PROJET ROBOTWAR

RAPPORT DE SYNTHESE 2016

Groupe : BEN HAMOUDA Rihab , GIMENEZ Réjane, LOUM Mor

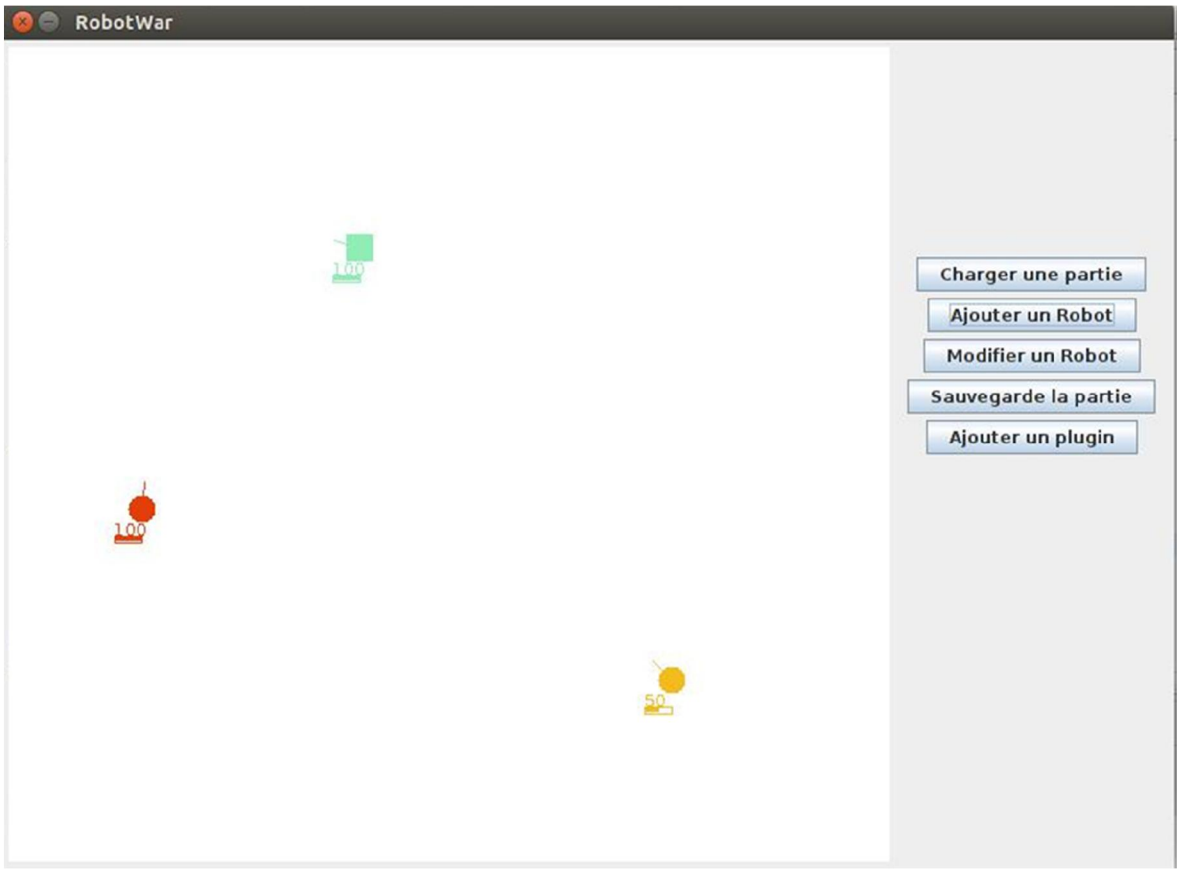
TABLE DES MATIERES

Contenu

INTRODUCTION

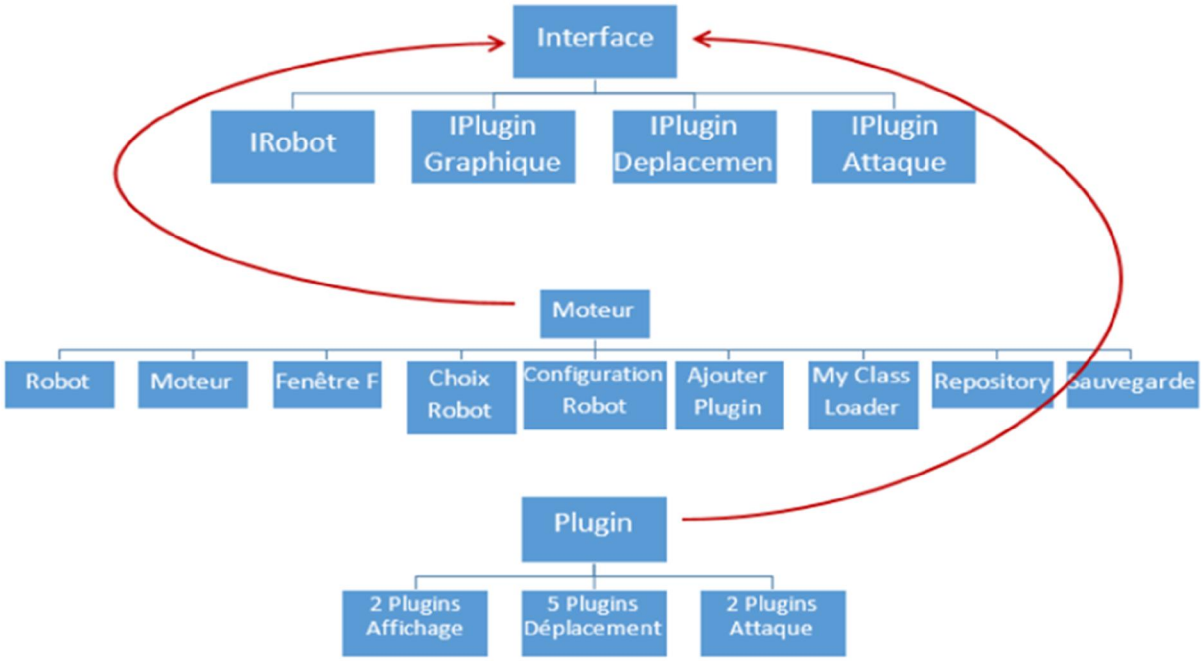
Introduction

Nous avons eu pour objectif de construire un jeu de combat virtuel de type RobotWar. Le comportement des robots du jeu tel que son graphisme, son déplacement et ses attaques est entièrement géré par des plugins. A chaque robot est associée une barre de vie qui va diminuer lors de l'attaque d'un robot sur un autre. Lorsque lancer, le jeu n'affiche aucun robot, l'utilisateur a la possibilité de les ajouter au fur et à mesure de la partie. Il sera également en mesure de modifier les comportements des robots en pleine partie ainsi qu'ajouter de nouveau comportement en ajoutant des nouveaux plugins. Le jeu possédera un système de sauvegarde qui permettra à l'utilisateur de sauvegarder des parties et de pouvoir les retrouver dans les mêmes états qu'au moment de la sauvegarde.



ARCHITECTURE DU PROJET

Architecture du projet



Dépendance

Nous avons créé un projet Maven en trois modules. Le module Plugin est Moteur sont dépendant du module Interface.

LES ETAPES DE LA CREATION DE NOTRE PROJET

Les étapes de la création de notre projet

1) Etape 1 :Création de notre ClassLoader .

Le TP3 de programmation avancée consistait à créer notre propre ClassLoader. Dans un premier temps, nous avons créés la classe “MyClassLoader” afin de pouvoir charger une classe puis la classe “Repository” pour pouvoir ensuite charger toutes les classes comprises dans un répertoire donné. Une fois ces deux classes terminées en TP, nous les avons ajoutées à notre projet afin de pouvoir lui charger les plugins.

Cette étape 1 a été réalisée par : BEN HAMOUDA Rihab

2) Etape 2 : Création d’une classe Moteur et d’une Classe Robot.

La deuxième étape a pour objectif de créer, dans un premier temps, une classe robot qui permettra de déterminer les caractéristiques du robot grâce aux attributs et méthodes qui lui seront attribués.

- Nos robots possèdent les attributs suivants :
  - Une couleur qui lui est attribué de manière aléatoire .
  - Une position qui est également attribué aléatoirement dans la zone jouable(JPanel).
  - Un niveau de vie initialisé à 100.
  - Un cap, qui sera initialisé par le plugin de déplacement associé ou, à défaut, à 0.
  - Une vitesse, également initialisé à 15.
  - Un plugin graphique.
  - Un plugin déplacement.
  - Un plugin attaque, initialisé à une distance de 50 entre les robots pour qu’ils s’attaquent.

- Cette classe possède 2 constructeurs :
  - Un constructeur Robot(), qui ne prend aucun paramètre
  - Un constructeur Robot(IPluginDeplacement d, IPluginGraphique g, IPluginAttaque a) et qui attribuera aux robots les plugins associés.

  C’est dans les constructeurs que sont initialisés tous les attributs.

LES ETAPES DE LA CREATION DE NOTRE PROJET

- La classe Robot contient tous les “setters” et “getters” des attributs qu’elle possède ainsi que des méthodes :

  -attaque(ArrayList<IRobot> listRobots) ;

  -dessiner(Graphics g);

  -deplacement() ;

  Ces méthodes implémentent les méthodes des plugins choisis.

  La classe moteur, quand à elle, possède à ce stade d’avancement les attributs ArrayList<IRobot> listRobots et FenetreF fenetre, FenetreF étant une classe héritant de Jframe, n’affichant encore qu’une simple fenêtre ou le jeu pourra être joué. Elle utilise une méthode gestionDesTours() afin de faire tourner notre jeu. Les “setters” et les “getters” des attributs y sont ajoutés et ces dernier sont également initialisés dans le constructeur Moteur().

  Cette étape a été réalisée par : GIMENEZ Réjane

  3) Etape 3 : Création des interfaces et des plugins de graphique et de déplacements.

- Nous créons ensuite une première interface IRobot avec les méthodes de la classe Robot que les Plugins auront besoin (ex : setVie(), getCouleur() , getPosition() etc…) et implémentons cette interface à la classe Robot. Nous ajoutons alors au module “Interface”, trois nouvelles interfaces :
  - IPluginGraphique
  - IPluginDeplacement
  - IPluginAttaque

  Chacune d’entre elles déclare une méthode, respectivement: void paint(Graphics g, IRobot robot), Point deplacement(IRobot robot, int vitesse), void attaque(IRobot robot, ArrayList<IRobot> listRobots). Ces méthodes pourront être utilisées par la classe robot dans les méthodes attaque, dessiner et déplacement.

  Nous pouvons alors créer un nouveau constructeur : Robot(IPluginDeplacement d, IPluginGraphique g, IPluginAttaque a) afin de pouvoir attribuer directement les plugins du robot.

LES ETAPES DE LA CREATION DE NOTRE PROJET

Nous complétons cette étape avec la création des premiers plugins : Un plugin graphique AfficherRobotCarre, qui affiche un robot carré (côté 20 pixels) avec la couleur et la position définies par la classe Robot. Puis deux plugins de déplacement: DeplacementHorizontal et DeplacementVertical.

Par la suite, au fur et à mesure que nous avançons, d'autres plugins ont été créés.

Cette étape a été réalisée par : GIMENEZ Réjane et BEN HAMOUDA Rihab.

4) Etape 4 : Création de l'interface Graphique

Après avoir créé la classe FenetreF héritant de JFrame, nous créons à l'intérieur de celle ci un JPanel , qui correspond à la zone de jeu pour les Robots. Nous ajoutons ensuite des boutons qui permettent de jouer au jeu, configurer les robots, ajouter un plugin, charger une partie ou encore la sauvegarder.

La classe ChoixRobot créé une boîte de dialogue, avec 3 JComboBox. Le constructeur de cette classe prend quatre paramètres dont le titre de la boîte de dialogue et un élément de type List<Class>, qui correspond à la liste des plugins.

D'où le fait de créer une méthode getList() à l'intérieur de la classe fenetre, pour récupérer la liste des plugins en invoquant la classe Repository. Les 3 JComboxBox correspondent aux 3 types de plugins (graphique, déplacement,attaque) que l'on identifie grâce à leur interface. Cette classe permet de créer un par un les robots, selon les plugins souhaités.

La classe ConfigurerRobot lance une autre boîte de dialogue qui permet de modifier un robot de notre choix en activant ou désactivant les plugins. Cette classe contient 4 JComboBox, le 4eme attribue un numéro à chaque robot, selon l'ordre dont il ont été créé chronologiquement, pour pouvoir connaître quel robot sera exactement modifié.

La Classe AjouterPlugin renvoie elle aussi une boîte de dialogue avec un JTextField. Cette classe permet de rajouter des plugins dans le répertoire à plugin directement depuis l'interface graphique, en tapant le chemin complet du plugin, ou de l'archive de plugin que l'on souhaite ajouter.

Ces trois classes possèdent chacune un bouton qui permet de les invoquer, dans l'interface graphique, en plus de deux autres boutons "Sauvegarder" et "Charger une partie" qui assurent la persistance.

Cette étape a été réalisé par : BEN HAMOUDA Rihab

LES ETAPES DE LA CREATION DE NOTRE PROJET

5) Etape 5 : Mécanisme de gestion de plugins et de chargement dynamique

Cette étape est liée à la précédente : Les classes ChoixRobot et ConfigurerRobot créent chacune une boîte de dialogue dynamique. Elles prennent toutes les deux en paramètre de leur constructeur un objet de type List<Class> qui correspond à la liste des plugins disponibles dans le répertoire de plugin. Cette liste est générée par la méthode getList() qui utilise la classe Repository.

La classe ChoixRobot possède un attribut ArrayList<IRobot> vide qui va se remplir grâce à la classe anonyme ActionListener, qui génère l'action du bouton "Ajouter un robot". Dans celle ci, nous allons récupérer les 3 plugins sélectionnés pour chaque catégorie. En utilisant la réflexivité avec des méthodes tel que getConctructeur() ou newInstance() ; pour créer une nouvelle instance du plugin sélectionné et pouvoir créer une nouvelle instance de robot avec pour paramètre une instance du plugin. Une fois le robot créé, il va être ajouté à la ArrayList<IRobot> qui sera ensuite récupéré par la classe FenetreF puis la classe Moteur de manière synchronisé afin que les robots puissent se dessiner, se déplacer et attaquer, dans le gestionnaire de tours.

La classe ConfigurationRobot fonctionne presque sur le même principe que ChoixRobot, elle récupère la liste des robots déjà créé à qui elle attribue un numéro selon l'ordre de création et dans l'ActionListener du bouton "Configurer le robot" : elle utilise les méthodes setPlugin….() de la classe Robot pour pouvoir activer ou désactiver des plugins.

La classer AjouterPlugin, quand à elle, récupère un chemin absolu sous forme de string que l'on utilise pour créer un new File(...). La classe va lire les bytes du fichier source pour les écrire dans le fichier destinataire qui se situe donc dans le répertoire des plugins grâce à FileInputStream et FileOutputStream: un nouveau plugin est créé. On peut aussi y ajouter des archives zip et jar.

Cette étape a été réalisé par : BEN HAMOUDA Rihab

6) Etape 6: La persistance

Dans notre projet, la persistance est assuré grâce à la classe Sauvegarde,. Elle contient deux méthodes : -La première save() assure la sauvegarde de l'état d'un jeu en enregistrant sous forme de byte les données du jeu.

-La deuxième load() permet de lire les fichiers précédemment sauvegardé et relance le jeu dans l'état où il était au moment de la sauvegarde.

LES ETAPES DE LA CREATION DE NOTRE PROJET

Les sauvegarde sont enregistrés avec l'extension ".rw" (initiale de RobotWar) pour être retrouver plus facilement, les 2 méthodes renvoie un JChooserFile qui n'affiche que les fichiers se terminant par cette extension.

Les classes qui ont du être séréalisé/déséréalisé ont implémenter l'interface Serializable et ont un ServialID généré pour eux.

Cette étape a été réalisé par :BEN HAMOUDA Rihab

7) Etape 7: Le JAR Exécutable

On veut maintenant créer un livrable sous la forme d'un JAR exécutable qui va englober les différentes dépendances liées à notre projet. Pour ce faire, nous avons configuré Maven en y ajoutant le plugin maven-assembly-plugin. Nous avons ensuite modifié le Pom du Moteur, puis, à l'aide du terminal, nous avons créé le .JAR avec la commande mvn package. Ce dernier se trouve alors dans le dossier moteur sous le nom de Moteur-1.0-SNAPSHOT.jar.

Commande sur le terminal pour ouvrir le JAR : java -jar Moteur-1.0-SNAPSHOT.jar

Cette étape a été réalisée par : GIMENEZ Réjane

La Répartion du travail

Nous étions 3 sur ce projet à la base. La 3ème personne est LOUM Mor mais ce dernier n'a malheureusement pas été présent lors de l'avancement du projet. Il avait pourtant l'air motivé et était présent à toutes les "réunions" que nous avons mise en place afin de suivre l'avancement de chacun. Sur le git, nous avons mis des tickets afin de répartir le travail, ce qui n'a pourtant pas changé l'implication de Mor dans notre projet.

DESCRIPTION DES PLUGINS

Description des Plugins

PLUGINS GRAPHIQUES

-AffichageRobotCarré : Ce plugin permet de dessiner des robots carrés à une position aléatoire, avec une couleur aléatoire comme défénis dans la classe Robots.Il affiche également le cap ainsi qu'une part de vie et le niveau de vie en chiffre.

-AffichageRobotCercle : Ce plugin permet de dessiner des robots ronds à une position aléatoire, avec une couleur aléatoire comme défénis dans la classe Robots.Il affiche également le cap ainsi qu'une part de vie et le niveau de vie en chiffre.

PLUGINS DEPLACEMENTS

-DeplacementHorizontal : Ce plugin permet d'atribuer un mouvement horizontal au robot. Le robot ne dépasse jamais les bords. Lorsqu'il touche les bords il fait demi-tour.

-DeplacementVertical : Ce plugin permet d'atribuer un mouvement vertical au robot. Le robot ne dépasse jamais les bords. Lorsqu'il touche les bords il fait demi-tour.

-DeplacementAleatoire : Ce plugin affecte un mouvement aléatoire au robot.Son cap est fixé de manière aléatoire à chaque tour et il se déplace en fonction de celui-ci. Comme tous les autres plugins il ne dépasse pas les bords.

-DeplacementLineaire : Ce plugin assigne un cap aléatoire au robot et celui-ci se déplacera de manière diagonale en fonction du cap. Il ne dépasse pas les bords et fait demi-tour lorsqu'il atteint un bord.

-Immobile : Ce plugin dessine un robot immobile avec un cap changeant.

PLUGINS ATTAQUES

-AttaqueAleatoire : Ce plugin permet au robot d'attaquer les autres robots qui sont à une distance inférieur à 100 de lui.Il se fait également attaquer lorsqu'il approche un autre robot.

-Inoffensif : Ce plugin n'attribue aucune attaque aux robots mais ceux-ci peuvent être attaquer par des robots avec un plugin attaque fonctionnant.

Procéduire pour créer de nouveaux plugins :

Les plugins pourront être ajouté directement depuis l'interface graphique du jeu mais devront implémenter l'interface correspondant au type de plugin créé ainsi que ces méthodes et doit posséder pour package unice.miage.m1.projet . Autrement il ne saura pas reconnu comme un package.

CONCLUSION

Conclusion

Malgré nos multiples efforts pour obtenir un véritable travail de groupe, une personne n'a malheuresement pas contribué au projet, l'avancement du être entièrement menés deux. Malgré cela le travail a été délivré dans les temps et toutes les conditions et exigences du projets ont été satisfaites.

La procédure à suivre afin de tester notre projet est la suivante:

- Il faut tout d'abord Run le App.java du projet Moteur, généré par maven, ou bien lancer directement le JAR executable qui se trouve dans Moteur sous le nom de Moteur-1.0-SNAPSHOT.jar;
- Une fenêtre apparaît ensuite, générée par la classe JFrame. Il faut cliquer sur "Ajouter un Robot";
- C'est le moment de choisir l'apparence, le déplacement et l'attaque du robot. Une fois les choix faits, il faut cliquer sur "Créer le Robot";
- Le nouveau Robot créé apparaît sur le JPanel. Pour que le projet devienne un jeu, il faut plusieurs robots, donc il faut ajouter de nouveaux robots afin de voir un gagnant, ce qui est évident.