
Documentation

Developer

CHÈZE THIBAUT, PLANSON SEBASTIEN

Contents

I	Documentation	5
1	UML diagrams	6
1.1	AntHill package	6
1.1.1	Inheritances	6
1.1.2	Dependences	6
1.1.3	Associations	6
1.2	ConfigurationParser package	6
1.2.1	Inheritances	6
1.2.2	Dependences	7
1.2.3	Associations	7
2	Description of classes	8
2.1	AntHill	8
2.1.1	Simulator	8
2.1.2	AntHill	8
2.1.3	Board	8
2.1.4	Element	9
2.1.5	Display	9
2.1.6	Bug	9
2.1.7	Ant	9
2.1.8	Queen	9
2.1.9	Insect	9
2.1.10	AStar	9
2.1.11	Food	9
2.1.12	LocationCache	10
2.1.13	God	10
2.1.14	GodOfAnts	10
2.1.15	GodOfInsects	10

2.1.16	BeastsManager	10
2.1.17	AntsManager	10
2.1.18	InsectsManager	10
2.1.19	BoardManager	10
2.2	Configuration files parser	11
2.2.1	XMLConfigurationParser	11
2.2.2	XMLConfigurationHandler	11
2.2.3	ConfigurationValue	11
3	Implementation details	12
3.1	Implementation choice	12
3.2	Howto create a new GodOfAnts ?	12
3.3	Howto create a new Bug ?	12
3.4	Howto create a new BeastsManager ?	13
II	Appendix	15
A	Information on this book	16

Part I

Documentation

Chapter 1

UML diagrams

To see the UML diagram, you need *eclipse* program and its plugin *omondo*.

1.1 AntHill package

1.1.1 Inheritances

See the file "anthill-inheritance.ucd" in the directory :
" *UML-diagrams/fr.ones.kribleur.anthill*".

1.1.2 Dependences

See the file "anthill-dependences.ucd" in the directory :
" *UML-diagrams/fr.ones.kribleur.anthill*".

1.1.3 Associations

See the file "anthill-associations.ucd" in the directory :
" *UML-diagrams/fr.ones.kribleur.anthill*".

1.2 ConfigurationParser package

1.2.1 Inheritances

See the file "configurationparser-inheritance.ucd" in the directory :
" *UML-diagrams/fr.ones.kribleur.anthill.configurationparser*".

1.2.2 **Dependences**

See the file "configurationparser-dependences.ucd" in the directory :
"*UML-diagrams/fr.ones.kribleur.anthill.configurationparser*".

1.2.3 **Associations**

See the file "configurationparser-associations.ucd" in the directory :
"*UML-diagrams/fr.ones.kribleur.anthill.configurationparser*".

Chapter 2

Description of classes

For more details, see the javadoc ! Thank you.

2.1 AntHill

2.1.1 Simulator

This classe is public. It contains the width and the height of the board, and accesseurs allowing other out classes to have access there. It create Anthill with parsed options in parameter. After having initialize Anthill it run the party with launch the insects' and ant's gods.

2.1.2 AntHill

This classe is package. It allows to link lists of elements. Its constructor use parsed options for initialize Bugs and board Elements which can't move are initialized by Board. Anthill stocks all bugs and their location and the grid of sprites for display the board.

This class internal classes to be transmissent to the gods, it is the interface between the gods and bugs!

2.1.3 Board

This classe is package. Board initialize the grid with a file which contains the list of sprites of the grid. It contains methods relative to unmovable elements.

It is also that starts threads dealing with them reappear with food !

2.1.4 **Element**

Element is a package enum. Its visibility is package. It contains all sprites use for the grid and methods for add or remove a sprite.

2.1.5 **Display**

This classe is package and use by Anthill for display the grid.

2.1.6 **Bug**

This classe is abstract. Its contains common methods and variables in Insect, Ant and Queen : A location, a sprite, movespeed and the method move.

This class contains the code common to the different species. All bugs are responsible for their display on the map !

2.1.7 **Ant**

Ant extends Bug and its visibility is package. It contains specific method and variables for ants. Additional methods and variables allow to regurgitate, lose life points and die.

2.1.8 **Queen**

Queen extends Ant and its visibility is package. It contains specific method and variable for the queen. Additional methods and variables allow to create new ants.

2.1.9 **Insect**

Insect extends Bug and its visibility is package. It contains specific method and variables for insects. Additional methods and variables allow to attack ants.

2.1.10 **AStar**

This class allows a calculation of optimized route because it knows the map...

2.1.11 **Food**

This class has been managing food on the map, add, delete...It is Thread Safe !

2.1.12 LocationCache

It is a Thread Safe cache implementation for location. Please use it for any new Location you need, your code will only get faster !

2.1.13 God

This is the mandatory minimum interface to create a new god. Implement it !

2.1.14 GodOfAnts

It is the public interface to create a god of ant. If someone outside the package creates a god of ant, he will inherit this class !

2.1.15 GodOfInsects

It is the internal god of insects.

2.1.16 BeastsManager

This is the object that we change the gods to have access to the map and their animals. To add functionality, you have inherited from the class and force the cast in the god who will use it. For more details see the code of classes AntManager¹, and GodOfAnts²...

2.1.17 AntsManager

Implement BeastsManager³, it added methods needed to the god of ants.

2.1.18 InsectsManager

Implement BeastsManager⁴, it added methods needed to the god of insects.

2.1.19 BoardManager

BeastsManager⁵ extends it, it defined all methods to discover the map.

¹ 2.1.17

² 2.1.14

³ 2.1.16

⁴ 2.1.16

⁵ 2.1.16

2.2 Configuration files parser

2.2.1 XMLConfigurationParser

It is a SaxParser...

2.2.2 XMLConfigurationHandler

It Is the real parser ! To parse a tag defined the internal class Tag and add it in the map. To parse an attribute defined the internal class Attribute and add it in a Tag. To get the tag and attributes, modify the class ConfigurationValue, she represent the parsed values ! Good luck !

2.2.3 ConfigurationValue

It contains all parsed values. If you change the XMLConfigurationHandler, change it !

Chapter 3

Implementation details

3.1 Implementation choice

We have not made a big Array where all cases contains all their datas to avoid the redundancy of information. Different elements are separately stocked and linked in Anthill. Every categories of elements are stocked on the same place.

Obstacles are stocked in an HashSet of location allow to access in a constant time. Location of obstacle can't is the same.

Ant and Insect are stocked in two HasMap for link a bug to an identifying and an identifying to a location. It allow to access in a constant time or according to the number of ants or insects at the location.

The identifying is an int the maximum of ants on the grid is limited.

3.2 Howto create a new GodOfAnts ?

You are out of the package ! To create a new GodOfAnts¹, extend the class GodOdAnts in the package and implement the method *void play(AntsManager)*. Use the AntsManager in argument to lead ants and the queen.

3.3 Howto create a new Bug ?

You are in the package ! Extend Bug or a sub class, for the compute task² use *submit* or *execute* methods, for all access in internal, please use *readLock()* / *readUnlock()* and *writeLock()* / *writeUnlock()* methods depending

¹ 2.1.14 on page 10

²move for example

on the type of access made to protect your job, and for access to superclass information, use the accessors...

The protected accessors are `getClassNameJob()`, for example : to get the ant's life points in internal¹, the accessor is `getAntLifePoints()` !

To manage your new bug, create a god² for this and the adapted `BeastManager`³

3.4 Howto create a new BeastsManager ?

You are in the package ! Extend `BeastsManager`, implements all methods in `BeastsManager` and add your own methods ! This is only useful if you have any methods to be added, new animals to lead and a god who wait this !

¹And inherit

² 2.1.13 on page 10

³ 2.1.16 on page 10 and 3.4

Part II

Appendix

Appendix A

Information on this book

- It were Thibaut Chèze and Sebastien Planson who took the initiative in creating this document.
- The language that allowed its writing is L^AT_EX.
- All informations contained in this document are accurated on the date of January 6, 2008