BELBECIR Riyadh Abbes
N° étudiant: P1926605
riyadh.belbecir@etu.univ-lyon1.fr

Université Claude Bernard Lyon 1
Département d'Informatique
Compétences Complémentaires en Informatique

Programmation Orientée Objet
JAVA
Le rapport final du projet -ROBOT-
25/02/2022

Contexte et objectifs

Le projet -Robot- fait partie d'un package académique à pour but d'apprendre le paradigme Orienté Objet en programmant avec langage JAVA.

Il consiste principalement à réaliser une interface graphique avec une grille de cases considérées comme des positions avec des robots qui bougent dedans et qu'ils ont la capacité de polluer la case (fond noir) ou de la nettoyer (fond blanc).

Il était un projet super amusant et en même temps trop didactique qui m'a permet d'acquérir le savoir nécessaire afin d'entamer tout ce qui est GUI et dessin 2D sans aucun doute ou sous-estime.

La description de l'interface graphique

L'interface graphique (Figure 1 et Figure 2) a été réalisé en utilisant les deux bibliothèques de JAVA: **Swing** et **AWT**, elle est divisée en cinq zones: la centrale contient l'arène où les robots bougent et appliquent leurs fonction de nettoyage ou de pollution. Une zone d'en-tête qui contient un indicateur de taux de pollution. A gauche, un bouton qui "CLEAN ARENA" qui remet l'état de l'arène à propre, deux boutons qui change la vitesse du robot pollueur libre (cf. Les classes fonctionnelles) (des Timers ont été utilisées) et un bonus qui est un quiz rapide de cinq questions sur l'informatique et tout en bas quelques informations utiles. A droite, un bouton qui ferme l'application et un autre qui montre comment les robots bougent, et finalement, un en-pied qui contient des boutons d'ajout ou de suppressions des robots.

L'interface à été réalisé d'une façon qu'elle soit "resizable" aussi l'utilisation des LayoutMAnagers est presque nulle alors que tout le positionnement des éléments et le recalcule à été fait d'une façon "manuelle" (beaucoup plus dur mais elle m'a permit d'apprendre beaucoup plus).

La gestion des collisions

Afin de gérer les collisions une matrice booléenne associé au monde et au robots indique vrai ci la case est occupée par un robot ou faux sinon. Donc si la case est occupé le robot ne bouge pas. Aussi le programme a été réalisé d'une façon que les robots ne sortent jamais de leurs monde.

La structure du programme

Le programme est constitué principalement de trois catégories de classes:

- **Les classes graphiques:**
  - La classe **myGUI**: qui prend en charge le dessin de la fenêtre principale (JFrame) qui est divisée en 5 zones (JPanel): un panel principale qui a la même taille que la fenêtre principale et les 4 qui restent sont un panel à droite, un panel à gauche, un panel d'en-tête et un panel d'en-pied. Chacun d'entre eux contient des boutons avec des fonctionnalités différentes. Aussi, c'est elle qui crée l'instance du **Monde** et **DessineTerrain** (cf. Les classes suivantes).
  - La classe **DessineTerrain**: c'est la classe qui prend en charge le dessin du panel central qui contient la grille des cases qui sont susceptibles d'être polluer ou nettoyer et aussi l'affichage des mouvements des robots.
- **Les classes fonctionnelles:**

prend en charge les fonctions des robots et leurs interactions dans le monde d'où ils appartiennent:

  - La classe **Monde**: elle définit la taille de la grille (nombre de lignes et de colonnes) ainsi que les listes de tous les robots qui font parties de l'instance d'un monde, l'état de la grille si elle est propre ou pas et aussi la gestion des collisions en précisant si une case est occupée par un robot ou pas.
  - La classe **Robot**: est une classe mère et abstraite qui définit les éléments communs entre tous les types de robots tels que le nom, la position, l'image, le monde, le terrain. Elle contient aussi des méthodes qui prennent en charge le positionnement initial des robots.
  - Les 5 classes qui hérites de la classe **Robot**. Elles sont catégorisées suivant deux comportement possibles: **Le type de la trace** (nettoyage ou pollution) et **Le type du mouvement** (déterministe ou stochastique):
    - la classe **RobotPollueurToutDroit**: c'est un robot à mouvement déterministe car il bouge du haut vers le bas dans une boucle tout en laissant une trace noir unique derrière lui.
    - la classe **RobotPollueurSauteur**: c'est un robot à mouvement stochastique car il bouge dans une des quatre directions (Haut, Bas, à droite, à gauche) d'une façon aléatoire tout en sautant une case et en laissant une trace noir unique derrière lui.
    - la classe **RobotPollueurLibre**: c'est un robot à mouvement stochastique,il bouge dans les huit directions qui l'entoure aléatoirement tout en laissant une trace noir unique derrière lui.
    - la classe **RobotNettoyeurHorizontal**: c'est un robot à mouvement déterministe car il bouge de la gauche vers la droite dans une boucle tout en laissant une trace blanche unique derrière lui.
    - la classe **RobotNettoyeurAutour**: c'est un robot à mouvement stochastique,il bouge dans les huit directions qui l'entoure aléatoirement tout en laissant huit traces blanches dans les cases qui l'entourent.

La classe principale:

  - la classe **MainClass:** c'est la classe d'entrée au programme, elle crée l'instance de la GUI (**myGUI**) et les instances des robots. Elle prend en charge aussi l'execution du programme d'une façon répétitive en utilisant l'interface JAVA **Runnable** et la classe **Timer** afin de controler l'image par seconde (FPS).

Des futures améliorations

- L'ajout d'un robot avec un comportement intelligent.
- L'ajout d'un robot contrôlé par le clavier.
- L'ajout de l'image d'un papier gras au lieu de seulement des cases blanches et noires.
- Mieux commenté le code.
- Mieux optimisé le code tout en remplaçant les lignes redondantes avec des méthodes et enlever les variables non utilisées.

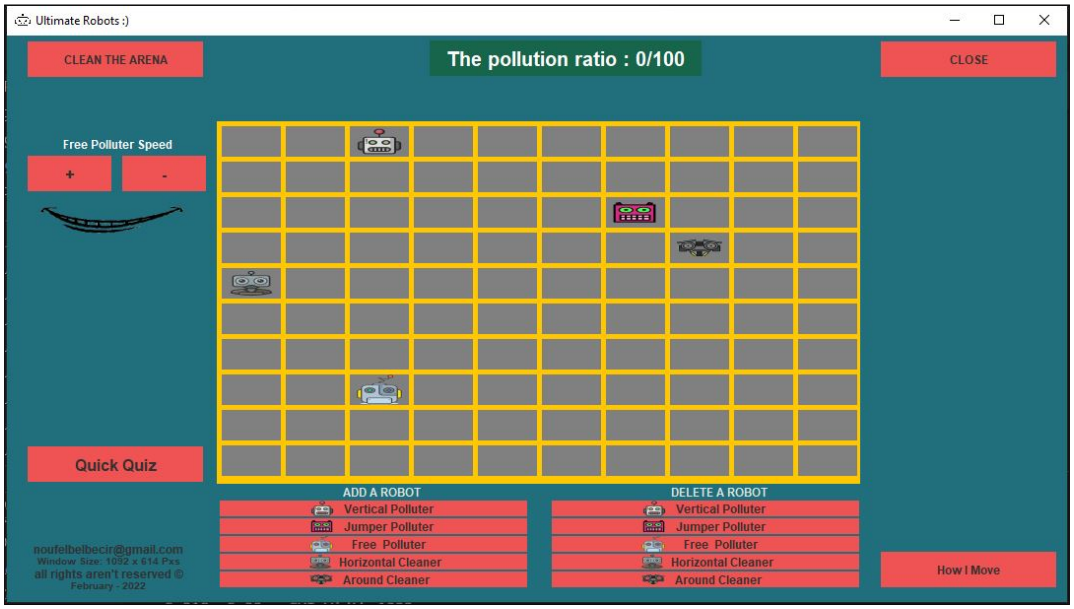

Figure 1: L'interface Graphique à l'état initiale

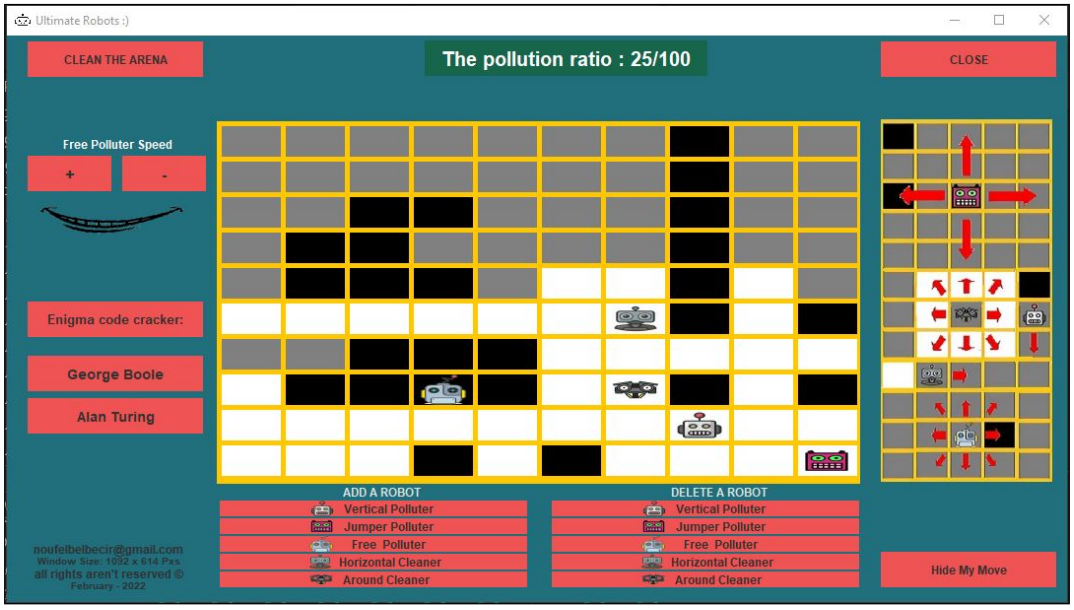

Figure 2: L'interface Graphique après un instant

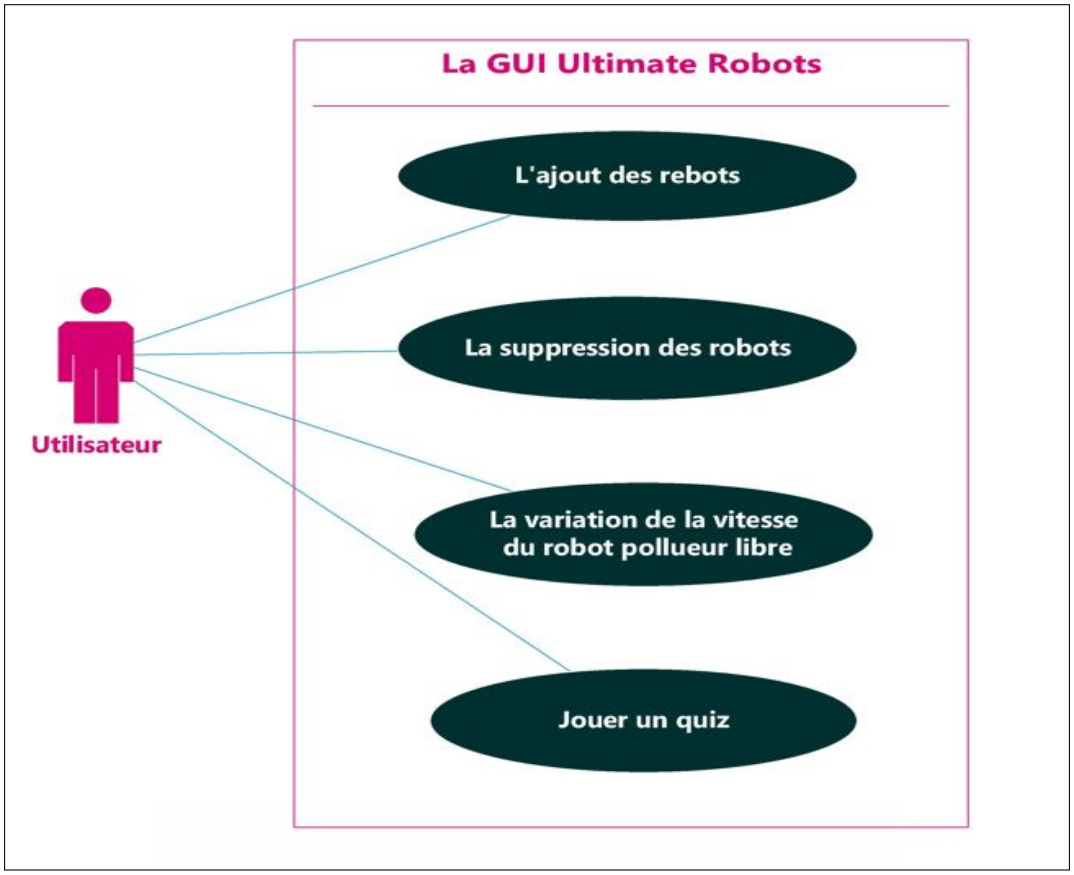

Figure 3: Le "use case" de l'application

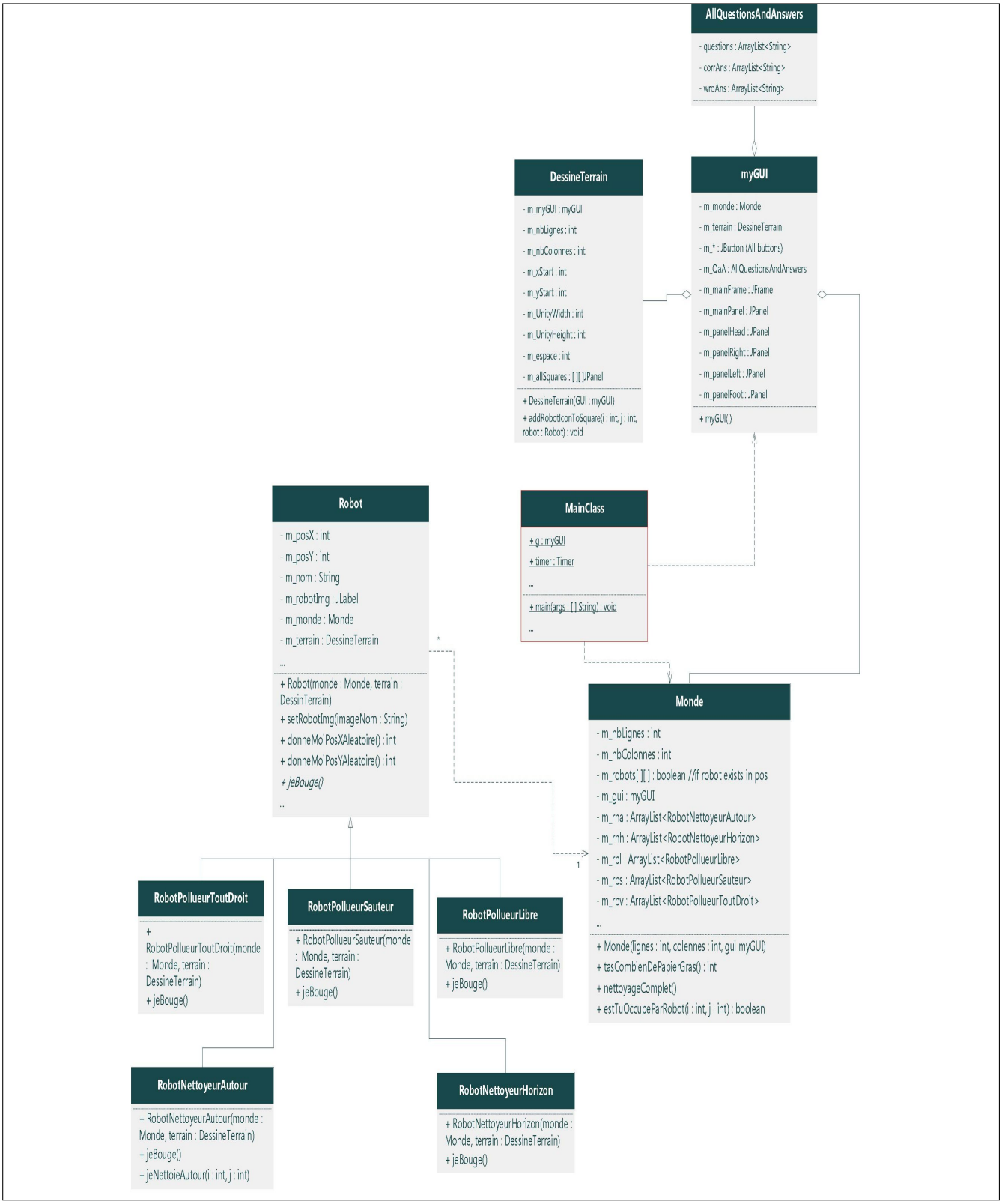

Figure 4: Le "class diagram" de l'application