Qualité-Test

Axel Gendillard
Kim Belassen
Valentin Cosson

Sommaire

1 Introduction **3**

2 Diagramme de classe **3**

3 Incrément 1 **5**

4 Incrément 2 **5**

5 Incrément 3 **6**

1 Introduction

L'objectif du projet Qualité Test était de réaliser, en respectant les exigences qualité demandées, un jeu multi-joueurs. Le but du jeu est d'être le premier pirate à découvrir un trésor sur le plateau de jeu. L'ensemble du jeu sera développé en java et des tests devaient être effectués tout au long de l'implémentation du jeu.

2 Diagramme de classe

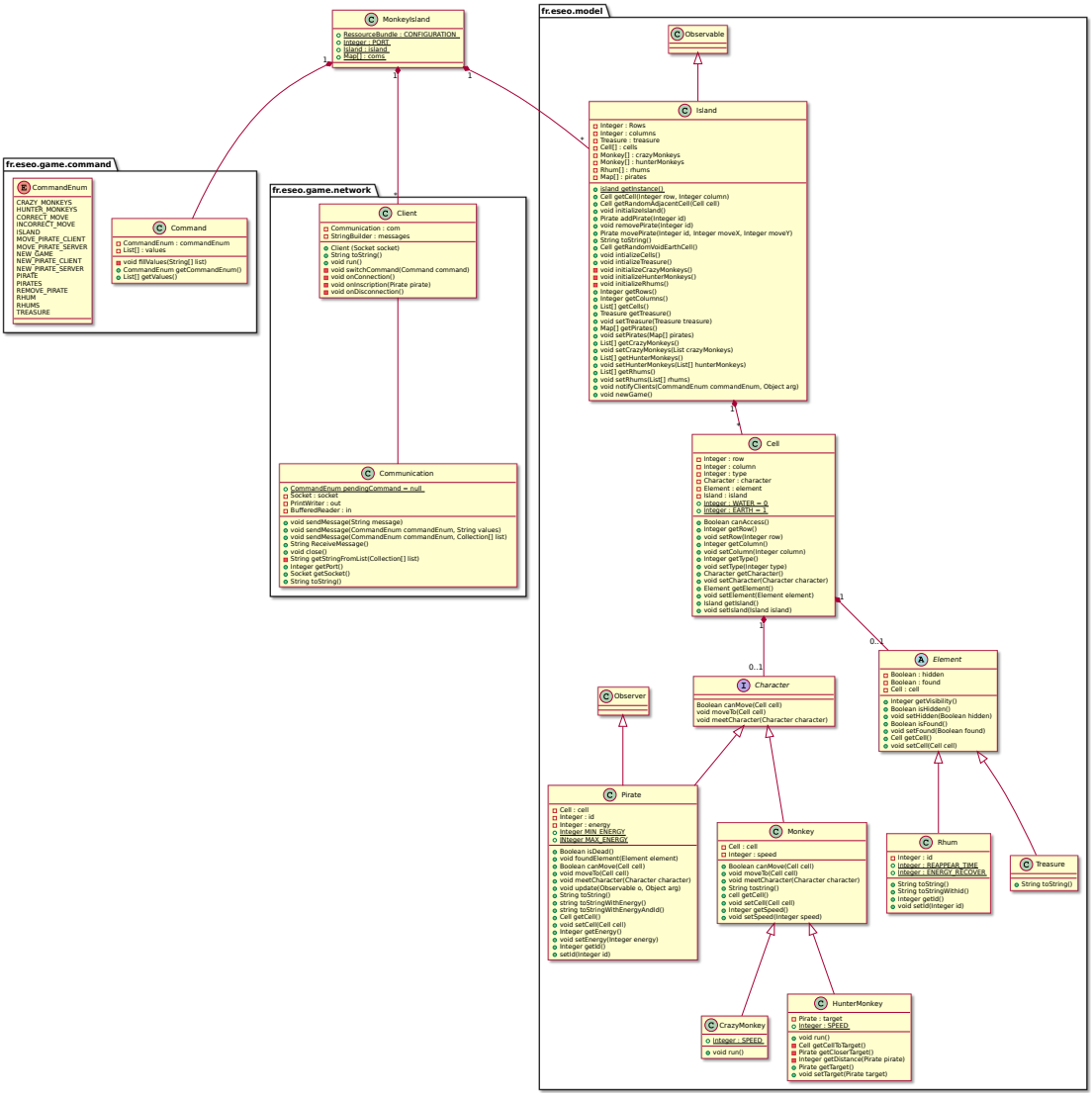
Nous avons séparé le projet en trois packages :

— Le package **commande** qui regroupe les classes qui gèrent la commande ;
— Le package **model** qui représente la logique du serveur. Ce package regroupe la gestion du jeu en général (déplacements des personnages, création des nouveaux pirates, initialisation d'une partie, etc.) ;
— Le package **network** se consacre à la connexion et la communication entre le serveur et les clients.

Nous avons implémenté deux design-pattern pour la réalisation de ce projet :

— Le **Singleton**. Le singleton est l'île. C'est à dire qu'il existe une unique instance de l'objet Island qui est partagée auprès de tous les clients ;
— L'**Observer**. Si il y a un changement sur l'île, tous les pirates sont notifiés.

diagramme UML du jeu monkey Island



3 Incrément 1

Le premier incrément devait rassembler les fonctionnalités suivantes :
— Le déplacement des singes erratiques,
— Le gestion des pirates (déplacement et décès),
— La gestion du trésor,
— La communication client-serveur.

Nous avons testé avec Junit toutes les méthodes publiques des classes *Monkey*, *Pirate*, *Treasure* et *Island* et *Cell*.

Afin de déplacer les singes, nous avons utilisé un *thread* pour chaque singe. La méthode *run* est appelée toute les secondes (vitesse du singe) grâce à la classe Java : *ScheduledExecutorService* et la méthode *scheduleAtFixedRate*.

```
ScheduledExecutorService execCrazy = Executors.
    newSingleThreadScheduledExecutor();
    for (Monkey monkey : island.getCrazyMonkeys()) {
        execCrazy.scheduleAtFixedRate(monkey,
            0, monkey.getSpeed(), TimeUnit.SECONDS);
    }
```

Nous avons testé le déplacement des singes erratiques dans toutes les directions (équiprobabilité), le déplacement du pirate dans toutes les directions, la rencontre des différents personnages et des éléments. Enfin, la communication a été testée avec JMeter.

4 Incrément 2

Le deuxième incrément devait rassembler les fonctionnalités suivantes :

— La gestion du fichier de configuration,
— La gestion de l'énergie de chaque pirate,
— La gestion des bouteilles de rhum,
— La gestion des parties.

La gestion du fichier de configuration a été effectuée grâce à la classe Java : *RessourceBundle*. Cette classe permet le chargement d'un fichier au format *properties* (clé=valeur). Cet objet est stocké dans la classe *MonkeyIsland*.

```
public static final RessourceBundle CONFIGURATION =
    RessourceBundle.getBundle("Configuration");
```

5 Incrément 3

Le troisième incrément devait rassembler les fonctionnalités suivantes :

— La gestion des singes chasseurs.

Pour les singes chasseurs, nous avons implémenté un algorithme qui détermine la cible(un pirate) pour chaque singe chasseur. Pour définir la cible la plus proche, nous calculons la distance entre le singe chasseur et les pirates présents sur le plateau. La cible d'un chasseur est donc définie comme étant le pirate qui se trouve le plus proche (la distance la plus petite).

Une fois la cible choisie, le singe se déplace dans la direction du pirate en incrémentant ou décrémentant la position x ou y du singe.