorModule m)**
Creates new form AgentsMonitorPanel

**Parameters**

p. 162 fungus.gui.module.AgentsMonitorModule m :

### 11.2.3 Methods

**private void initComponents()**

This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**private void agentlistSelection(ListSelectionEvent evt)**

**Parameters**

`javax.swing.event.ListSelectionEvent evt` :

**private void updateAgents(ActionEvent evt)**

**Parameters**

`java.awt.event.ActionEvent evt` :

**private void updateAgentsList()**

**private void setAgentInfo(String agent)**

**Parameters**

`java.lang.String agent` :

## 11.3 CommandModule

public class CommandModule extends java.lang.Object
implements Module

java.lang.Object
p. 166 fungus.gui.module.CommandModule

Panneau de commande des simulations.

### 11.3.1 Attributes

p. 160 private fungus.gui.SharingSpace sspc :

p. 63 private fungus.core.infrastructure.Simulation st :

private java.lang.String codeName :

p. 171 private fungus.gui.module.CommandPanel p :

p. 175 private fungus.gui.module.CommandPreferencePanel pp :

### 11.3.2 Constructors

**public CommandModule()**

### 11.3.3 Methods

**public void init(SharingSpace s)**

**Parameters**

p. 160 fungus.gui.SharingSpace s :

**public ImageIcon getIcon()**

**return** javax.swing.ImageIcon :

**public JMenuItem getMenuItem()**

**return** javax.swing.JMenuItem :

**public String getName()**

**return** java.lang.String :

**public JPanel getPanel()**

**return** javax.swing.JPanel :

**public JPanel getPreferencePanel()**

**return** javax.swing.JPanel :

**public String getHelp()**

**return** java.lang.String :

**public void load(String s)**

Charge une simulation

**Parameters**

`java.lang.String s` : le nom de la simulation.

**public void play()**

Démare la simulation

**public void stop()**

arrête la simulation

**public void getPreferences(Properties prop)**

**Parameters**

`java.util.Properties prop` :

**public void setPreferences(Properties prop)**

**Parameters**

`java.util.Properties prop` :

**public void saveState(Properties prop)**

**Parameters**

`java.util.Properties prop` :

**public void loadState(Properties prop)**

**Parameters**

`java.util.Properties prop` :

## 11.4 CommandModuleParamCollector

public class CommandModuleParamCollector extends javax.swing.JPanel
implements ParametersCollectorInterface

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          fungus.gui.module.CommandModuleParamCollector
```
p. 168

Le collecteur de paramètre de la gui.

### 11.4.1 Attributes

p. 171 public static fungus.gui.module.CommandPanel p :

p. 138 private fungus.core.utils.hash.ObjectHash t :

private int nbattr :

### 11.4.2 Constructors

**public CommandModuleParamCollector()**
Creates new form CommandModuleParamCollector

### 11.4.3 Methods

**private void initComponents()**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**private JTextField _addParam(String title)**

**Parameters**

java.lang.String title :

**return** javax.swing.JTextField :

**public void addParam(String title)**
Ajoute un champ paramètre à la fenêtre

**Parameters**

java.lang.String title : le nom du champ

**public void addParam(String title, String defaut)**
Ajoute un champ paramètre à la fenêtre

**Parameters**

java.lang.String title : le nom du champ

java.lang.String defaut :

**public void collect()**
Attend que l'on ai remplit les champs.

**public boolean getBoolean(String title)**
Renvoie la valeur de ce champ sous la forme d'un boolean

**Parameters**

`java.lang.String title` : le nom du champ

**return** `boolean` :
la valeur du champ (true, autre)

**public double getDouble(String title)**
Renvoie la valeur de ce champ sous la forme d'un double

**Parameters**

`java.lang.String title` : le nom du champ

**return** `double` :
la valeur du champ

**public float getFloat(String title)**
Renvoie la valeur de ce champ sous la forme d'un float

**Parameters**

`java.lang.String title` : le nom du champ

**return** `float` :
la valeur du champ

**public int getInt(String title)**
Renvoie la valeur de ce champ sous la forme d'un int

**Parameters**

`java.lang.String title` : le nom du champ

**return** `int` :
la valeur du champ

**public long getLong(String title)**
Renvoie la valeur de ce champ sous la forme d'un long

**Parameters**

`java.lang.String title` : le nom du champ

**return** `long` :
la valeur du champ

**public String getString(String title)**
Renvoie la valeur de ce champ sous la forme d'un String

**Parameters**

`java.lang.String title` : le nom du champ

**return** `java.lang.String` :
la valeur du champ

**`public void setTitle(String title)`**

Donne un tire à la fenetre.

**Parameters**

`java.lang.String title` : a `String` value

## 11.5 CommandPanel

public class CommandPanel extends javax.swing.JPanel

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          fungus.gui.module.CommandPanel
```
p. 171

### 11.5.1 Attributes

p. 166 private fungus.gui.module.CommandModule cm :

private javax.swing.JButton reset :

private javax.swing.JButton playb :

private javax.swing.JTextField remoteHost :

private javax.swing.JTabbedPane jTabbedPane1 :

private javax.swing.JTextField jTextField1 :

private javax.swing.JPanel running :

private javax.swing.JProgressBar jProgressBar1 :

private javax.swing.JPanel st :

private javax.swing.JPanel Command :

private javax.swing.JButton jButton31 :

private javax.swing.JButton jButton3 :

private javax.swing.JButton jButton2 :

private javax.swing.JTextArea outBuffer :

private javax.swing.JScrollPane jScrollPane2 :

private javax.swing.JLabel jLabel3 :

private javax.swing.JLabel jLabel2 :

private javax.swing.JPanel simparam :

private javax.swing.JLabel jLabel1 :

private javax.swing.JPanel jPanel5 :

private javax.swing.JButton stopb :

`private javax.swing.JTextField scriptName :`

`private javax.swing.JPanel jPanel4 :`

`private javax.swing.JPanel jPanel3 :`

`private javax.swing.JPanel jPanel2 :`

`private javax.swing.JPanel stc :`

`private javax.swing.JPanel jPanel1 :`

`private int maxLine :`
nombres de lignes max de la sortie.

### 11.5.2 Constructors

**`public CommandPanel(CommandModule c)`**
Creates new form CommandPanel

**Parameters**

p. 166 `fungus.gui.module.CommandModule c :`

### 11.5.3 Methods

**`private void initComponents()`**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**`private void scriptBrowse(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**`private void stopAction(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**`private void savaAction(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**`private void loadAction(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**`private void playAction(ActionEvent evt)`**

**Parameters**

`java.awt.event.ActionEvent evt :`

**public void setSimPrefPanel(JPanel p)**

**Parameters**

`javax.swing.JPanel p :`

**public void setPref(int p)**

**Parameters**

`int p :`

**public void saveState(Properties state, String c)**

**Parameters**

`java.util.Properties state :`

`java.lang.String c :`

**public void loadState(Properties state, String c)**

**Parameters**

`java.util.Properties state :`

`java.lang.String c :`

**public void setStartUpName(String name)**

**Parameters**

`java.lang.String name :`

**public void setScriptName(String name)**

**Parameters**

`java.lang.String name :`

**public void setRemoteHostName(String name)**

**Parameters**

`java.lang.String name :`

## 11.6 CommandPanel.StandardOutputMonitor

private class CommandPanel.StandardOutputMonitor extends java.lang.Thread

```
java.lang.Object
  java.lang.Thread
    fungus.gui.module.CommandPanel.StandardOutputMonitor
```

p. 174

Écoute la sortie standard et la renvoie dans la fenêtre Swing.

### 11.6.1 Attributes

java.io.ByteArrayOutputStream out :

private int sleep :

### 11.6.2 Constructors

**public CommandPanel.StandardOutputMonitor(CommandPanel this$0)**

**Parameters**

p. 171 fungus.gui.module.CommandPanel this$0 :

### 11.6.3 Methods

**public void run()**

### 11.7 CommandPreferencePanel

public class CommandPreferencePanel extends javax.swing.JPanel

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          fungus.gui.module.CommandPreferencePanel
```
p. 175

#### 11.7.1 Attributes

p. 166 private fungus.gui.module.CommandModule cm :

private int oldPref :

private javax.swing.JTextField jTextField1 :

private javax.swing.JCheckBox params :

private javax.swing.JCheckBox running :

private javax.swing.JCheckBox st :

private javax.swing.JLabel jLabel2 :

private javax.swing.JLabel jLabel1 :

private javax.swing.JCheckBox stc :

private javax.swing.JPanel jPanel1 :

#### 11.7.2 Constructors

**public CommandPreferencePanel(CommandModule c)**
Creates new form CommandPreferencePanel

**Parameters**

p. 166 fungus.gui.module.CommandModule c :

#### 11.7.3 Methods

**private void initComponents()**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**public int getPref()**

**return** int :

**public void setPref(int p)**

**Parameters**

int p :

### 11.8 TestModule

`public class TestModule extends java.lang.Object`
`implements Module`

`java.lang.Object`
p. 176 `fungus.gui.module.TestModule`

Module de test

#### 11.8.1 Attributes

p. 160 `private fungus.gui.SharingSpace sspc` :

#### 11.8.2 Constructors

**`public TestModule()`**
Creates new TestModule

#### 11.8.3 Methods

**`public void init(SharingSpace s)`**

**Parameters**

p. 160 `fungus.gui.SharingSpace s` :

**`public ImageIcon getIcon()`**

**return** `javax.swing.ImageIcon` :

**`public JMenuItem getMenuItem()`**

**return** `javax.swing.JMenuItem` :

**`public String getName()`**

**return** `java.lang.String` :

**`public JPanel getPanel()`**

**return** `javax.swing.JPanel` :

**`public JPanel getPreferencePanel()`**

**return** `javax.swing.JPanel` :

**`public String getHelp()`**

**return** `java.lang.String` :

**`public void getPreferences(Properties p)`**

**Parameters**

`java.util.Properties p` :

**`public void setPreferences(Properties prop)`**

**Parameters**

java.util.Properties prop :

**public void saveState(Properties states)**

**Parameters**

java.util.Properties states :

**public void loadState(Properties states)**

**Parameters**

java.util.Properties states :

## 11.9 TestPanel

`public class TestPanel extends javax.swing.JPanel`

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          fungus.gui.module.TestPanel
```

p. 178

### 11.9.1 Attributes

`private javax.swing.JLabel jLabel1 :`

### 11.9.2 Constructors

**`public TestPanel()`**
Creates new form InnerPanel

### 11.9.3 Methods

**`private void initComponents()`**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

## 11.10 TestPreferencePanel

`public class TestPreferencePanel extends javax.swing.JPanel`

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          fungus.gui.module.TestPreferencePanel
```
p. 179

### 11.10.1 Attributes

`private javax.swing.JLabel jLabel1` :

`private javax.swing.ButtonGroup buttonGroup1` :

### 11.10.2 Constructors

**`public TestPreferencePanel()`**
Creates new form TestPreferencePanel

### 11.10.3 Methods

**`private void initComponents()`**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

## 11.11 VisualisationModule

```
public class VisualisationModule extends java.lang.Object
   implements Module, VisualisationContainer, VisualisationFactory
```

java.lang.Object

p. 180 fungus.gui.module.VisualisationModule

Visualise la simulation en cours.

### 11.11.1 Attributes

p. 160 private fungus.gui.SharingSpace sspc :

p. 182 private fungus.gui.module.VisualisationPanel p :

### 11.11.2 Constructors

**public VisualisationModule()**

### 11.11.3 Methods

**public void init(SharingSpace s)**

**Parameters**

p. 160 fungus.gui.SharingSpace s :

**public ImageIcon getIcon()**

**return** javax.swing.ImageIcon :

**public JMenuItem getMenuItem()**

**return** javax.swing.JMenuItem :

**public String getName()**

**return** java.lang.String :

**public JPanel getPanel()**

**return** javax.swing.JPanel :

**public JPanel getPreferencePanel()**

**return** javax.swing.JPanel :

**public String getHelp()**

**return** java.lang.String :

**public void getPreferences(Properties p)**

**Parameters**

java.util.Properties p :

**public void setPreferences(Properties prop)**

**Parameters**

java.util.Properties prop :

**public void saveState(Properties states)**

**Parameters**

java.util.Properties states :

**public void loadState(Properties states)**

**Parameters**

java.util.Properties states :

**public SharingSpace getSharingSpace()**

p. 160 **return** fungus.gui.SharingSpace :

**public boolean addVisualisationComponent(Component c)**

**Parameters**

java.awt.Component c :

**return** boolean :

**public Component getNewVisualisation(Object type) throws FungusException**

**Parameters**

java.lang.Object type :

**return** java.awt.Component :

**Exception**

p. 22 fungus.core.FungusException :

## 11.12 VisualisationPanel

`public class VisualisationPanel extends javax.swing.JPanel`

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          fungus.gui.module.VisualisationPanel
```
p. 182

### 11.12.1 Attributes

`private javax.swing.JSplitPane jSplitPane1` :

`private javax.swing.JTabbedPane jTabbedPane1` :

`private javax.swing.JLabel jLabel2` :

`private javax.swing.JLabel jLabel1` :

### 11.12.2 Constructors

**`public VisualisationPanel()`**
Creates new form VisualisationPanel

### 11.12.3 Methods

**`private void initComponents()`**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**`public boolean addVisualisation(JComponent p)`**

**Parameters**

`javax.swing.JComponent p` :

**return** `boolean` :

# 12 fungus.gui.module.visualisation

## 12.1 IntPlotPanel

`public class IntPlotPanel extends javax.swing.JPanel`

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          fungus.gui.module.visualisation.IntPlotPanel
```
p. 183

Courbe à valeurs entieres.

### 12.1.1 Attributes

p. 123 `private fungus.core.utils.IntAccumulator ac` :

`private int hp` :

`private int min` :

`private float coef` :

### 12.1.2 Constructors

**`public IntPlotPanel(IntAccumulator a)`**

**Parameters**

p. 123 `fungus.core.utils.IntAccumulator a` :

### 12.1.3 Methods

**`private void initComponents()`**

**`private int clip(int x)`**

**Parameters**

`int x` :

**return** `int` :

**`public void paintComponent(Graphics graphics)`**

**Parameters**

`java.awt.Graphics graphics` :

**`public void save(File f, PlotFileChooser.FileProperty prop) throws IOException`**

Sauce un graphique

**Parameters**

`java.io.File f` : le fichier de sortie

p. 186 `fungus.gui.module.visualisation.PlotFileChooser.FileProperty prop` : les propriétés du fichier

**Exception**

`java.io.IOException` : if an error occurs

## 12.2 PlotFileChooser

public class PlotFileChooser extends javax.swing.JFileChooser

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JFileChooser
          fungus.gui.module.visualisation.PlotFileChooser
```

p. 184

FileChosser spécialisé dans la sauvegarde des graphiques.

### 12.2.1 Constructors

**public PlotFileChooser()**

### 12.2.2 Methods

**public PlotFileChooser.FileProperty getFileProperty()**
Donne les propriétés du fichier selectionné.

p. 186 **return** fungus.gui.module.visualisation.PlotFileChooser.FileProperty :

## 12.3 PlotFileChooser.Accessory

```
private class PlotFileChooser.Accessory extends javax.swing.JPanel
   implements PropertyChangeListener, PlotFileChooser.FileProperty
```

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          fungus.gui.module.visualisation.PlotFileChooser.Accessory
```
p. 185

Paneau de configuration des fichiers à sauvegarder.

### 12.3.1 Attributes

`private java.lang.String type :`

`private javax.swing.JCheckBox grid :`

`private javax.swing.JCheckBox absolute :`

`private javax.swing.JComboBox output :`

### 12.3.2 Constructors

**`public PlotFileChooser.Accessory(PlotFileChooser this$0, PlotFileChooser fc)`**

**Parameters**

p. 184 `fungus.gui.module.visualisation.PlotFileChooser this$0 :`

p. 184 `fungus.gui.module.visualisation.PlotFileChooser fc :`

### 12.3.3 Methods

**`public void propertyChange(PropertyChangeEvent e)`**

**Parameters**

`java.beans.PropertyChangeEvent e :`

**`public String getType()`**

**return** `java.lang.String :`

**`public String getTerminalType()`**

**return** `java.lang.String :`

**`public boolean getGrid()`**

**return** `boolean :`

**`public boolean getAbsolutePath()`**

**return** `boolean :`

## 12.4 `PlotFileChooser.FileProperty`

`public static interface class PlotFileChooser.FileProperty extends null`

`fungus.gui.module.visualisation.PlotFileChooser.FileProperty`

Propriété du fichier à sauver

### 12.4.1 Methods

**`public String getType()`**
Type du fichier. raw, gp, ...

**return** `java.lang.String` :

**`public String getTerminalType()`**
Type du terminal pour GnuPlot.

**return** `java.lang.String` :

**`public boolean getGrid()`**
Grille dans Gnuplot ?

**return** `boolean` :

**`public boolean getAbsolutePath()`**
Mettre le chemin du fichier de données en absolue dans Gnuplot.

**return** `boolean` :

### 12.5 PlotFileChooser.GnuPlotFilter

`private class PlotFileChooser.GnuPlotFilter extends javax.swing.filechooser.FileFilter`

```
java.lang.Object
  javax.swing.filechooser.FileFilter
```
p. 187
```
    fungus.gui.module.visualisation.PlotFileChooser.GnuPlotFilter
```

Filtre pour "Gnu Plot"

#### 12.5.1 Constructors

**`PlotFileChooser.GnuPlotFilter(PlotFileChooser this$0)`**

**Parameters**

p. 184 `fungus.gui.module.visualisation.PlotFileChooser this$0` :

#### 12.5.2 Methods

**`public boolean accept(File f)`**

**Parameters**

`java.io.File f` :

**return** `boolean` :

**`public String getDescription()`**

**return** `java.lang.String` :

## 12.6 PlotFileChooser.RawFilter

`private class PlotFileChooser.RawFilter extends javax.swing.filechooser.FileFilter`

```
java.lang.Object
  javax.swing.filechooser.FileFilter
    fungus.gui.module.visualisation.PlotFileChooser.RawFilter
```
p. 188

Filtre pour le "Raw Text"

### 12.6.1 Constructors

**`PlotFileChooser.RawFilter(PlotFileChooser this$0)`**

**Parameters**

p. 184 `fungus.gui.module.visualisation.PlotFileChooser this$0` :

### 12.6.2 Methods

**`public boolean accept(File f)`**

**Parameters**

`java.io.File f` :

**return** `boolean` :

**`public String getDescription()`**

**return** `java.lang.String` :

## 12.7 PlotPanel

public class PlotPanel extends javax.swing.JPanel
implements DataCollector

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
p. 189 fungus.gui.module.visualisation.PlotPanel

### 12.7.1 Attributes

private long d :

private javax.swing.JToolBar jToolBar1 :

private javax.swing.JToggleButton jToggleButton1 :

private javax.swing.JButton saveAs :

private javax.swing.JButton jButton3 :

private javax.swing.JButton jButton2 :

private javax.swing.JButton jButton1 :

private javax.swing.JSeparator jSeparator1 :

private javax.swing.JScrollPane scroll :

p. 191 private fungus.gui.module.visualisation.Rule hr :

p. 191 private fungus.gui.module.visualisation.Rule vr :

p. 123 private fungus.core.utils.IntAccumulator ac :

p. 183 private fungus.gui.module.visualisation.IntPlotPanel panel :

### 12.7.2 Constructors

**public PlotPanel()**
Creates new form PlotPanel

### 12.7.3 Methods

**private void initComponents()**
This method is called from within the constructor to initialize the form. WARNING : Do NOT modify this code. The content of this method is always regenerated by the Form Editor.

**private void saveAsActionPerformed(ActionEvent evt)**

**Parameters**

java.awt.event.ActionEvent evt :

**public void addData(int x)**

Ajoute une donnée à la courbe

**Parameters**

`int x :`

**public void sendData(Object o)**

**Parameters**

`java.lang.Object o :`

**public void setAgentID(AgentID id)**

**Parameters**

p. 28 `fungus.core.agent.AgentID id :`

**public void start()**

**public void stop()**

## 12.8 Rule

`public class Rule extends javax.swing.JComponent`

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        fungus.gui.module.visualisation.Rule
```

p. 191

Une reglure

### 12.8.1 Attributes

`public static final int HORIZONTAL` :

`public static final int VERTICAL` :

`public static int SIZE` :

`public int orientation` :

`private java.awt.Font font` :

`private float start` :

`private float end` :

`private int parentSize` :

`private double log_10` :

### 12.8.2 Constructors

**`public Rule(int o)`**

**Parameters**

`int o` :

### 12.8.3 Methods

**`public void setPreferredHeight(int ph, int min, int max)`**

**Parameters**

`int ph` :

`int min` :

`int max` :

**`public void setPreferredWidth(int pw)`**

**Parameters**

`int pw` :

**`public void paintComponent(Graphics g)`**

**Parameters**

`java.awt.Graphics g` :

**`private double log10(double x)`**

**Parameters**

`double x` :

**return** `double` :

## 12.9 Space2D

```
public class Space2D extends javax.swing.JPanel
   implements DataCollector, Runnable
```

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          fungus.gui.module.visualisation.Space2D
```
p. 193

Visualise les espaces 2D

### 12.9.1 Attributes

p. 253 `private fungus.visualisation.Collectable ag :`

`private java.lang.Thread t :`

p. 53 `private fungus.core.infrastructure.Infrastructure infra :`

`private java.awt.image.MemoryImageSource mi :`

`private java.awt.Image img :`

`private int [] array :`

p. 249 `fungus.space.TwoDimensionLayer layer :`

`private boolean go :`

### 12.9.2 Constructors

**`public Space2D(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i :`

### 12.9.3 Methods

**`public void sendData(Object o)`**

**Parameters**

`java.lang.Object o :`

**`public void setAgentID(AgentID id)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID id :`

**`public void start()`**

**`public void stop()`**

**`public void run()`**

**public void paintComponent(Graphics g)**

**Parameters**

`java.awt.Graphics g` :

# 13 fungus.heatbugs

Simulation des *HeatBugs*.
Voir nullSwarmnull.
null nullnullValvassori Moïsenullnull null null Last modified : Wed Apr 24 11 :34 :00 CEST 2002 null

## 13.1 Bug

`public class Bug extends fungus.core.agent.Agent`

```
java.lang.Object
  fungus.core.agent.Agent
    fungus.heatbugs.Bug
```

p. 26
p. 195

### 13.1.1 Constructors

**public Bug()**
Creates a new Bug instance.

### 13.1.2 Methods

**public void init(ObjectHash arg)**

**Parameters**

p. 138 `fungus.core.utils.hash.ObjectHash arg` :

**public void act()**

## 13.2 HeatBugSpace

`public class HeatBugSpace extends fungus.space.TwoDimensionLayeredSapce`
`implements Collectable`

```
java.lang.Object
p. 26   fungus.core.agent.Agent
p. 30     fungus.core.agent.AgentsGroup
p. 246      fungus.space.Space
p. 243        fungus.space.LayeredSpace
p. 250          fungus.space.TwoDimensionLayeredSapce
p. 196            fungus.heatbugs.HeatBugSpace
```

Espace des HeatBug. Espace 2D avec une structure diffusive.

### 13.2.1 Constructors

**public HeatBugSpace()**
Creates a new `HeatBugSpace` instance.

### 13.2.2 Methods

**public void init(ObjectHash arg)**

**Parameters**

p. 138 `fungus.core.utils.hash.ObjectHash arg` :

**public void act()**

**public Object getData()**

**return** `java.lang.Object` :

**public void sendDataCollector(DataCollector dc)**

**Parameters**

p. 254 `fungus.visualisation.DataCollector dc` :

**public void run()**
Permet de couper le thread de l'agent. Le service scheduler prend la main.

# 14 `fungus.mycelium`

Langage de script pour créer des simulations sans JAVA.

## 14.1 `AgentMaker`

```
public class AgentMaker extends java.lang.Object
   implements AgentInstanceMaker
```

```
java.lang.Object
  fungus.mycelium.AgentMaker
```
p. 197

Fabraique des instances d'`Agent` avec mycelium.

### 14.1.1 Attributes

p. 205 `private fungus.mycelium.MyceliumService ms` :

### 14.1.2 Constructors

**`public AgentMaker(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

### 14.1.3 Methods

**`public Agent newInstance(String name) throws Exception`**

**Parameters**

`java.lang.String name` :

p. 26 **return** `fungus.core.agent.Agent` :

**Exception**

`java.lang.Exception` :

## 14.2 Mycelium

```
public class Mycelium extends fungus.core.infrastructure.SimulationAdapter
   implements Simulation
```

```
java.lang.Object
```
p. 64 `fungus.core.infrastructure.SimulationAdapter`
p. 198 `fungus.mycelium.Mycelium`

Classe de démarage pour les simulations utilisants Mycelium.

### 14.2.1 Attributes

p. 205 `private fungus.mycelium.MyceliumService ms` :

p. 217 `private fungus.mycelium.xml.Node mycNode` :

p. 125 `private fungus.core.utils.ParametersCollector p` :

### 14.2.2 Constructors

**`public Mycelium(Infrastructure infra)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure infra` :

### 14.2.3 Methods

**`public void declareParameters(Infrastructure infra)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure infra` :

**`public void getParameters(Infrastructure infra)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure infra` :

**`public void buildAgent(Infrastructure i)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

## 14.3 MyceliumAgent

```
public class MyceliumAgent extends fungus.core.agent.NotifyAgent
   implements MessageListener
```

```
        java.lang.Object
p. 26     fungus.core.agent.Agent
p. 24       fungus.core.agent.AccessibleAgent
p. 34         fungus.core.agent.NotifyAgent
p. 199          fungus.mycelium.MyceliumAgent
```

Agent de base de mycelium.
Il est capable d'interpreter les sequences Scheme

### 14.3.1 Attributes

p. 205 `protected fungus.mycelium.MyceliumService my` :

`protected java.lang.String act` :

`protected gnu.mapping.Environment toplevel` :

### 14.3.2 Constructors

**`public MyceliumAgent()`**

### 14.3.3 Methods

**`public void init(ObjectHash p)`**

**Parameters**

p. 138 `fungus.core.utils.hash.ObjectHash p` :

**`public void act()`**

**`public void messageRecevied(MessageEvent m)`**

**Parameters**

p. 47 `fungus.core.canal.MessageEvent m` :

## 14.4 MyceliumException

```
public class MyceliumException extends fungus.core.FungusException
```

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      fungus.core.FungusException
        fungus.mycelium.MyceliumException
```

p. 22
p. 200

Throws if an error occurs in Mycelium Process.

### 14.4.1 Constructors

**public MyceliumException()**

**public MyceliumException(String e)**

**Parameters**

java.lang.String e :

## 14.5 MyceliumProcessingException

`public class MyceliumProcessingException extends fungus.mycelium.MyceliumException`

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      fungus.core.FungusException
        fungus.mycelium.MyceliumException
          fungus.mycelium.MyceliumProcessingException
```

p. 22
p. 200
p. 201

Throws if an error occurs in Mycelium Process.

### 14.5.1 Constructors

**`public MyceliumProcessingException()`**

**`public MyceliumProcessingException(String e)`**

**Parameters**

`java.lang.String e` :

## 14.6 MyceliumRealService

`public class MyceliumRealService extends fungus.core.service.AbstractService implements MyceliumService`

```
java.lang.Object
```
p. 77 `fungus.core.service.AbstractService`
p. 202 `fungus.mycelium.MyceliumRealService`

Implémentation du service Mycelium.

### 14.6.1 Attributes

p. 229 `private fungus.mycelium.xml.processing.ProcessStrategy ps :`

p. 211 `private fungus.mycelium.procedures.SchemeSession scm :`

`private gnu.mapping.Environment _env :`

p. 138 `private fungus.core.utils.hash.ObjectHash vars :`

p. 138 `private fungus.core.utils.hash.ObjectHash behs :`

`public static final java.lang.String classDefinition :`
Envoie la definition d'une classe Mycelium.

### 14.6.2 Constructors

**`public MyceliumRealService()`**

### 14.6.3 Methods

**`public void init()`**

**`public Node parse(URL url) throws MyceliumException`**

**Parameters**

`java.net.URL url :`

p. 217 **return** `fungus.mycelium.xml.Node :`

**Exception**

p. 200 `fungus.mycelium.MyceliumException :`

**`public void process(Node root) throws MyceliumProcessingException`**

**Parameters**

p. 217 `fungus.mycelium.xml.Node root :`

**Exception**

p. 201 `fungus.mycelium.MyceliumProcessingException :`

**`public Object evalExpression(String expr) throws MyceliumException`**

**Parameters**

`java.lang.String expr :`

**return** `java.lang.Object` :

**Exception**

p. 200 `fungus.mycelium.MyceliumException` :

**`public Object evalExpression(Object expr) throws MyceliumException`**

**Parameters**

`java.lang.Object expr` :

**return** `java.lang.Object` :

**Exception**

p. 200 `fungus.mycelium.MyceliumException` :

**`public Object evalExpression(String expr, Environment env) throws MyceliumException`**

**Parameters**

`java.lang.String expr` :

`gnu.mapping.Environment env` :

**return** `java.lang.Object` :

**Exception**

p. 200 `fungus.mycelium.MyceliumException` :

**`public Object evalExpression(Object expr, Environment env) throws MyceliumException`**

**Parameters**

`java.lang.Object expr` :

`gnu.mapping.Environment env` :

**return** `java.lang.Object` :

**Exception**

p. 200 `fungus.mycelium.MyceliumException` :

**`public Environment newEnvironment() throws MyceliumException`**

**return** `gnu.mapping.Environment` :

**Exception**

p. 200 `fungus.mycelium.MyceliumException` :

**`public static NetMessageListener getNetMessageListener(Infrastructure i)`**

donne une instance de `NetMessageListener`

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` : an `Infrastructure` value

p. 59 **return** `fungus.core.infrastructure.NetMessageListener` :

## 14.7 MyceliumRealService.MessageListener

`private static class MyceliumRealService.MessageListener extends java.lang.Object implements NetMessageListener`

```
java.lang.Object
  fungus.mycelium.MyceliumRealService.MessageListener
```
p. 204

repond au message de mycelium envoye sur le reseau

### 14.7.1 Attributes

p. 53 `private fungus.core.infrastructure.Infrastructure infra` :

p. 205 `private fungus.mycelium.MyceliumService ms` :

### 14.7.2 Constructors

**MyceliumRealService.MessageListener(Infrastructure i)**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

### 14.7.3 Methods

**private void startMycelium()**

**public boolean performMessage(String type, ObjectInputStream in, NetConnection out)**

**Parameters**

- `java.lang.String type` :
- `java.io.ObjectInputStream in` :
- p. 56 `fungus.core.infrastructure.NetConnection out` :

**return** `boolean` :

### 14.8 MyceliumService

`public interface class MyceliumService extends null`
`implements Service`

`fungus.mycelium.MyceliumService`

**Service :**

]ressource

mycelium ]role

Gere les agents "mycelium".

**Attention** Cette interface va bouger dans un futur proche.

#### 14.8.1 Attributes

`public static final java.lang.String ressource` :

#### 14.8.2 Methods

**`public Node parse(URL url) throws MyceliumException`**
parse une URL.

**Parameters**

`java.net.URL url` : l'url a parser

p. 217 **return** `fungus.mycelium.xml.Node` : la racine de l'arbre parser

**Exception**

p. 200 `fungus.mycelium.MyceliumException` : if an error occurs

**`public void process(Node root) throws MyceliumException`**
Execute l'arbre XML.

**Parameters**

p. 217 `fungus.mycelium.xml.Node root` : la racine de l'arbre.

**Exception**

p. 200 `fungus.mycelium.MyceliumException` : if an error occurs

**`public Object evalExpression(String expr) throws MyceliumException`**
Evalue une expression Scheme

**Parameters**

`java.lang.String expr` : L'expression à évaluer

**return** `java.lang.Object` :

**Exception**

p. 200 `fungus.mycelium.MyceliumException` : if an error occurs

**`public Object evalExpression(Object expr) throws MyceliumException`**
Evalue une expression Scheme

**Parameters**

`java.lang.Object expr` : L'expression à évaluer

**return** `java.lang.Object` :

**Exception**

p. 200 `fungus.mycelium.MyceliumException` : if an error occurs

**`public Object evalExpression(String expr, Environment env) throws MyceliumException`**

Evalue une expression Scheme

**Parameters**

`java.lang.String expr` : L'expression à évaluer

`gnu.mapping.Environment env` : l'environment d'évaluation

**return** `java.lang.Object` :

**Exception**

p. 200 `fungus.mycelium.MyceliumException` : if an error occurs

**`public Object evalExpression(Object expr, Environment env) throws MyceliumException`**

Evalue une expression Scheme

**Parameters**

`java.lang.Object expr` : L'expression à évaluer

`gnu.mapping.Environment env` : l'environment d'évaluation

**return** `java.lang.Object` :

**Exception**

p. 200 `fungus.mycelium.MyceliumException` : if an error occurs

**`public Environment newEnvironment() throws MyceliumException`**

Fabrique un nouvel environment.
Cet environment sera directement lié au *Top Level*.

**return** `gnu.mapping.Environment` :

**Exception**

p. 200 `fungus.mycelium.MyceliumException` :

## 14.9 Test

`public class Test extends java.lang.Object`

```
java.lang.Object
  fungus.mycelium.Test
```
p. 207

Cice est un test de kawa.

### 14.9.1 Constructors

**`public Test()`**

### 14.9.2 Methods

**`public static void main(String [] args) throws Exception`**

**Parameters**

`java.lang.String [] args` :

**Exception**

`java.lang.Exception` :

**`public static void test1()`**

ce test definit *b* dans un premier environnement. puis crée deux env qui hérite du premier ou l'on met deux valeurs différentes de *a*. Lors de l'evaluation de `(+ a b)` dans chacun des environnements ont constate des valeur différentes.
On affiche aussi toutes les binding contenu dans l'environnement.

**`public static void test2()`**

Dans ce test, on perse juste une expresion que l'on evalue

**`public static void test3()`**

marche pas

**`public static void test4() throws Exception`**

**Exception**

`java.lang.Exception` :

## 14.10 XmlTest

`public class XmlTest extends java.lang.Object`

```
java.lang.Object
  fungus.mycelium.XmlTest
```

p. 208

Cice est un test de kawa.

### 14.10.1 Constructors

**public XmlTest()**

### 14.10.2 Methods

**public static void main(String [] args) throws Exception**

**Parameters**

`java.lang.String [] args` :

**Exception**

`java.lang.Exception` :

**public static void test() throws Exception**

**Exception**

`java.lang.Exception` :

# 15 fungus.mycelium.procedures

Nouvelles procedures nullSchemenull.
null nullnullValvassori Moïsenullnull null null Last modified : Fri Apr 12 12 :15 :10 CEST 2002 null

## 15.1 Get

`public class Get extends gnu.mapping.Procedure2`

```
java.lang.Object
  gnu.mapping.Procedure
    gnu.mapping.Procedure2
      fungus.mycelium.procedures.Get
```
p. 209

raccourci pour les acces aux tables de hachages.

### 15.1.1 Constructors

**`public Get()`**

### 15.1.2 Methods

**`public Object apply2(Object map, Object key)`**

**Parameters**

`java.lang.Object map` :

`java.lang.Object key` :

**return** `java.lang.Object` :

## 15.2 Id

public class Id extends kawa.standard.thisRef

```
java.lang.Object
  kawa.lang.Syntax
    kawa.standard.thisRef
      fungus.mycelium.procedures.Id
```
p. 210

Fait un "getID" sur un agent. alias de `(invoke (this) 'getID)` ou encore `this.getID`.

### 15.2.1 Constructors

**public Id()**

### 15.2.2 Methods

**public Expression rewriteForm(Pair form, Translator tr)**

**Parameters**

- gnu.lists.Pair form :
- kawa.lang.Translator tr :

**return** gnu.expr.Expression :

## 15.3 SchemeSession

`public class SchemeSession extends kawa.standard.Scheme`

```
java.lang.Object
  gnu.expr.Interpreter
    gnu.kawa.lispexpr.LispInterpreter
      kawa.standard.Scheme
        fungus.mycelium.procedures.SchemeSession
```
p. 211

Extension du Scheme de Kawa. Rajout de procedures.

### 15.3.1 Constructors

**public SchemeSession()**
Creates a new `SchemeSession` instance.

### 15.3.2 Methods

**public void initScheme()**

## 15.4 Send

public class Send extends kawa.lang.Syntax

```
java.lang.Object
  kawa.lang.Syntax
    fungus.mycelium.procedures.Send
```
p. 212

### 15.4.1 Constructors

**public Send()**

### 15.4.2 Methods

**public Expression rewriteForm(Pair form, Translator tr)**

**Parameters**

gnu.lists.Pair form :

kawa.lang.Translator tr :

**return** gnu.expr.Expression :

## 15.5 SendTo

`public class SendTo extends gnu.mapping.Procedure3`

```
java.lang.Object
  gnu.mapping.Procedure
    gnu.mapping.Procedure3
      fungus.mycelium.procedures.SendTo
```
p. 213

Envoie un message dans un canal.

### 15.5.1 Constructors

**`public SendTo()`**

### 15.5.2 Methods

**`public Object apply3(Object canal, Object to, Object message)`**

**Parameters**

- `java.lang.Object canal` :
- `java.lang.Object to` :
- `java.lang.Object message` :

**return** `java.lang.Object` :

## 15.6 Set

`public class Set extends gnu.mapping.Procedure3`

```
java.lang.Object
  gnu.mapping.Procedure
    gnu.mapping.Procedure3
      fungus.mycelium.procedures.Set
```
p. 214

raccourci pour les acces aux tables de hachages.

### 15.6.1 Constructors

**`public Set()`**

### 15.6.2 Methods

**`public Object apply3(Object map, Object key, Object value)`**

**Parameters**

- `java.lang.Object map` :
- `java.lang.Object key` :
- `java.lang.Object value` :

**return** `java.lang.Object` :

# 16 fungus.mycelium.xml

Classes pour parser l'XML de mycelium.

## 16.1 Handler

`public class Handler extends gnu.xml.ParsedXMLHandler`

```
java.lang.Object
  gnu.xml.ParsedXMLHandler
    fungus.mycelium.xml.Handler
```
p. 215

XML Handler pour Mycelium.
Construit un arbre de `Node`.

### 16.1.1 Attributes

p. 217 `private fungus.mycelium.xml.Node currentNode` :

p. 217 `private fungus.mycelium.xml.Node rootNode` :

`private java.lang.String currentAttr` :

`private java.lang.String currentChar` :

`private boolean alreadyHaveAttr` :

### 16.1.2 Constructors

**`public Handler()`**

### 16.1.3 Methods

**`public Node getRootNode()`**

p. 217 **return** `fungus.mycelium.xml.Node` :

**`public void emitCharacters(char [] text, int start, int length)`**
Process raw text.

**Parameters**

- `char [] text` :
- `int start` :
- `int length` :

**`public void emitCharacterReference(int value, char [] name, int start, int length)`**

**Parameters**

- `int value` :
- `char [] name` :
- `int start` :
- `int length` :

**`public void emitBeginElement(char [] name, int start, int length)`**
Process a start tag, with the given element name.

**Parameters**

`char [] name` :

`int start` :

`int length` :

**`public void emitBeginAttribute(char [] name, int start, int length)`**

Process an attribute, with the given attribute name. The attribute value is given using emitCharacters. The value is terminated by either another emitBeginAttribute or an emitEndAttributes.

**Parameters**

`char [] name` :

`int start` :

`int length` :

**`public void emitEndAttributes()`**

Process the end of a start tag. There are no more attributes.

**`public void emitEndElement(char [] name, int start, int length)`**

Process an end tag. An abbrevated tag (such as 'null') has a name==null.

**Parameters**

`char [] name` :

`int start` :

`int length` :

## 16.2 Node

`public class Node extends java.lang.Object`

`java.lang.Object`
p. 217 `fungus.mycelium.xml.Node`

Un Noeud XML.

### 16.2.1 Attributes

`private java.lang.String tag :`

p. 138 `private fungus.core.utils.hash.ObjectHash attrs :`

p. 217 `private fungus.mycelium.xml.Node parent :`

`private java.util.ArrayList childs :`

### 16.2.2 Constructors

**`public Node(String tag)`**
Ce noeud n'a pas de pere.

**Parameters**

`java.lang.String tag` : le nom du tag

**`public Node(String tag, Node parent)`**
Creates a new `Node` instance.

**Parameters**

`java.lang.String tag` : le nom du tag

p. 217 `fungus.mycelium.xml.Node parent` : le noeud parent

### 16.2.3 Methods

**`public void addAttribute(String name, String value)`**
Ajoute un attribut à ce noeud

**Parameters**

`java.lang.String name` : le nom de l'attribut

`java.lang.String value` : la valeur de l'attribut

**`public String getAttributeValue(String attr)`**
Renvoie la valeur d'une attribut

**Parameters**

`java.lang.String attr` : l'attribut que l'on interoge, `null` si l'attribut n'existe pas

**return** `java.lang.String` :

**`public void addChildNode(Node child)`**
Ajoute un fils à ce noeud

**Parameters**

p. 217 `fungus.mycelium.xml.Node child` : a `Node` value

**`public Iterator getChildrens()`**

Donne tous les fils de ce noeud

**return** `java.util.Iterator` :
un iterateur sur les fils du noeud.

**`public Iterator getChildsTag(String tag)`**

Donne la liste des fils ayant un tag donné

**Parameters**

`java.lang.String tag` : le tag que l'on souhaite avoir

**return** `java.util.Iterator` :
un iterateur sur les fils.

**`public Iterator getChildsTag(Set tags)`**

Donne la liste des fils ayant un des tags donnés

**Parameters**

`java.util.Set tags` :

**return** `java.util.Iterator` :
un iterateur sur les fils.

**`public Node getParentNode()`**

donne le pere de ce noeud

p. 217 **return** `fungus.mycelium.xml.Node` :

**`public String getTagName()`**

Donne le tag de ce noeud

**return** `java.lang.String` :

**`public String toString()`**

**return** `java.lang.String` :

**`public void prettyPrint()`**

Pretty Print de l'arbre

**`public void prettyPrint(int style)`**

Pretty Print de l'arbre

**Parameters**

`int style` : le style du pretty print.

**`private void __prettyPrint(int lvl, int style)`**

**Parameters**

`int lvl` :

`int style` :

**`public void dispose()`**

On détruit cet arbre.

# 17 fungus.mycelium.xml.processing

Méthodes de traitement des noeuds XML.
null nullnullValvassori Moïsenullnull null null Last modified : Tue Mar 5 22 :58 :21 CET 2002 null

## 17.1 AbstractNodeProcessor

```
public abstract class AbstractNodeProcessor extends java.lang.Object
   implements NodeProcessor
```

```
java.lang.Object
  fungus.mycelium.xml.processing.AbstractNodeProcessor
```
p. 220

Strategy Absraite.
stocke que le ProcessStrategy createur.

### 17.1.1 Attributes

p. 229 `protected fungus.mycelium.xml.processing.ProcessStrategy ps :`

p. 53 `protected fungus.core.infrastructure.Infrastructure infra :`

### 17.1.2 Constructors

**public AbstractNodeProcessor()**

### 17.1.3 Methods

**public void setStrategy(ProcessStrategy ps)**

**Parameters**

p. 229 `fungus.mycelium.xml.processing.ProcessStrategy ps :`

**public void setInfrastructure(Infrastructure i)**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i :`

## 17.2 Agent

`public class Agent extends fungus.mycelium.xml.processing.AbstractNodeProcessor`

```
java.lang.Object
p. 220  fungus.mycelium.xml.processing.AbstractNodeProcessor
p. 221    fungus.mycelium.xml.processing.Agent
```

Execute les tag *Agent*.

### 17.2.1 Attributes

p. 83 `private fungus.core.service.CreateAgentService ca` :

`private static boolean firstUse` :
Si `false` alors on a encore jamais créer d'agent mycelium.

`private java.util.HashSet s` :

### 17.2.2 Constructors

**`public Agent()`**

### 17.2.3 Methods

**`public void processNode(Node node) throws MyceliumProcessingException`**

**Parameters**

p. 217 `fungus.mycelium.xml.Node node` :

**Exception**

p. 201 `fungus.mycelium.MyceliumProcessingException` :

## 17.3 AgentClass

`public class AgentClass extends fungus.mycelium.xml.processing.AbstractNodeProcessor`

```
java.lang.Object
  fungus.mycelium.xml.processing.AbstractNodeProcessor
    fungus.mycelium.xml.processing.AgentClass
```
p. 220
p. 222

Execute les tags *AgentClass*.

### 17.3.1 Attributes

`private java.util.HashSet s` :

### 17.3.2 Constructors

**`public AgentClass()`**

### 17.3.3 Methods

**`public void processNode(Node node) throws MyceliumProcessingException`**

**Parameters**

p. 217 `fungus.mycelium.xml.Node node` :

**Exception**

p. 201 `fungus.mycelium.MyceliumProcessingException` :

## 17.4 Behavior

public class Behavior extends fungus.mycelium.xml.processing.AbstractNodeProcessor

java.lang.Object
p. 220 fungus.mycelium.xml.processing.AbstractNodeProcessor
p. 223 fungus.mycelium.xml.processing.Behavior

Execute les tags *Bahavior*.

### 17.4.1 Constructors

**public Behavior()**

### 17.4.2 Methods

**public void processNode(Node node) throws MyceliumProcessingException**

**Parameters**

p. 217 fungus.mycelium.xml.Node node :

**Exception**

p. 201 fungus.mycelium.MyceliumProcessingException :

## 17.5 Definition

`public class Definition extends fungus.mycelium.xml.processing.AbstractNodeProcessor`

```
        java.lang.Object
p. 220    fungus.mycelium.xml.processing.AbstractNodeProcessor
p. 224      fungus.mycelium.xml.processing.Definition
```

Execute les tag *Definition*.

### 17.5.1 Attributes

`private java.util.HashSet s` :

### 17.5.2 Constructors

**`public Definition()`**

### 17.5.3 Methods

**`public void processNode(Node node) throws MyceliumProcessingException`**

**Parameters**

p. 217 `fungus.mycelium.xml.Node node` :

**Exception**

p. 201 `fungus.mycelium.MyceliumProcessingException` :

## 17.6 Initialisation

public class Initialisation extends fungus.mycelium.xml.processing.AbstractNodeProcessor

java.lang.Object
p. 220 fungus.mycelium.xml.processing.AbstractNodeProcessor
p. 225 fungus.mycelium.xml.processing.Initialisation

Execute les tags *initialisation*.

### 17.6.1 Constructors

**public Initialisation()**

### 17.6.2 Methods

**public void processNode(Node node) throws MyceliumProcessingException**

**Parameters**

p. 217 fungus.mycelium.xml.Node node :

**Exception**

p. 201 fungus.mycelium.MyceliumProcessingException :

## 17.7 Mycelium

public class Mycelium extends fungus.mycelium.xml.processing.AbstractNodeProcessor

java.lang.Object
p. 220 fungus.mycelium.xml.processing.AbstractNodeProcessor
p. 226 fungus.mycelium.xml.processing.Mycelium

Execute les tag *Mycelium*.

### 17.7.1 Constructors

**public Mycelium()**

### 17.7.2 Methods

**public void processNode(Node node) throws MyceliumProcessingException**

**Parameters**

p. 217 fungus.mycelium.xml.Node node :

**Exception**

p. 201 fungus.mycelium.MyceliumProcessingException :

## 17.8 `NodeProcessor`

`public interface class NodeProcessor extends null`

`fungus.mycelium.xml.processing.NodeProcessor`

La strategy d'execution employe face a un noeud particulier.

### 17.8.1 Methods

**`public void processNode(Node node) throws MyceliumProcessingException`**
Le traitement du noeud

**Parameters**

p. 217 `fungus.mycelium.xml.Node node` : le noeud a traiter

**Exception**

p. 201 `fungus.mycelium.MyceliumProcessingException` : if an error occurs

**`public void setStrategy(ProcessStrategy ps)`**
Fixe la strategy qui a crée cette instance. N'est utilise que lors de la creation.

**Parameters**

p. 229 `fungus.mycelium.xml.processing.ProcessStrategy ps` :

**`public void setInfrastructure(Infrastructure i)`**
Fixe l'infrastructure. N'est utilise que lors de la creation.

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure i` :

## 17.9 Parameter

public class Parameter extends fungus.mycelium.xml.processing.AbstractNodeProcessor

java.lang.Object
p. 220 fungus.mycelium.xml.processing.AbstractNodeProcessor
p. 228 fungus.mycelium.xml.processing.Parameter

Execute les tags *Parameter*. La valeur du paramètre est évalué par Mycelium.

### 17.9.1 Constructors

**public Parameter()**

### 17.9.2 Methods

**public void processNode(Node node) throws MyceliumProcessingException**

**Parameters**

p. 217 fungus.mycelium.xml.Node node :

**Exception**

p. 201 fungus.mycelium.MyceliumProcessingException :

## 17.10 ProcessStrategy

`public class ProcessStrategy extends java.lang.Object`

`java.lang.Object`
p. 229 `fungus.mycelium.xml.processing.ProcessStrategy`

Strategie d'execution d'un noeud XML.
Cette classe sait executer tous les noeuds de l'arbre XML.
Les classes qui traitent chacun des noeuds est définit dans le fichier `/fungus/fungus.prop`.

### 17.10.1 Attributes

p. 138 `private fungus.core.utils.hash.ObjectHash tags` :

p. 53 `private fungus.core.infrastructure.Infrastructure infra` :

### 17.10.2 Constructors

**`public ProcessStrategy(Infrastructure infra)`**

**Parameters**

p. 53 `fungus.core.infrastructure.Infrastructure infra` :

### 17.10.3 Methods

**`public void process(Node node) throws MyceliumProcessingException`**
Execute une noeud.

**Parameters**

p. 217 `fungus.mycelium.xml.Node node` : a `Node` value

**Exception**

p. 201 `fungus.mycelium.MyceliumProcessingException` : if an error occurs

## 17.11 Repeat

public class Repeat extends fungus.mycelium.xml.processing.AbstractNodeProcessor

java.lang.Object
p. 220 fungus.mycelium.xml.processing.AbstractNodeProcessor
p. 230 fungus.mycelium.xml.processing.Repeat

Execute les tags *Repeat*.

### 17.11.1 Attributes

private java.util.HashSet s :

### 17.11.2 Constructors

**public Repeat()**

### 17.11.3 Methods

**public void processNode(Node node) throws MyceliumProcessingException**

**Parameters**

p. 217 fungus.mycelium.xml.Node node :

**Exception**

p. 201 fungus.mycelium.MyceliumProcessingException :

## 17.12 Scheme

public class Scheme extends fungus.mycelium.xml.processing.AbstractNodeProcessor

java.lang.Object
p. 220 fungus.mycelium.xml.processing.AbstractNodeProcessor
p. 231 fungus.mycelium.xml.processing.Scheme

Execute les tag *Scheme*.

### 17.12.1 Constructors

**public Scheme()**

### 17.12.2 Methods

**public void processNode(Node node) throws MyceliumProcessingException**

**Parameters**

p. 217 fungus.mycelium.xml.Node node :

**Exception**

p. 201 fungus.mycelium.MyceliumProcessingException :

## 17.13 Simulation

public class Simulation extends fungus.mycelium.xml.processing.AbstractNodeProcessor

java.lang.Object
p. 220 fungus.mycelium.xml.processing.AbstractNodeProcessor
p. 232 fungus.mycelium.xml.processing.Simulation

Execute les tag *Simulation*.

### 17.13.1 Attributes

private java.util.HashSet s :

### 17.13.2 Constructors

**public Simulation()**

### 17.13.3 Methods

**public void processNode(Node node) throws MyceliumProcessingException**

**Parameters**

p. 217 fungus.mycelium.xml.Node node :

**Exception**

p. 201 fungus.mycelium.MyceliumProcessingException :

## 17.14 Variable

public class Variable extends fungus.mycelium.xml.processing.AbstractNodeProcessor

java.lang.Object
p. 220 fungus.mycelium.xml.processing.AbstractNodeProcessor
p. 233 fungus.mycelium.xml.processing.Variable

Execute les tags *variable*.

### 17.14.1 Constructors

**public Variable()**

### 17.14.2 Methods

**public void processNode(Node node) throws MyceliumProcessingException**

**Parameters**

p. 217 fungus.mycelium.xml.Node node :

**Exception**

p. 201 fungus.mycelium.MyceliumProcessingException :

## 17.15 Visualize

public class Visualize extends fungus.mycelium.xml.processing.AbstractNodeProcessor

java.lang.Object
p. 220 fungus.mycelium.xml.processing.AbstractNodeProcessor
p. 234 fungus.mycelium.xml.processing.Visualize

Execute les tags *visualize*.

### 17.15.1 Constructors

**public Visualize()**

### 17.15.2 Methods

**public void processNode(Node node) throws MyceliumProcessingException**

**Parameters**

p. 217 fungus.mycelium.xml.Node node :

**Exception**

p. 201 fungus.mycelium.MyceliumProcessingException :

# 18 fungus.plot

## 18.1 Collector

public class Collector extends java.lang.Object

java.lang.Object
p. 235 fungus.plot.Collector

### 18.1.1 Attributes

java.util.Vector data :

### 18.1.2 Constructors

**public Collector()**

### 18.1.3 Methods

**public void setData(Serializable f)**

**Parameters**

java.io.Serializable f :

**public Serializable getData(int i)**

**Parameters**

int i :

**return** java.io.Serializable :

**public int getSize()**

**return** int :

## 18.2 PlotOutput

`public class PlotOutput extends java.lang.Object`

`java.lang.Object`
p. 236 `fungus.plot.PlotOutput`

Affiche un ■plot■ sur le terminal

### 18.2.1 Attributes

`private float min` :

`private float max` :

`private int w` :

`private int h` :

`private java.awt.Graphics g` :

`private java.awt.Panel p` :

### 18.2.2 Constructors

**`public PlotOutput()`**

### 18.2.3 Methods

**`public void plot(Collector col)`**
Dessine le graph

**Parameters**

p. 235 `fungus.plot.Collector col` : a `Collector` value

## 18.3 PlotOutput.MenuAction

```
private class PlotOutput.MenuAction extends java.lang.Object
   implements ActionListener
```

```
java.lang.Object
```
p. 237
```
  fungus.plot.PlotOutput.MenuAction
```

### 18.3.1 Constructors

**PlotOutput.MenuAction(PlotOutput this$0)**

**Parameters**

p. 236
fungus.plot.PlotOutput this$0 :

### 18.3.2 Methods

**public void actionPerformed(ActionEvent e)**

**Parameters**

java.awt.event.ActionEvent e :

## 18.4 Plotter

`public class Plotter extends fungus.core.agent.NotifyAgent`

`java.lang.Object`<br>
p. 26 `fungus.core.agent.Agent`<br>
p. 24 `fungus.core.agent.AccessibleAgent`<br>
p. 34 `fungus.core.agent.NotifyAgent`<br>
p. 238 `fungus.plot.Plotter`

Classe qui affiche des données à l'écran

### 18.4.1 Attributes

p. 235 `fungus.plot.Collector list` :

p. 236 `fungus.plot.PlotOutput out` :

p. 28 `fungus.core.agent.AgentID watch` :

### 18.4.2 Constructors

**`public Plotter()`**

### 18.4.3 Methods

**`public void init(ObjectHash p)`**

**Parameters**

p. 138 `fungus.core.utils.hash.ObjectHash p` :

**`public void act()`**

**`public synchronized void messageRecevied(MessageEvent m)`**

**Parameters**

p. 47 `fungus.core.canal.MessageEvent m` :

## 18.5 Ruler

public class Ruler extends java.awt.Canvas

```
java.lang.Object
  java.awt.Component
    java.awt.Canvas
      fungus.plot.Ruler
```

p. 239

### 18.5.1 Attributes

public static final int horizontal :

public static final int vertical :

private static int size :

private int sens :

### 18.5.2 Constructors

**public Ruler(int s)**

**Parameters**

int s :

### 18.5.3 Methods

**public Dimension getPreferredSize()**

**return** java.awt.Dimension :

**public void paint(Graphics g)**

**Parameters**

java.awt.Graphics g :

# 19 fungus.space

Représentation des espaces. Un espace doit pouvoir renvoyer son voisinage ie une boule. Il s'aide pour cela de sa *structure*.

null nullnullValvassori Moïsenullnull null null null

## 19.1 Diffusive2DLayer

`public class Diffusive2DLayer extends fungus.space.TwoDimensionLayer`

```
java.lang.Object
  fungus.space.TwoDimensionLayer
    fungus.space.Diffusive2DLayer
```

p. 249
p. 240

Layer 2D qui diffuse...

### 19.1.1 Attributes

`protected int [][] back` :

### 19.1.2 Constructors

**`public Diffusive2DLayer(int w, int h)`**

Creates a new `Diffusive2DLayer` instance.

**Parameters**

- `int w` :
- `int h` :

### 19.1.3 Methods

**`public void diffuse()`**

## 19.2 EuclidianStructure

`public class EuclidianStructure extends java.lang.Object`
`implements Structure`

`java.lang.Object`
p. 241 `fungus.space.EuclidianStructure`

Structure euclidienne
l'espace est lineaire et ne boucle pas sur lui même

### 19.2.1 Attributes

`int size` :

### 19.2.2 Constructors

**`public EuclidianStructure(int size)`**

**Parameters**

`int size` :

### 19.2.3 Methods

**`public double distance(double p1, double p2)`**

**Parameters**

`double p1` :

`double p2` :

**return** `double` :

**`public double canonicalPosition(double p)`**

**Parameters**

`double p` :

**return** `double` :

## 19.3 Layer

```
public interface class Layer extends null
```

```
fungus.space.Layer
```

Définit un layer.

## 19.4 LayeredSpace

`public abstract class LayeredSpace extends fungus.space.Space`

```
java.lang.Object
p. 26     fungus.core.agent.Agent
p. 30       fungus.core.agent.AgentsGroup
p. 246        fungus.space.Space
p. 243          fungus.space.LayeredSpace
```

Espace Abstrait avec des 'layers'

### 19.4.1 Attributes

p. 242 `protected fungus.space.Layer [] layers` :

### 19.4.2 Constructors

**`public LayeredSpace()`**

### 19.4.3 Methods

**`public synchronized void messageRecevied(MessageEvent m)`**

**Parameters**

p. 47 `fungus.core.canal.MessageEvent m` :

**`protected void sendLayerObject()`**

**`protected abstract void getLayersObject(AgentID source, Serializable param)`**

Demande un object d'un layer

**Parameters**

p. 28 `fungus.core.agent.AgentID source` : an `AgentID` value

`java.io.Serializable param` : a `Serializable` value

**`protected abstract void setLayersObject(AgentID source, Serializable param)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID source` :

`java.io.Serializable param` :

## 19.5 LayeredSpaceMessage

`public class LayeredSpaceMessage extends fungus.space.SpaceMessage`

`java.lang.Object`
p. 31 `fungus.core.agent.AgentsGroupMessage`
p. 247 `fungus.space.SpaceMessage`
p. 244 `fungus.space.LayeredSpaceMessage`

Messages des espaces avec 'layers'

### 19.5.1 Attributes

`public static final int type` :

`public static final byte getLayersObject` :
Prend un objet d'un layer à un endroit donné

`public static final byte sendLayersObject` :
Envoie une object du layer.

`public static final byte setLayersObject` :
place un objet sur un layer à une position donnée.

### 19.5.2 Constructors

**`public LayeredSpaceMessage(AgentID source, byte cmd, Serializable param)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID source` :
`byte cmd` :
`java.io.Serializable param` :

### 19.5.3 Methods

**`public int getType()`**

**return** `int` :

**`public String toString()`**

**return** `java.lang.String` :

## 19.6 OneDim

`public class OneDim extends fungus.space.Space`

```
java.lang.Object
p. 26    fungus.core.agent.Agent
p. 30      fungus.core.agent.AgentsGroup
p. 246       fungus.space.Space
p. 245         fungus.space.OneDim
```

Espace discret à une dimension ou une cellule contient qu'un seul agent
paramètre initial :
- size : Taille initiale de l'espace.
- struct :

### 19.6.1 Attributes

`private int size` :

p. 28 `private fungus.core.agent.AgentID [] space` : Les agents qui seront placés dans cet espace

p. 136 `private fungus.core.utils.hash.AgentIDHash position` : AgentID -> position dans l'espace.

p. 248 `private fungus.space.Structure topo` :

`private int __pos` :

### 19.6.2 Constructors

**`public OneDim()`**

### 19.6.3 Methods

**`public void init(ObjectHash p)`**

**Parameters**

p. 138 `fungus.core.utils.hash.ObjectHash p` :

**`public void addAgent(AgentID id)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID id` :

**`public void addAgent(AgentID id, int pos)`**
Ajoute un agent à une position donnée

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : l'Agent à ajouter
`int pos` : la position dans l'espace.

**`public void askforNeighbours(AgentID source, Serializable param)`**
Renvoie le voisinage d'un agent en fonction de param

**Parameters**

p. 28 `fungus.core.agent.AgentID source` : Agent au centre du voisinage
`java.io.Serializable param` : parametre du voisinage (par exple la taille)
pas utilise ici

## 19.7 Space

public abstract class Space extends fungus.core.agent.AgentsGroup

java.lang.Object
p. 26 fungus.core.agent.Agent
p. 30 fungus.core.agent.AgentsGroup
p. 246 fungus.space.Space

Espace Abstrait

### 19.7.1 Constructors

**public Space()**

### 19.7.2 Methods

**public synchronized void messageRecevied(MessageEvent m)**

**Parameters**

p. 47 fungus.core.canal.MessageEvent m :

**public abstract void askforNeighbours(AgentID source, Serializable param)**
Renvoie le voisinage d'un agent en fonction de param

**Parameters**

p. 28 fungus.core.agent.AgentID source : Agent au centre du voisinage

java.io.Serializable param : parametre du voisinage (par exple la taille)

## 19.8 SpaceMessage

`public class SpaceMessage extends fungus.core.agent.AgentsGroupMessage`

```
java.lang.Object
  fungus.core.agent.AgentsGroupMessage
    fungus.space.SpaceMessage
```
p. 31
p. 247

### 19.8.1 Attributes

`public static final byte askforNeighbours :`
Demande la liste des voisins

`public static final byte giveNeighbours :`
Donne la liste des voisins

`public static final int type :`

### 19.8.2 Constructors

**public SpaceMessage(AgentID source, byte cmd, Serializable param)**

**Parameters**

p. 28
- `fungus.core.agent.AgentID source` :
- `byte cmd` :
- `java.io.Serializable param` :

### 19.8.3 Methods

**public int getType()**

**return** `int` :

**public String toString()**

**return** `java.lang.String` :

## 19.9 `Structure`

`public interface class Structure extends null`

`fungus.space.Structure`

Décrit la structure d'un espace.
Cette structure n'a un effet que sur un seul axe.

### 19.9.1 Methods

**`public double distance(double p1, double p2)`**
Donne la distance entre deux points

**Parameters**

- `double p1` : premier point
- `double p2` : second point

**return** `double` :
la distance

**`public double canonicalPosition(double p)`**
Donnes la position canoniques d'un point
Par exemple : Pour un espace circulaire, la position modulo le cyle

**Parameters**

- `double p` : le point

**return** `double` :
la position canonique

## 19.10 TwoDimensionLayer

public class TwoDimensionLayer extends java.lang.Object
implements Layer

java.lang.Object
p. 249 fungus.space.TwoDimensionLayer

Un layer à deux dimension.

### 19.10.1 Attributes

protected int [][] layer :

protected int width :

protected int height :

### 19.10.2 Constructors

**public TwoDimensionLayer(int w, int h)**
Creates a new TwoDimensionLayer instance.

**Parameters**

int w :

int h :

### 19.10.3 Methods

**public void setInt(int x, int y, int val)**
Place un point dans le layer

**Parameters**

int x : an int value

int y : an int value

int val : an int value

**public int getInt(int x, int y)**
Renvoie la valeur d'un point du layer.

**Parameters**

int x : an int value

int y : an int value

**return** int :
an int value

**public int getWidth()**
Donne la largeur de ce layer.

**return** int :

**public int getHeight()**
Donne la hauteur de ce layer.

**return** int :

## 19.11 TwoDimensionLayeredSapce

`public class TwoDimensionLayeredSapce extends fungus.space.LayeredSpace`

`java.lang.Object`<br>
p. 26 `fungus.core.agent.Agent`<br>
p. 30 `fungus.core.agent.AgentsGroup`<br>
p. 246 `fungus.space.Space`<br>
p. 243 `fungus.space.LayeredSpace`<br>
p. 250 `fungus.space.TwoDimensionLayeredSapce`

Un espace 2D fini avec des layers.
Paramètres initiaux :
- *size* : taille de la grille.
- *struct_x* : structure de sur l'axe x.
- *struct_y* : structure de sur l'axe y.

Pour l'instant, la seule structure reconnue est "*euclidian*".

### 19.11.1 Attributes

`protected int size` :

p. 28 `private fungus.core.agent.AgentID [][] space` :

p. 248 `private fungus.space.Structure sx` :

p. 248 `private fungus.space.Structure sy` :

`private java.util.Random r` :

`private int layercount` :

### 19.11.2 Constructors

**`public TwoDimensionLayeredSapce()`**

### 19.11.3 Methods

**`public void init(ObjectHash p)`**

**Parameters**

p. 138 `fungus.core.utils.hash.ObjectHash p` :

**`public void addAgent(AgentID id)`**
Ajoute un agent à une position choisie au hasard.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` :

**`public void addAgent(AgentID id, int x, int y)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID id` :<br>
`int x` :<br>
`int y` :

**`protected void getLayersObject(AgentID source, Serializable param)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID source` :

`java.io.Serializable param` :

**`protected void setLayersObject(AgentID source, Serializable param)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID source` :

`java.io.Serializable param` :

**`public void askforNeighbours(AgentID source, Serializable param)`**

**Parameters**

p. 28 `fungus.core.agent.AgentID source` :

`java.io.Serializable param` :

**`protected void setLayerNumber(int n)`**

Indique le nombre de layers de cet espace et initialise la table.
Si on avait déjà définit des layers, ils sont perdus...

**Parameters**

`int n` : le nombre de layer souhaité ;

**`protected int newLayer(Layer l)`**

place un nouveau layer

**Parameters**

p. 242 `fungus.space.Layer l` : le Layer à placer.

**return** `int` :

# 20 fungus.visualisation

Outils pour visualiser une partie d'une simulation. Les agents visualisables implémetnent l'interface `Collectable`. On déclare la visualisation lors de la creation de la simulation et le tour est joué.

null nullnullValvassori Moïsenullnull null null Last modified : Thu Mar 28 12 :20 :51 CET 2002 null

## 20.1 AWTVisualisationContainer

```
public class AWTVisualisationContainer extends java.awt.Frame
   implements VisualisationContainer
```

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          fungus.visualisation.AWTVisualisationContainer
```
p. 252

Container par default

### 20.1.1 Constructors

**`public AWTVisualisationContainer()`**
Creates a new `AWTVisualisationContainer` instance.

### 20.1.2 Methods

**`public boolean addVisualisationComponent(Component c)`**

**Parameters**

`java.awt.Component c` :

**return** `boolean` :

## 20.2 `Collectable`

`public interface class Collectable extends null`

`fungus.visualisation.Collectable`

If implemented an agent is visualible.

### 20.2.1 Methods

**`public Object getData()`**
Recoit une donnée.

**return** `java.lang.Object` :

**`public void sendDataCollector(DataCollector dc)`**
Envoie le Collecteur de donnée lie à cette visualisation.
Le Collectable peut donc envoyer directement ses données à la visualisation.

**Parameters**

p. 254 `fungus.visualisation.DataCollector dc` : a `DataCollector` value

## 20.3 `DataCollector`

`public interface class DataCollector extends null`

`fungus.visualisation.DataCollector`

Collecte des données. Ces données pourront ensuite être visualisées.

### 20.3.1 Methods

**`public void sendData(Object o)`**
Envoie une donnée à la visualisation.

**Parameters**

`java.lang.Object o` :

**`public void setAgentID(AgentID id)`**
Donne l'id de l'agent à traire.

**Parameters**

p. 28 `fungus.core.agent.AgentID id` : an `AgentID` value

**`public void start()`**
Démare la visualisation.

**`public void stop()`**
Stoppe la visualisation.

## 20.4 NullVisualisation

public class NullVisualisation extends java.awt.Label
implements DataCollector

java.lang.Object
java.awt.Component
java.awt.Label
p. 255 fungus.visualisation.NullVisualisation

Ne visualize rien.

### 20.4.1 Constructors

**public NullVisualisation()**
Creates a new NullVisualisation instance.

### 20.4.2 Methods

**public void sendData(Object o)**

**Parameters**

java.lang.Object o :

**public void setAgentID(AgentID id)**

**Parameters**

p. 28 fungus.core.agent.AgentID id :

**public void start()**

**public void stop()**

## 20.5 NullVisualisationFactory

```
public class NullVisualisationFactory extends java.lang.Object
   implements VisualisationFactory
```

```
java.lang.Object
  fungus.visualisation.NullVisualisationFactory
```

p. 256

Renvoie pas de visualisation

### 20.5.1 Constructors

**public NullVisualisationFactory()**

Creates a new NullVisualisationFactory instance.

### 20.5.2 Methods

**public Component getNewVisualisation(Object type) throws FungusException**

**Parameters**

java.lang.Object type :

**return** java.awt.Component :

**Exception**

p. 22

fungus.core.FungusException :

## 20.6 Visualisation

```
public class Visualisation extends fungus.core.service.AbstractService
   implements VisualisationService
```

```
java.lang.Object
  fungus.core.service.AbstractService
    fungus.visualisation.Visualisation
```

p. 77
p. 257

Implémentation du service de visu. !\ ne gère pas le rezo.

### 20.6.1 Constructors

**public Visualisation()**

### 20.6.2 Methods

**public void addVisualisation(AgentID ag, Object type)**

**Parameters**

p. 28
- fungus.core.agent.AgentID ag :
- java.lang.Object type :

**public void addVisualisation(AgentID ag, Object type, Object arg)**

**Parameters**

p. 28
- fungus.core.agent.AgentID ag :
- java.lang.Object type :
- java.lang.Object arg :

## 20.7 VisualisationContainer

`public interface class VisualisationContainer extends null`

`fungus.visualisation.VisualisationContainer`

Receptable pour une visualisation.

### 20.7.1 Methods

**`public boolean addVisualisationComponent(Component c)`**
Rajoute une visualisation dans le container

**Parameters**

`java.awt.Component c` :

**return** `boolean` :

## 20.8 VisualisationFactory

`public interface class VisualisationFactory extends null`

`fungus.visualisation.VisualisationFactory`

Renvoie un nouveau composant (panel) pour y faire une visualisation. le composant sera fonction du type.

### 20.8.1 Methods

**`public Component getNewVisualisation(Object type) throws FungusException`**
Renvoie le composant

**Parameters**

`java.lang.Object type` : critère de selection du composant.

**return** `java.awt.Component` :

**Exception**

p. 22 `fungus.core.FungusException` : si aucune visualisation du type nulltypenull n'est disponible.

## 20.9 VisualisationManager

`public class VisualisationManager extends java.lang.Object`

`java.lang.Object`
p. 260 `fungus.visualisation.VisualisationManager`

Gère les différents types de container.
Un container de visualisation est un frame ou un panel pouvant recevoir une visualisation.

### 20.9.1 Attributes

p. 258 `private static fungus.visualisation.VisualisationContainer v` :

p. 259 `private static fungus.visualisation.VisualisationFactory vs` :

### 20.9.2 Constructors

**`public VisualisationManager()`**

### 20.9.3 Methods

**`public static VisualisationContainer getVisualisationContainer()`**
Donne le container courant

p. 258 **return** `fungus.visualisation.VisualisationContainer` :

**`public static void setVisualisationContainer(VisualisationContainer c)`**
Fixe un nouveau container

**Parameters**

p. 258 `fungus.visualisation.VisualisationContainer c` :

**`public static void setVisualisationFactory(VisualisationFactory s)`**
Fixe le nouveau selectionner

**Parameters**

p. 259 `fungus.visualisation.VisualisationFactory s` :

**`public static Component getVisualisationComponent(Object type)`**
Renvoie le composant ou va s'effectue la visualisation.

**Parameters**

`java.lang.Object type` : an `Object` value

**return** `java.awt.Component` :
a `Component` value

# 21 gov.sandia.postscript

## 21.1 PSGraphics

`public class PSGraphics extends gov.sandia.postscript.PSGrBase`

```
java.lang.Object
  java.awt.Graphics
p. 262   gov.sandia.postscript.PSGrBase
p. 261     gov.sandia.postscript.PSGraphics
```

PSGr1 is a Graphics subclass for Java 1.1 that images to PostScript. (C) 1996 E.J. Friedman-Hill and Sandia National Labs

### 21.1.1 Constructors

**`public PSGraphics()`**
Constructs a new PSGr1 Object. Unlike regular Graphics objects, PSGr contexts can be created directly.

**`public PSGraphics(Writer o)`**
Constructs a new PSGr1 Object. Unlike regular Graphics objects, PSGr contexts can be created directly.

**Parameters**

`java.io.Writer o` : Output stream for PostScript output

**`public PSGraphics(Writer o, boolean emitProlog)`**
Constructs a new PSGr1 Object. Unlike regular Graphics objects, PSGr contexts can be created directly.

**Parameters**

`java.io.Writer o` : Output stream for PostScript output

`boolean emitProlog` :

## 21.2 PSGrBase

`public abstract class PSGrBase extends java.awt.Graphics`

```
java.lang.Object
  java.awt.Graphics
    gov.sandia.postscript.PSGrBase
```

p. 262

PSGr is a Graphics subclass that images to PostScript. (C) 1996 E.J. Friedman-Hill and Sandia National Labs

### 21.2.1 Attributes

`public static final boolean DEBUG` :

`protected static final int PAGEHEIGHT` :

`protected static final int PAGEWIDTH` :

`protected static final int XOFFSET` :

`protected static final int YOFFSET` :

`protected static final char [] hd` :
hexadecimal digits

`protected static final int charsPerRow` :
number of chars in a full row of pixel data

`protected java.io.PrintWriter os` :
Output stream where postscript goes

`protected java.awt.Color clr` :
The current color

`protected java.awt.Color backClr` :
The background color of the current widget. It's up to the client software to set this correctly !

`protected java.awt.Font font` :
The current font

`protected java.awt.Rectangle clippingRect` :

### 21.2.2 Constructors

**`public PSGrBase()`**
Constructs a new PSGr Object that write to stdout. Unlike regular Graphics objects, PSGr contexts can be created directly.

**`public PSGrBase(Writer w)`**
Constructs a new PSGr Object. Unlike regular Graphics objects, PSGr contexts can be created directly.

**Parameters**

`java.io.Writer w` :

**`public PSGrBase(Writer w, boolean emitProlog)`**
Constructs a new PSGr Object. Unlike regular Graphics objects, PSGr contexts can be created directly.

**Parameters**

`java.io.Writer w` :

`boolean emitProlog` :

### 21.2.3 Methods

**`public Graphics create()`**

Creates a new PSGr Object that is a copy of the original PSGr Object. Not implemented ; throws RuntimeException

**return** `java.awt.Graphics` :

**`public void setOutput(Writer w)`**

Change the Writer this context's output goes to.

**Parameters**

`java.io.Writer w` :

**`public void translate(int x, int y)`**

Translates the specified parameters into the origin of the graphics context. All subsequent operations on this graphics context will be relative to this origin.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

**`protected void scale(double x, double y)`**

Scales the graphics context. All subsequent operations on this graphics context will be affected.

**Parameters**

`double x` :

`double y` :

**`protected void lineto(int x, int y)`**

**Parameters**

`int x` :

`int y` :

**`protected void moveto(int x, int y)`**

**Parameters**

`int x` :

`int y` :

**`public Color getColor()`**

Gets the current color.

**return** `java.awt.Color` :

**`public void setBackground(Color c)`**

Gets the current color.

**Parameters**

`java.awt.Color c` :

**`public void setColor(Color c)`**

Sets the current color to the specified color. All subsequent graphics operations will use this specified color.

**Parameters**

`java.awt.Color c` : the color to be set

**`public void setPaintMode()`**

Sets the default paint mode to overwrite the destination with the current color. PostScript has only paint mode.

**`public void setXORMode(Color c1)`**

Sets the paint mode to alternate between the current color and the new specified color. PostScript does not support XOR mode.

**Parameters**

`java.awt.Color c1` : the second color

**`public Font getFont()`**

Gets the current font.

**return** `java.awt.Font` :

**`public void setFont(Font f)`**

Sets the font for all subsequent text-drawing operations.

**Parameters**

`java.awt.Font f` :

**`public FontMetrics getFontMetrics()`**

Gets the current font metrics.

**return** `java.awt.FontMetrics` :

**`public FontMetrics getFontMetrics(Font f)`**

Gets the current font metrics for the specified font.

**Parameters**

`java.awt.Font f` : the specified font

**return** `java.awt.FontMetrics` :

**`public Rectangle getClipRect()`**

Returns the bounding rectangle of the current clipping area.

**return** `java.awt.Rectangle` :

**`public Shape getClip()`**

Returns the bounding rectangle of the current clipping area.

**return** `java.awt.Shape` :

**public void setClip(Shape s)**

Sets the clipping region using the shape's bounding rectangle.

**Parameters**

`java.awt.Shape s` : A shape to set the clipping rectangle to

**public Rectangle getClipBounds()**

Returns the bounding rectangle of the current clipping area.

**return** `java.awt.Rectangle` :
the bounding rectangle

**public void setClip(int x, int y, int width, int height)**

Clips to a rectangle. The resulting clipping area is the intersection of the current clipping area and the specified rectangle. Graphic operations have no effect outside of the clipping area.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

**public void clipRect(int x, int y, int width, int height)**

**Parameters**

`int x` :

`int y` :

`int width` :

`int height` :

**public void copyArea(int x, int y, int width, int height, int dx, int dy)**

Copies an area of the screen.

**Parameters**

`int x` : the x-coordinate of the source

`int y` : the y-coordinate of the source

`int width` : the width

`int height` : the height

`int dx` : the horizontal distance

`int dy` : the vertical distance Note : copyArea not supported by PostScript

**public void drawLine(int x1, int y1, int x2, int y2)**

Draws a line between the coordinates (x1,y1) and (x2,y2). The line is drawn below and to the left of the logical coordinates.

**Parameters**

`int x1` : the first point's x coordinate

`int y1` : the first point's y coordinate

`int x2` : the second point's x coordinate

`int y2` : the second point's y coordinate

**protected void doRect(int x, int y, int width, int height, boolean fill)**

**Parameters**

`int x` :

`int y` :

`int width` :

`int height` :

`boolean fill` :

**public void fillRect(int x, int y, int width, int height)**

Fills the specified rectangle with the current color.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

**public void drawRect(int x, int y, int width, int height)**

Draws the outline of the specified rectangle using the current color. Use drawRect(x, y, width-1, height-1) to draw the outline inside the specified rectangle.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

**public void clearRect(int x, int y, int width, int height)**

Clears the specified rectangle by filling it with the current background color of the current drawing surface. Which drawing surface it selects depends on how the graphics context was created.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

**protected void doRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight, boolean fill)**

**Parameters**

`int x` :

`int y` :

`int width` :

`int height` :

`int arcWidth` :

`int arcHeight` :

`boolean fill` :

**protected void stroke(boolean fill)**

**Parameters**

`boolean fill` :

**protected void arcTo(int x1, int y1, int x2, int y2, int dim)**

**Parameters**

`int x1` :

`int y1` :

`int x2` :

`int y2` :

`int dim` :

**public void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)**
Draws an outlined rounded corner rectangle using the current color.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

`int arcWidth` : the diameter of the arc

`int arcHeight` : the radius of the arc

**public void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)**
Draws a rounded rectangle filled in with the current color.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

`int arcWidth` : the diameter of the arc

`int arcHeight` : the radius of the arc

**public void draw3DRect(int x, int y, int width, int height, boolean raised)**
Draws a highlighted 3-D rectangle.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

`boolean raised` : a boolean that states whether the rectangle is raised or not

**public void fill3DRect(int x, int y, int width, int height, boolean raised)**
Paints a highlighted 3-D rectangle using the current color.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

`boolean raised` : a boolean that states whether the rectangle is raised or not

**`public void drawOval(int x, int y, int width, int height)`**

Draws an oval inside the specified rectangle using the current color.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

**`public void fillOval(int x, int y, int width, int height)`**

Fills an oval inside the specified rectangle using the current color.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

**`protected void doArc(int x, int y, int width, int height, int startAngle, int arcAngle, boolean fill)`**

**Parameters**

`int x` :

`int y` :

`int width` :

`int height` :

`int startAngle` :

`int arcAngle` :

`boolean fill` :

**`public void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)`**

Draws an arc bounded by the specified rectangle from startAngle to endAngle. 0 degrees is at the 3-o'clock position.Positive arc angles indicate counter-clockwise rotations, negative arc angles are drawn clockwise.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

`int startAngle` : the beginning angle

`int arcAngle` : the angle of the arc (relative to startAngle).

**public void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)**
Fills an arc using the current color. This generates a pie shape.

**Parameters**

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the arc

`int height` : the height of the arc

`int startAngle` : the beginning angle

`int arcAngle` : the angle of the arc (relative to startAngle).

**protected void doPoly(int [] xPoints, int [] yPoints, int nPoints, boolean fill, boolean close)**

**Parameters**

`int [] xPoints` :

`int [] yPoints` :

`int nPoints` :

`boolean fill` :

`boolean close` :

**public void drawPolyline(int [] xPoints, int [] yPoints, int nPoints)**
Draws a polygon defined by an array of x points and y points.

**Parameters**

`int [] xPoints` : an array of x points

`int [] yPoints` : an array of y points

`int nPoints` : the total number of points

**public void drawPolygon(int [] xPoints, int [] yPoints, int nPoints)**
Draws a polygon defined by an array of x points and y points.

**Parameters**

`int [] xPoints` : an array of x points

`int [] yPoints` : an array of y points

`int nPoints` : the total number of points

**public void drawPolygon(Polygon p)**
Draws a polygon defined by the specified point.

**Parameters**

`java.awt.Polygon p` : the specified polygon

**public void fillPolygon(int [] xPoints, int [] yPoints, int nPoints)**
Fills a polygon with the current color.

**Parameters**

`int [] xPoints` : an array of x points

`int [] yPoints` : an array of y points

`int nPoints` : the total number of points

**public void fillPolygon(Polygon p)**

Fills the specified polygon with the current color.

**Parameters**

`java.awt.Polygon p` : the polygon

**public void drawString(String str, int x, int y)**

Draws the specified String using the current font and color. The x,y position is the starting point of the baseline of the String.

**Parameters**

`java.lang.String str` : the String to be drawn

`int x` : the x coordinate

`int y` : the y coordinate

**public void drawString(AttributedCharacterIterator ai, int x, int y)**

**Parameters**

`java.text.AttributedCharacterIterator ai` :

`int x` :

`int y` :

**public void drawChars(char [] data, int offset, int length, int x, int y)**

Draws the specified characters using the current font and color.

**Parameters**

`char [] data` : the array of characters to be drawn

`int offset` : the start offset in the data

`int length` : the number of characters to be drawn

`int x` : the x coordinate

`int y` : the y coordinate

**public void drawBytes(byte [] data, int offset, int length, int x, int y)**

Draws the specified bytes using the current font and color.

**Parameters**

`byte [] data` : the data to be drawn

`int offset` : the start offset in the data

`int length` : the number of bytes that are drawn

`int x` : the x coordinate

`int y` : the y coordinate

**protected boolean doImage(Image img, int x, int y, int width, int height, int sx, int sy, int sw, int sh, ImageObserver observer, Color bgcolor)**

**Parameters**

`java.awt.Image img` :

`int x` :

`int y` :

`int width` :

`int height` :

`int sx` :

`int sy` :

`int sw` :

`int sh` :

`java.awt.image.ImageObserver observer` :

`java.awt.Color bgcolor` :

**return** `boolean` :

**`public boolean drawImage(Image img, int x, int y, ImageObserver observer)`**

Draws the specified image at the specified coordinate (x, y). If the image is incomplete the image observer will be notified later.

**Parameters**

`java.awt.Image img` : the specified image to be drawn

`int x` : the x coordinate

`int y` : the y coordinate

`java.awt.image.ImageObserver observer` : notifies if the image is complete or not

**return** `boolean` :

**`public boolean drawImage(Image img, int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4, ImageObserver observer)`**

Warning this is not yet supported

**Parameters**

`java.awt.Image img` :

`int x1` :

`int y1` :

`int x2` :

`int y2` :

`int x3` :

`int y3` :

`int x4` :

`int y4` :

`java.awt.image.ImageObserver observer` :

**return** `boolean` :

**`public boolean drawImage(Image img, int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4, Color c, ImageObserver observer)`**

Warning this is not yet supported

**Parameters**

`java.awt.Image img` :

`int x1` :

`int y1` :

`int x2` :

`int y2` :

`int x3` :

`int y3` :

`int x4` :

`int y4` :

`java.awt.Color c` :

`java.awt.image.ImageObserver observer` :

**return** `boolean` :

**public boolean drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)**

Draws the specified image inside the specified rectangle. The image is scaled if necessary. If the image is incomplete the image observer will be notified later.

**Parameters**

`java.awt.Image img` : the specified image to be drawn

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

`java.awt.image.ImageObserver observer` : notifies if the image is complete or not

**return** `boolean` :

**public boolean drawImage(Image img, int x, int y, Color bgcolor, ImageObserver observer)**

Draws the specified image at the specified coordinate (x, y). If the image is incomplete the image observer will be notified later.

**Parameters**

`java.awt.Image img` : the specified image to be drawn

`int x` : the x coordinate

`int y` : the y coordinate

`java.awt.Color bgcolor` : the background color

`java.awt.image.ImageObserver observer` : notifies if the image is complete or not

**return** `boolean` :

**public boolean drawImage(Image img, int x, int y, int width, int height, Color bgcolor, ImageObserver observer)**

Draws the specified image inside the specified rectangle. The image is scaled if necessary. If the image is incomplete the image observer will be notified later.

**Parameters**

`java.awt.Image img` : the specified image to be drawn

`int x` : the x coordinate

`int y` : the y coordinate

`int width` : the width of the rectangle

`int height` : the height of the rectangle

`java.awt.Color bgcolor` : the background color

`java.awt.image.ImageObserver observer` : notifies if the image is complete or not

**return** `boolean` :

**`public void dispose()`**

Disposes of this graphics context. The Graphics context cannot be used after being disposed of.

**`public void finalize()`**

Disposes of this graphics context once it is no longer referenced.

**`public String toString()`**

Returns a String object representing this Graphic's value.

**return** `java.lang.String` :

**`protected int transformY(int y)`**

Flip Y coords so Postscript looks like Java

**Parameters**

`int y` :

**return** `int` :

**`protected void emitProlog()`**

Top of every PS file

**`protected void emitColorImageProlog(int xdim)`**

**Parameters**

`int xdim` :

**`public void gsave()`**

**`public void grestore()`**

**`public void emitThis(String s)`**

**Parameters**

`java.lang.String s` :

**`protected void diagnostic(String s)`**

**Parameters**

`java.lang.String s` :