Guide d'utilisateur

Groupe numéro 20

1 Contexte

Le présent document a pour but de présenter comment fonctionne notre projet de programmation orientée objet visant à mettre au point une application simulant une équipe de robots pompiers. Il s'adresse à toute personne souhaitant tester le fonctionnemnt de notre projet.

2 Prérequis

Ce projet est entièrement codé en Java, il est donc préférable d'avoir des connaissances dans ce language ou du moins en programmation orientée objet pour une bonne compréhensio des tenants et des aboutissant de celui-ci. Assurez de bien avoir installé une machine virtuelle Java afin de pouvoir compiler notre code.

3 Contenu

Si vous lisez ce document c'est surement que vous êtes en possession de tout le codes source nécessaire à l'exécution et au bon fonctionnement de notre application. Nous avons décidé de construire notre application autour de plusieurs packages. - Un package objets contenant toutes les entités, i.e tous les objets et les classes filles de la classe Robot qui représentent les différents types de robots. - Un package io contenant seulement la classe LecteurDonnees qui fait le lien entre l'entrée utilisateur et le simulateur. - Un package simulation qui contient les classes nécessaires à afficher la carte et à effectuer la simulation. - Un package Aetoile contenant la classe Astar qui contient toutes les méthodes nécessaires à calculer le plus court chemin. - Un package evenements contenant la classe abstraite evenements qui implémente les différents évènements.

4 Exécution

4.1 Tests élémentaires

Deux tests basiques vous sont fournis. Le premier permet de vérifier que lorsque le robot sort de la carte, un message d'erreur s'affiche bien dans le terminal (test

KO). Le deuxième permet de vérifier que les robots se déplacent bien comme il faut et qu'ils effectuent des interventions (test OK). Dans ces tests, les robots n'éteignent pas les incendies et c'est normal.

Ces tests ne sont pas très convaincants mais vous pourrez quand même les lancer en effectuant la suite de commandes depuis la racine du projet.

```
make clean
make testKO
make exeKO
```

pour le test KO, et pour le test OK

```
make clean
make testOK
make exeOK
```

4.2 Tests complets

Nous avonc mis au point des programmes permettant de tester que nos robots éteignent bien tous les incendies selon une des trois attributions possibles.

Afin de lancer ce test il faut ouvrir un terminal et se positionner à la racine de notre projet. Nous vous conseillons de commencer par supprimer les éventuels fichiers .class présent dans le dossier bin avec la commande :

```
make clean
```

Ensuite, pour compiler tout le code nécessaire au bon fonctionnement de notre application, exécuter la comande suivante :

```
make testChefPompier
```

Elle va compiler tous les fichiers .java et placer les .class associés dans le dossier bin

Enfin, pour exécuter le tout lancer la commande :

```
make exeChefPompier FILE={Chemin/vers/le/fichier.map} ATTRIBUTION={Type_attribution_robots}
```

Le paramètre FILE correspond au chemin vers le fichier de données que vous voulez simuler. Le paramètre ATTRIBUTION permet de choisir la manière avec laquelle le chef pompier va alouer les incendies aux robots. Pour celui-ci vous avez le choix entre : simple, avancee ou reflechie.

Un exemple d'exécution sur la carte "cartes/mushroomOfHell-20x20.map" avec l'attribution la plus efficace ("réfléchie") donne :

```
make exeChefPompier FILE=cartes/mushroomOfHell-20x20.map ATTRIBUTION=reflechie
```

Une fois l'application lancée, nous vous conseillons de diminuer le temps entre deux affichages pour ne pas attendre trop longtemps, et au besoin d'augmenter le nombre de pas simulés entre 2 affichages si cela prend vraiment trop de temps.

Si vous voulez générer la documentation Java, executer :

```
javadoc -d docs -sourcepath src  -subpackages objets simulation io evenements -classpath
lib/gui.jar
```