Réalisation d'application

Rapport itération I

Jeu D'aventure

Zuul-bad

Auteurs
Groupe 5
LOKO Loïc
GHOUIBI Ghassen
BOUCHICHA Abdelrahim

Table des matières

I Présentation

Dans le cadre du cours "Réalisation d'application", nous avons eu la tâche de réaliser l'itération 1 du jeu d'aventure intitulé "Zuul".
Zuul-bad est une version de world-of-zuuul c'est un jeu d'aventure qui était développer en 1970 par Will Crowther et étendu par Don Woods.Le jeu original est connu sous le nom Colossal Cave Adventure.
Le principe du jeu c'est trouvé un chemin dans un système compliqué de cave autrement dit une labyrinthe avec des portes et le joueur doit essayer de trouver des mots secrets et des trésors cachés et autres.Toute en visant de recolter le maximum de point possible. Dans ce rapport on se base sur Zuul-bad c'est une implémentation avec des mauvais design de classe.
Le but de cette itération est de découvrir certains facteurs qui influencent la conception d'une classe.
Dans cette itération, en répondant aux différentes questions, nous avons découvert des principes importants dans la conception d'une classe. Ce rapport contient donc les différentes réponses aux questions.

II Exercice 7.1

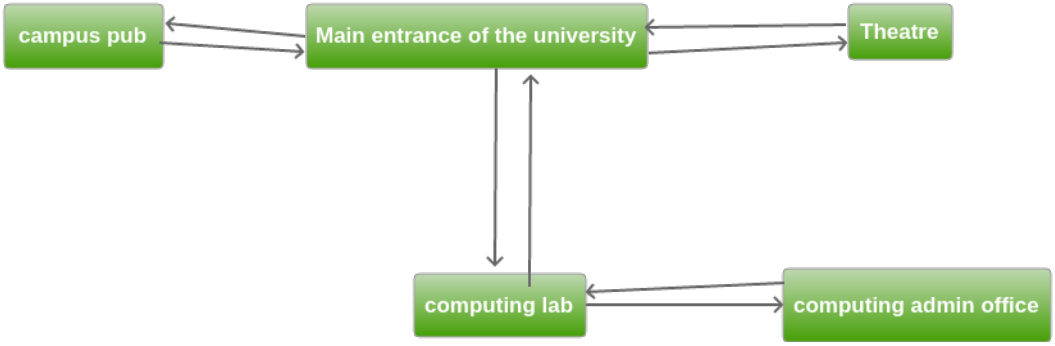
7.1.1 Que fait cette application ?
Cette application est implémenté en mode texuelle permet le joueur de se balader dans des pièces a traves l'interaction en ligne de commande (ou autres).

7.1.2 Quelles commandes le jeu accepte-t-il ?
Le jeu accepete les commandes suivantes :
go quit help

7.1.3 Que fait chaque commande ?
go : permet le joueur de se déplacer dans quatre direction *north,south,east,west* et entrée dans une chambre s'il existe.
help : pour le moment cette commande permet de localisé vagement l'endroit du joueur. quit : permet de quitter le jeu.

7.1.4 Combien de pièces y a-t-il dans le scénario ?
il existe 5 chambres : *outside, theatre, pub, lab, office*

7.1.5 Dessinez une carte des pièces existantes.
Plan du jeu inventé :



III Exercice 7.2

Rôle des classes :
CommandWord : Cette classe permet tout simplement de tester si une chaine est equivalente au tableau de commande qu'elle contient à traves la méthode isCommand qui renvoie true dans le cas les chaines sont equivalentes sinon false.

Parser : Cette classe permet de récuperer la chaine à partir de la ligne de commande en utilisant la bibliothèque Scanner à traves la méthode getCommand On récuperer le premier mot et le deuxième et ignore le reste.

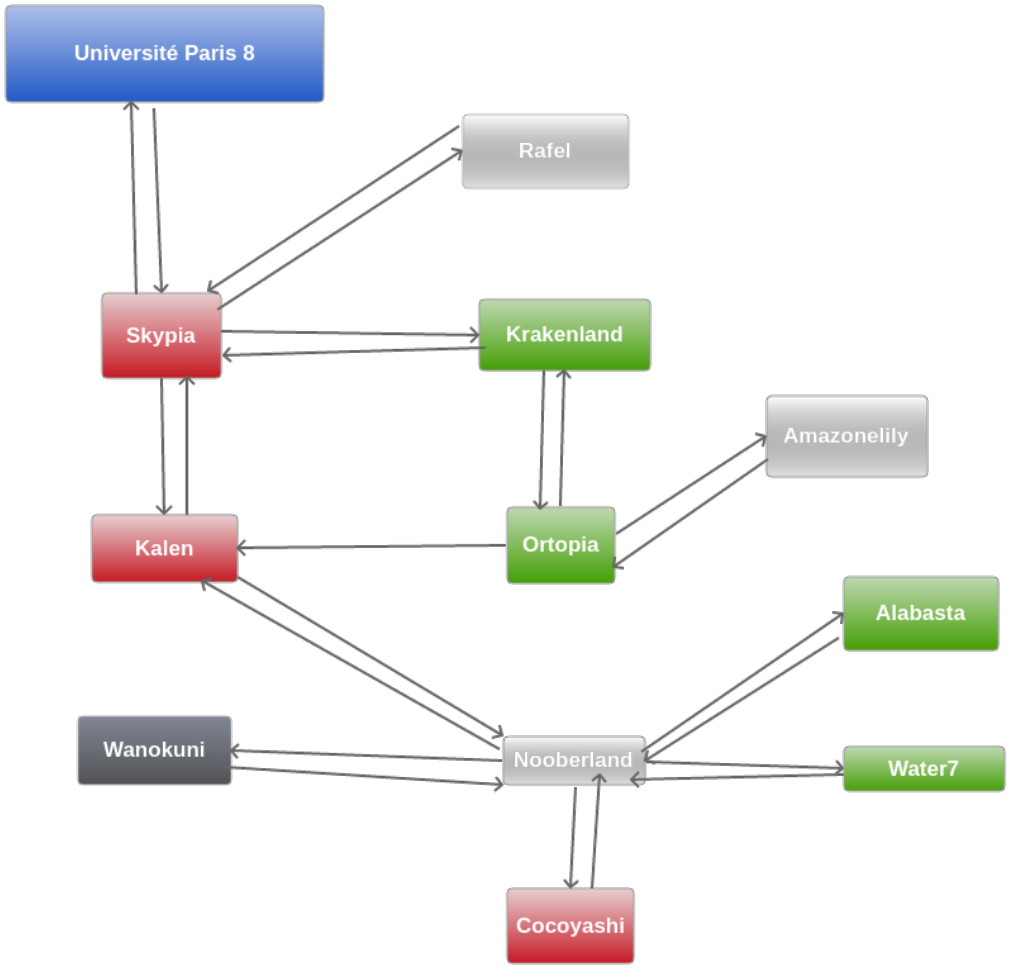
Command : Cette classe permet la vérification individuel des commandes saisie de la part de l'utilisateur décomposé en deux parties au par avant avec le parser.

Room :Cette classe permet la création des salles avec une petite description on constate aussi la méthode setExits qui permet de mettre des sorties de cette salle.

Game :C'est la classe main du jeu permet d'excuter le jeu en boucle tant que on quit pas le jeu toute en lisant les commandes entrées par l'utilisateur et faire les déplacements dans les salles selon les envies du joueur.

IV Exercice 7.3

Plan du jeu inventé :
Vous êtes le chef d'équipage d'un bateau de pirates, vous naviguez sur les mers afin de visiter différentes îles, se ravitailler en vivres ou équipements, combattre d'autres équipages et la marine afin de s'approprier et trouver les plus grands trésors dont le One Piece.



V Exercice 7.4

Voir code.

VI Exercice 7.5

Voir code.

VII Exercice 7.6

Voir code.

VIII Exercice 7.7

Voir code.

IX Exercice 7.8

Voir code.

X Exercice 7.9

Recherchez la méthode keySet dans la documentation de HashMap. Qu'est ce que ça fait ?
La méthode keySet() est utilisée pour obtenir une vue d'un Map,Un Map est un ensemble de clé valeur,et là méthode keySet permet de voir nos clés.Les modifications apportées au Map sont donc reflétées dans l'ensemble, et inversement.

XI Exercice 7.10

Expliquez, en détail et par écrit, comment la méthode getExitString indiqué dans le code 7.7 fonctionne.

```
String returnString = "Exits:";
```

Cette ligne on déclare une chaine de caractère simple.

```
Set<String> keys = exits.keySet();
```

Ici on déclare notre Set on obtient toute les clés grâce a keySet() du HashMap exits

```
for(String exit : keys)
        returnString += " " + exit;
```

Ici on boucle sur ces clés et on ajoute nos clés a notre chaine de départ returnString

```
return returnString;
```

Renvoie la chaine avec les clés de exits .

XII Exercice 7.11

Voir code.

XIII Exercice 7.12

Draw an object diagram with all objects in your game, the way they are just after starting the game

XIV Exercice 7.13

How does the object diagram change when you execute a go command ?

XV Exercice 7.14

Voir code.

XVI Exercice 7.15

Voir code.

XVII Exercice 7.16

Voir code.

XVIII Exercice 7.17

Si vous ajoutez maintenant une nouvelle commande, avez-vous encore besoin de changer la classe de jeu ? Pourquoi ?
Oui ,dans la méthode processCommand qui permettera de tester si la commande saisie par l'utilisateur existe ou pas comme ceci :

```
if(commandWord.equals("newcommand"))
                    dosomething();
```

par contre on est plus obligé de l'écrire dans printHelp puisqu'elle s'appelle automatiquement a partir du parser :

```
public void showCommands(){
                    commands.getCommandList();
    }
```