isep
École d'ingénieurs du numérique

Rapport de Projet

A1: II.1102
Algorithmique et Programmation
Java

Jacques BONNAND
Clément BRISSE
Grégoire DEBRAY-GENTY
G4

Sommaire

I. Description du projet

A. Concept du jeu

Le but du module II.1102 est de faire travailler les étudiant sur le développement d'un jeu en Java. Cette année nous devons donc développer le jeu "Robot Turtles".

Robot Turtles est un jeu un de plateau conçu pour introduire des notions élémentaires de l'algorithmique à des jeunes enfants. Robot Turtles se joue de 2 à 4 joueurs, et chaque joueur incarne une tortue se déplaçant sur un plateau de taille 8×8.

L'objectif de chaque joueur est d'amener sa tortue sur un joyau placé sur le plateau en construisant un petit algorithme. Cet algorithme est construit à l'aide de cartes qui permettent de faire avancer la tortue ou de la faire tourner d'un quart de tour vers la gauche ou la droite. Des cartes supplémentaires permettent de créer des obstacles ou de les détruire.

Pour ce projet, nous utiliserons les règles "Galapagos" dont l'objectif est d'atteindre une gemme avant les autres joueurs. Un même joyau peut être atteint par plusieurs joueurs. Le jeu se termine quand il ne reste plus qu'un joueur en jeu.

B. Réalisation

Nous avons décidé de faire marcher le projet intégralement en console avant de se lancer dans l'interface graphique. Le seul point graphique pour le jeu en console est la sélection du nombre de joueur (2/3/4) au lancement du programme.

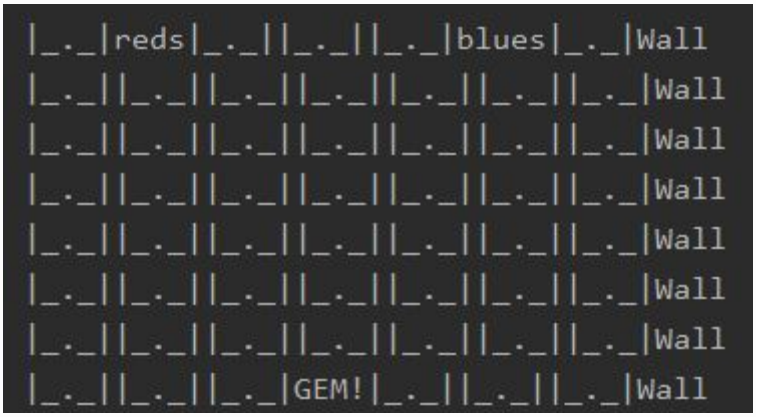
- Console: déroulement du jeu

Le programme, initialise le nombre de joueur en fonction du bouton sélectionné par l'utilisateur. Sont ensuite créé: les joueur, leur deck (qui est ensuite mélangé), leur main et leur tortue respective.

```
deck
deck shuffle
player1 done
deck
deck shuffle
player2 done
turtle colorred
turtle colorblue
```

cas pour une partie à 2 joueurs

Le plateau de jeu, qui diffère en fonction du nombre de joueur comme indiqué dans les figure suivantes est ensuite initialisé:



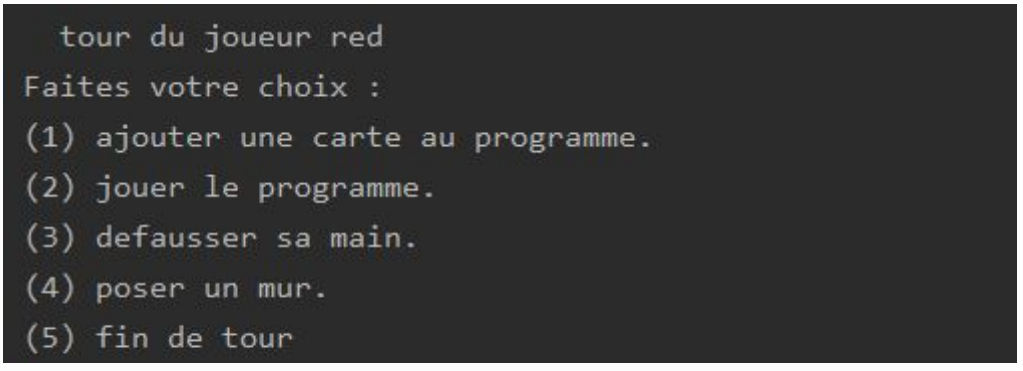
cas pour une partie à 2 joueurs

Légende:

- les tortues des joueurs sont indiquées par leur couleur en anglais (‘red’, ’blue’ etc), avec la dernière lettre indiquant la direction de la tortue (‘n’ pour nord, ‘s’ pour sud, ‘e’ pour est et ‘w’ pour ouest.
- Un mur est représenté par ‘Wall’ si c’est un mur de pierre ou ‘Walli’ si c’est un mur de glace.
- Une gemme est représentée par ‘GEM!’

Vient ensuite le tour du premier joueur (par défaut: le joueur rouge)
Le joueur accède à un choix d’action en fonction de la valeur qu’il tape en console:

- 1: ajouter une carte à la suite du programme
- 2: jouer le programme
- 3: défausser sa main
- 4: poser un mur
- 5: finir son tour



A noter que si le joueur effectue une des actions principale 1, 2 ou 4, son tour est considéré comme fini, il n’a donc plus que la possibilité de se défausser de sa main (3) et/ou de finir son tour (5). Si le joueur essaie d’effectuer une des actions principale à nouveau, il est directement redirigé vers le menu de choix afin de finir de

défausser sa main. Si le joueur commence par défausser sa main, il ne peut pas ajouter de cartes car il ne pioche que lorsque son tour est terminé.

Si le joueur choisit d'ajouter une carte au programme (1), la console renvoie la couleur des cartes actuellement dans sa main, accompagné d'un numéro entre 0 et 4 pour choisir une des 5 cartes. Il a aussi la possibilité de revenir en arrière en entrant "5" en console comme affiché.

```
  tour du joueur red
Faites votre choix :
(1) ajouter une carte au programme.
(2) jouer le programme.
(3) defausser sa main.
(4) poser un mur.
(5) fin de tour
1
blue (0)red (1)red (2)blue (3)yellow (4) go back (5)0
```

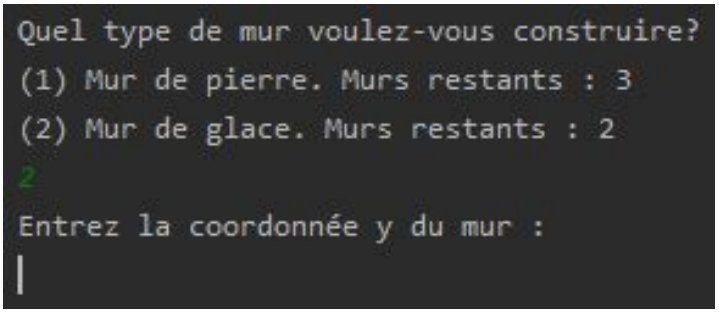
Correspondance couleur de carte et action de la tortue au lancement du programme:

- red: carte tirer un laser
- blue: avance d'une case
- yellow: tourner sur soi-même de 90° vers la gauche
- purple: tourner sur soi-même de 90° vers la droite

Exemple d’exécution du programme après avoir mis un certain nombre de carte dedans:

Le programme exécute dans l’ordre les cartes “red, purple, blue, yellow, blue, blue”, ce qui se traduit par: “laser, droite, avancer, gauche, avancer, avancer”. On remarque que la tortue red est bien à la bonne position après ce programme au vu de sa position de départ: première ligne, deuxième colonne et regardant vers le bas.

Si le joueur choisit de placer un mur, la console demande au joueur le type de mur ainsi que ses coordonnées



- Graphique

Nous avons effectué un certain nombre de test non concluant pour ajouter une interface graphique, nous somme donc resté à une version console fonctionnelle du projet.

II. Architecture du projet

A. Structure

- Structure du projet

Nous avons utilisé un répertoire git afin de faciliter le travaille collaboratif du groupe: https://github.com/jacques-bobo/RobotTurtle

- Descriptif des **classes** du projet:

Board:
Cette classe a pour but de créer le terrain de 8x8 tiles (cases) mais aussi d'initialiser la position des tortues, les gemmes et les murs sur le terrain en fonction du nombre de joueurs sélectionnés (2/3/4).
Il existe aussi une méthode "getBoard()" qui permet d'accéder à la matrice de tiles du terrain.

Card:
Cette classe a pour but de créer les cartes de couleur: "red", "blue", "yellow", "purple" ("rouge", "bleu", "jaune", "violet").
Elle possède des méthodes "setColor(String newVar)" et "getColor()" qui permettent respectivement d'initialiser la couleur d'une carte et de récupérer la couleur d'une carte.

Deck:
Cette classe a pour but de créer une liste de cartes, l'intègre en tant que deck pour chaque joueur avec le bon nombre de chaque carte de couleur et de mélanger l'ordre des cartes à l'aide de la méthode "deckshuffle(List deck)"
Elle possède une méthode "getDeck()" qui retourne le deck, soit la liste des cartes.

Discard:
Cette classe a pour but de créer la liste de carte défaussées par un joueur. Chaque joueur à sa propre défausse. Un deck vide est rempli à l'aide de la défausse. Cette défausse est initialisée vide.

Game:
Cette classe a pour but d'exécuter toutes les actions pendant le tour d'un joueur: afficher le menu de choix, récupérer le choix, exécuter le choix du joueur et de vérifier la condition de victoire en fonction de la position de la tortue du joueur après avoir exécuté le programme.
Elle possède une méthode "getPlayer_count()" qui retourne le nombre de joueurs afin d'alterner les tours entre le bon nombre de joueur. Il y a aussi une autre méthode "setActive_Player(int newVar)" qui définit le joueur actif, soit le joueur dont c'est le tour.

GameWindow:
Cette classe a pour but d'exécuter la fenêtre graphique de choix du nombre de joueur, et donc de définir pour la suite du jeu le nombre de joueurs.
(La suite du jeu se fait en console)

Gem:
Cette classe a pour but de créer une Tile de Gem (gemme ou joyau) à la position donnée.

Hand:
Cette classe a pour but de créer la liste de carte représentant la main du joueur (5 cartes). Les cartes sont récupérées depuis le deck du joueur et sont défaussées dans le liste de défausse.
Elle possède une méthode "getHand()" pour récupérer la liste des cartes actuellement dans la main du joueur.

Menu:
Cette classe est la classe contenant la méthode main qui exécute le projet en exécutant la fenêtre de jeu (aucun affichage graphique, voir le jeu en console).

Panneau:
Cette classe avait été à l'origine créé pour l'affichage du jeu avec swing mais suite a des contrainte de temps elle n'as pas été utilisée.

Player:
Cette classe a pour but la création de joueur. Elle crée un joueur et lui attribue une couleur, un deck, une main, une défausse, une tortue, un programme de tortue à l'aide de méthodes de set et get pour chaque élément.

Program:
Cette classe a pour but de créer une liste de carte qui représenteront le programme de déplacement exécuté par la tortue via la fonction "executeProgram(int playerNumber)". Chaque carte exécute une instruction (décrite en I.)

Tile:
Cette classe a pour but de créer les cases du tableau représentant le plateau de jeu.
Elle possède une méthode "setTurtle(Turtle newVar)" qui permet de nommer la tortue avec la bonne couleur sur la bonne Tile.

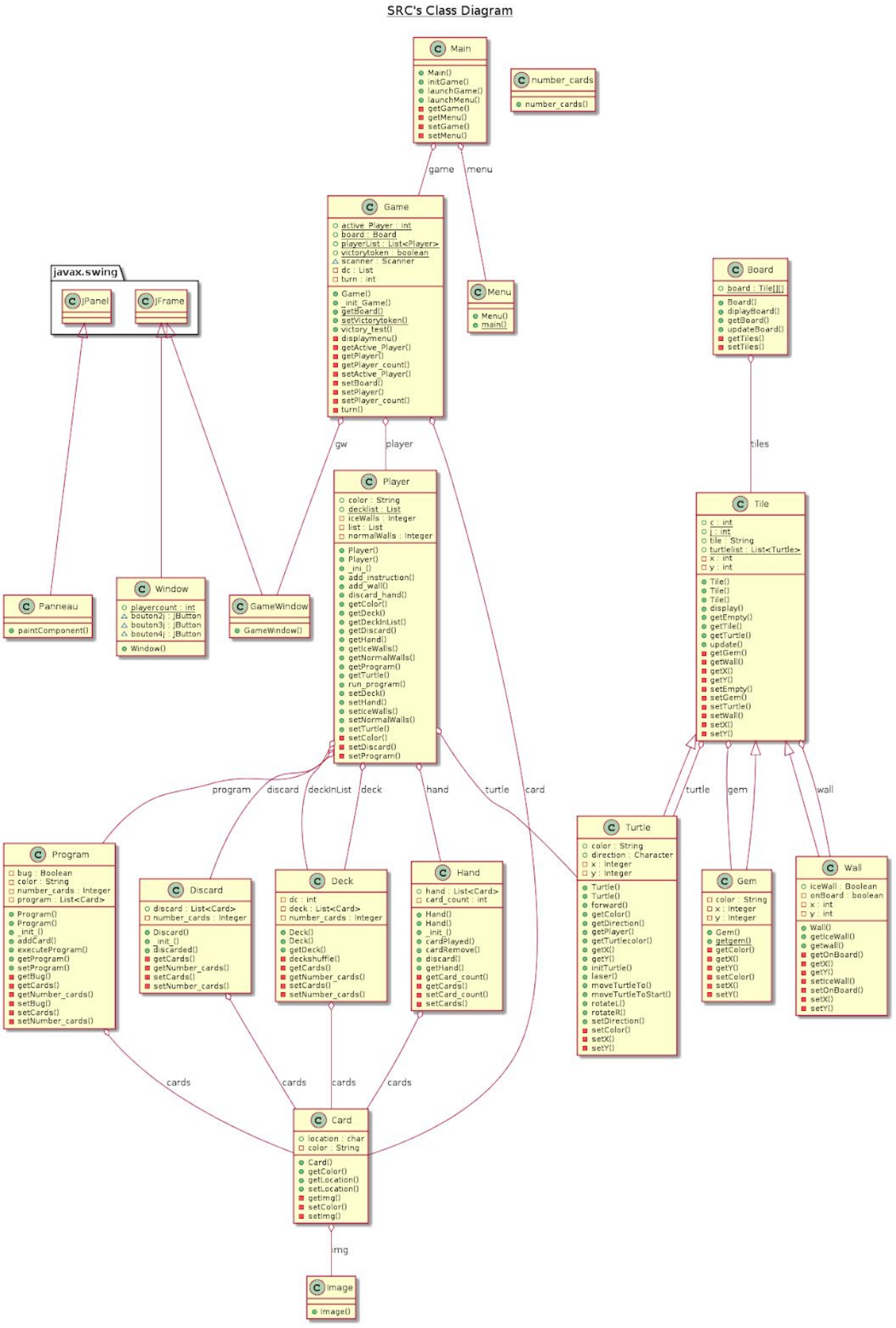
Turtle:
Cette classe a pour but de créer une tortue et de l'assigner au bon joueur avec la bonne couleur et son initialisation sur le terrain, notamment la bonne direction (sud: 's').

C’est dans cette classe que sont intégré les méthodes pour les actions des cartes sur la tortue: “rotateR()”, “rotateL()”, “laser()”, “forward()”.
Pour simplifier certaines actions dans les nombreux cas possible (nombre de joueur, statut de la Tile sur le trajet…) nous avons créé des méthodes aux noms explicite: “moveTurtleTo(Integer y, Integer x)”, “moveTurtleToStart()”.
Il existe aussi un certain nombre de méthodes de “set” et “get” pour la direction, la couleur, le position.

Wall:
Cette classe a pour but de créer les cases (Tiles) de mur (Wall). Un mur est construit à des coordonnées données et possède un booléen “iceWall” qui défini si oui ou non le mur est un mur de glace et donc est destructible par un laser.
On utilise ici des méthodes de “getWall”, “setIceWall()”.

Window:
La classe Window est “extends” de JFrame et permet la création de la fenêtre de choix du nombre de joueur et définir pour la suite du jeu le nombre de joueur avec la variable “playercount” en fonction du bouton sélectionné par l’utilisateur.

B. UML



III. Conclusion

Pour conclure, notre groupe de projet a réalisé le jeu Robot Turtles en console. Il est possible de jouer au jeu entre 2 et 4 joueurs. Nous n'avons pas réussi à faire fonctionner l'interface graphique Swing à l'exception de la fenêtre de choix de nombre de joueur, cependant l'algorithme de jeu fonctionne entièrement et des displays de texte clair en console permettent de jouer avec des input des joueurs.

Il nous a fallu acquérir des compétences pour y arriver et malgré le fait que le jeu ne soit pas entièrement fonctionnel, nous avons essayé de proposer un projet du jeu montrant aux maximum notre implication ainsi qu'un jeu qui puisse fonctionner avec un début, les mécaniques du jeu et une fin.