DARROU Cédric
PERROT Elise
RAMORINO Eva

Programmation
19 Décembre 2018

Rapport du projet :
HORDES

Nous avons décidé de faire 12 classes différentes pour ce projet en java.

Tout d'abord la classe:

- `public class Hordes`

  Cette classe reprend toutes les fonctions que nous avons besoin pour le bon déroulement du jeu, et est séparé en plusieurs fonctions que nous reverrons plus tard. C'est dans cette classe que se trouve toutes les variables nécessaires au jeu. Et c'est aussi dans cette classe que nous avons décidé de mettre le scaneur (l'entrée) que nous refermerons à la fin dans le main. C'est dans cette classe que nous allons procéder au déroulement automatique du jeu c'est à dire : les changements de tour ainsi que les changements de jour (qui va donc comprendre bien sûr l'attaque des zombies sur la ville à la fin de la journée), définir aléatoirement le nombre de zombies sur une case à chaque journée, et la consultation du journal contenant les morts de la veille. Pour terminer, nous avons la fonction `public static void game()`. Cette fonction permet de dérouler le jeu : elle indique le tour du joueur, lance la fonction menu, lance le changement de tour, etc... Cette fonction s'arrête dès qu'il y a un gagnant, qu'elle affiche.

  L'héritage

  Le but de l'héritage est de créer des classes qui sont des extensions de la classe Hordes, autrement appelés des sous-classe de la classe Hordes.java, et nous avons remplacé les variables private en protected, qui autorise ainsi les sous classes à utiliser les variables de la classe principale. Ainsi les sous-classes peuvent utiliser les variables d'Hordes.java. Le principal avantage de l'héritage est de permettre de séparer son code, et d'éviter que nous ayons plus de 2000 lignes dans le même fichier.

Il y a également:

- `class Initialization extends Hordes`

  La classe initialise la carte, les items sur chaque case de la carte (définis aléatoirement) et montre les items une fois découverts. Elle va permettre aussi de demander le nombre de joueurs et leurs pseudonymes. Cette classe contient deux fonctions showMap, pour afficher les objets cachés, la première (sans paramètres) pour toute la carte (donc les 625 cases) et la deuxième pour une seule case, elles sont en commentaires, mais nous vous les laissons à disposition.

Ensuite il y a la classe:

- `class AP extends Hordes {`

  La classe AP est aussi une extension de Hordes. Elle va nous permettre de nous faire gagner des points d'actions en reprenant des forces (en buvant de l'eau ou en mangeant par exemple).

- `class Bank extends Hordes {`
  Bank.java est aussi une extension de la classe Hordes. Elle va nous permettre d'ajouter, d'enlever et de consulter tous les objets qui sont déposés dans notre réserve (que nous avons appelé Banque) dans la ville.

- `public class City {`
  La classe City nous permet de retourner les créations de tableaux des objets en banque, des points d'action pour le chantier avec la construction des défenses et de retourner si la porte de la ville est ouverte ou non.

- `class Inside extends Hordes {`
  Inside est ainsi aussi une extension de la classe Hordes. Elle va nous dire si nous pouvons sortir ou entrer dans la ville ainsi que décider si nous pouvons oui ou non participer aux constructions des défenses.

- `class Outside extends Hordes {`
  Cette classe va nous permettre toutes les actions en dehors de la ville, elle permet de nous déplacer une fois hors de la ville. Outside permet également de fouiller dans les cases de la carte, de choisir de prendre, de déposer des items et/ou de tuer un zombie ou des zombies si nous en rencontrons.

- `public class Player {`
  La classe Player va générer un objet pour le joueur contenant son pseudonyme, ses points d'actions (ainsi que le gain et la perte), sa position, son nombre de points de vie (que nous gardons en caché), la nourriture et l'eau (s'il a mangé ou non par exemple), savoir si le joueur est en ville et avoir accès à l'inventaire du joueur (son sac dos).

- `public class Map {`
  La classe Map va générer un objet correspondant à une partie de la carte (une simple case parmi les 625) et contient les items (cachés ou non) présents et de mettre à jour le talkie sur la zone et le nombre de zombie. Elle permet aussi d'afficher les items non-cachés qui sont au sol. Elle contient également le nombre de zombies (y compris quand le nombre de zombies décroît après en avoir tué). La classe Map communique aussi aux autres joueurs le contenu d'une case de la carte par talkie-walkie.

- `class Talkie extends Hordes {`
  La classe Talkie, sous classe de la classe Hordes, va permettre aux joueurs d'accéder au contenu d'une case, si elle a été mise à jour par les autres joueurs par talkie-walkie.

- `public class Main {`
  Le Main nous permet de procéder à l'initialisation du jeu (carte, items, joueurs, zombies, et le jeu bien entendu) qui va nous permettre de le démarrer.

- `class Menu extends Hordes {`
  La classe Menu est la classe qui contient les différents menus qui va nous permettre d'exécuter les suites du jeu. Il va exécuter dans l'ordre demandé par le joueur et va appeler les fonctions des classes énoncés au dessus et demander au joueur ce qu'il veut faire.

Tous les achievements ont été effectué, sauf le 5e concernant l'interface graphique.

Nous allons maintenant décrire l'exécution du jeu.

Pour l'exécution du jeu, il suffit de faire Lancer le projet. Il va d'abord nous saluer puis il va initialiser tout ce dont nous avons besoin pour jouer, c'est-à-dire la carte, les items présents sur chaque case (ils sont définis aléatoirement), il va nous demander le nombre de joueur au cours de cette partie et leurs pseudonymes, pour enfin initialiser le nombre de zombies (aléatoirement, compris entre 0 et 7) sur chaque case. On va ensuite passer au tour 1 du jour 1 pour chaque joueur. On commence bien entendu par le joueur 1 qui, après avoir été informé de son nombre de points d'action en sa possession, aura le choix entre plusieurs propositions:

1. Accéder à son inventaire
2. Accéder à la banque
3. Prendre de l'eau
4. Accéder aux chantiers
5. Accéder à la porte
6. Sortir de la ville
7. Consulter le talkie
8. Consulter le journal

Ou encore Passer son tour (noté 0 dans le code)

C'est donc d'abord le joueur 1 qui va choisir d'exécuter l'une de ces choses. Bien sûr en choisir une implique d'autres choix (par exemple, en sortant de la ville, on peut choisir de se déplacer, puis de fouiller, de déposer/prendre un objet ou encore de se battre contre des zombies si nous en rencontrons).

Chaque joueur fait ses actions durant son tour, puis doit passer l'ordinateur au joueur suivant.

Il y a 12 tours en une journée. A la fin du Jour 1 (et donc des 12 tours pour chaque joueur), le programme affiche le nombre de zombies qui attaquent, le nombre de morts durant cette attaque avec leurs noms et passe au Jour 2.

Le programme se finit seulement quand il ne reste qu'un seul joueur qui sera le gagnant ou si tout le monde est mort.

N.B. : Au début de chaque tour de chaque joueur, la console demande de taper ok, afin de créer une pause dans le jeu pour bien montrer le changement de joueur. Il est normal que le jeu continue même si le joueur ne tape pas ok dans la console (mais on est obligé de rentrer un élément).