Sangavy SHASEEHARAN
A2

Rattrapage de Java
Jeu des six couleurs

I. Rappel du sujet

Ce dossier a pour but de décrire le jeu java, le jeu des six couleurs. Il s'agit d'un jeu de stratégie se déroulant sur un plateau découpé en cases de 6 couleurs différentes (rouge, orange, jaune, vert, bleu ou violet). Le but du jeu est de contrôler plus de cases que les adversaires à la fin de la partie.

Les joueurs (de 2 à 4) commencent la partie en contrôlant chacun une case de la grille, ces doivent être de couleurs différentes. Dans notre cas, nous avons seulement 2 joueurs. Les joueurs jouent alors chacun leur tour. A son tour, un joueur choisit une couleur différente de celle qu'il a actuellement, et non utilisée par ses adversaires.

Cependant deux conditions sont à respecter :

- Toutes les cases contrôlées par le joueur deviennent alors de la couleur choisie.
- Toutes les cases de la couleur choisie et qui touchent une case contrôlée par le joueur passent sous son contrôle.

La partie s'arrête lorsqu'il n'y a plus de case non contrôlée ou lorsqu'un joueur contrôle plus de la moitié des cases. Le joueur gagnant est celui qui contrôle le plus de cases.

II. Modélisation

Afin de réussir ce projet, j'ai décidé de faire 4 classes java qui sont la classe Colors, Main, Utils et Player. Dans le but de bien comprendre le fonctionnement de ce jeu java, je vais vous décrire ces classes une par une pour comprendre le rôle qu'elles jouent dans le bon fonctionnement de ce dernier.

La classe Main est la classe principale du jeu. C'est cette dernière qui va appeler toutes les autres classes qui vont faire marcher le jeu. Cette classe fait que la structure du jeu est alors plus propre et plus facilement compréhensible.

La classe Player est la classe qui va définir nos personnages avec leur id, name, color etc. Nous pouvons voir sur le code que nous importons 3 librairies : ArrayList,List et UUID. Ce dernier sert notamment pour générer automatiquement l'id du joueur.

La classe Colors c'est la classe qui s'occupe de toutes les fonctions qui touchent les couleurs du jeu.

La classe Utils est la classe la « plus importante » car c'est elle qui va définir les conditions de jeux, les mouvements, l'implémentation des personnages etc.

III. Choix et descriptions des algorithmes utilisés

Dans cette partie je vais vous expliquer certaines des fonctions les plus importantes et vous montrer à quoi elles servent dans le code.

Public void initGame() : cette méthode explique ce qui est définit dans la classe Player, notamment avec le nom du joueur, la couleur par défaut qui lui est attribuée etc. Ainsi nous avons donc 2 personnages dans notre ArrayList. Si on aurait voulu plus de personnages, on aurait pu les initialiser dans cette fonction. La boucle for présente dans cette méthode nous permet de créer la map avec n'importe quelles couleurs. Ainsi c'est cette boucle qui va permettre de définir les couleurs randoms sur toute la map sauf sur la 1ère case et la dernière qui sont attribuées aux joueurs 1 et 2.

Private void savePlayerStatus() : cette méthode permet de garder en mémoire les cases que chaque joueur a « gagné »/ acquis.

Private void play() : signifie que c'est la création de la méthode play protégé par le "private" et on n'attends pas de retour particulier grâce à cette méthode (void). Dans cette fonction nous utilisons deux boucles while. On utilise une boucle "while" qui va nous permettre de mettre en place la partie où on pourra voir les différentes couleurs disponibles à choisir, puis une autre boucle "while" qui nous permettra de sélectionner la couleur voulue. Les deux boucles for qui suivent permettent de parcourir tout le tableau.
Cette méthode va permettre de « comptabiliser » au cours du jeu les cases contrôlées par chaque joueur et en déduire ainsi le gagnant.

```
C'est au tour de Joueur 1 !
 | I | I | I | I | I | I | I | I | I | I | I | I | I | r | I | I |
 | I | I | I | I | I | I | I | I | I | I | I | I | v | I | j | I | I |
 | I | I | I | I | I | I | I | I | I | I | I | I | v | j | I | I | I |
 | I | I | I | I | I | I | I | I | I | I | I | I | I | v | I | I | I |
 | I | I | I | I | I | I | I | I | I | I | I | I | r | o | I | I | I |
 | I | I | I | I | I | r | I | r | r | r | i | r | I | I | I | I |
 | I | I | I | r | j | I | I | I | b | r | i | v | r | I | I | b |
 | I | I | I | r | I | I | I | v | v | o | o | i | b | r | b |
 | v | I | I | I | I | v | v | i | i | i | o | b | o | b | b |
 | i | j | r | v | v | j | b | i | i | o | b | b | b | b |
 | o | r | b | o | o | j | v | r | j | b | o | b | b |
 | o | j | v | r | r | i | v | r | b | b | b | b | b |
 | o | o | b | i | r | o | b | j | b | b | b | b | b |
Veuillez choisir une couleur parmis : r, o, j, v
v
Le jeu et fini ! Et le grand gagnant est ... Joueur 1 !!! Bravo !!!
```

Private Player nextPlayer() : cette méthode contient la boucle qui permet de changer de joueur lorsque l'on passe sur la boucle.

Private boolean endOfGame() : il s'agit de l'algorithme qui explique si c'est la fin du jeu ou non .

Private Player getWinner() : cette méthode montre comme le gagnant est définit. On part du principe qu'il n'y a qu'un seul gagnant et on compare avec : (player.controlledBlocks.size() > winner.controlledBlocks.size()).

Public void renderBlocks() : dans cette méthode nous avons plusieurs boucles. La première permet de nous donner le contrôle si le personnage ne l'a pas encore. La deuxième boucle permet de renvoyer la case contrôlée par le personnage. Et enfin la troisième boucle permet d'afficher toute la map afin de voir toutes les cases contrôlées par le joueur en question.

Private void getControlOnBlock : cette fonction permet de faire passer une case de non contrôlée à contrôlée.

Public boolean checkPlayerControlled : cette boucle permet de rafraichir le tableau des blocks contrôlés par le joueur en question.

Public String getUncontrolled() : quand le joueur est sur une case non contrôlée, cette méthode lui retourne le première lettre de la couleurs en minuscule.

Public String getControlled() : quand le joueur est sur une case contrôlée, cette méthode lui retourne la première lettre de la couleurs en majuscule.

Public static String listOptions : c'est cette fonction qui va permettre aux jeux de définir les possibilités de couleurs autour du joueur.

Public static Colors findByUnControlled : cette fonction permet d'afficher toutes les couleurs non contrôlées.

IV. Conclusion

Pour conclure ce projet, le jeu en mode console marche bien. Ce projet m'a appris à me débrouiller toute de seule avec le langage java. Cependant j'ai quand même rencontré des difficultés, la première étant la visualisation du jeu par rapport à l'énoncé. Puis la concrétisation de mes idées, les traduire dans le langage de programmation. Ainsi ayant rencontré des difficultés de différents niveaux, je n'ai pas pu concrétiser mon temps. Ceci est dont la conséquence du manque de design sur le jeu. Cependant, le bilan de ce projet me semble positif.