

RÉALISATION D'APPLICATION

RAPPORT ITÉRATION II

Jeu D'aventure

Zuul-bad

Auteurs
Groupe 5
LOKO Loïc
GHOUIBI Ghassen
BOUCHICHA Abdelrahim

Table des matières

I Présentation	2
II Itération I	2
II.1. Exercice 7.1	2
II.2. Exercice 7.2	4
II.3. Exercice 7.3	4
II.4. Exercice 7.4	5
II.5. Exercice 7.5	5
II.6. Exercice 7.6	6
II.7. Exercice 7.7	6
II.8. Exercice 7.8	6
II.9. Exercice 7.9	6
II.10. Exercice 7.10	6
II.11. Exercice 7.11	7
II.12. Exercice 7.12	7
II.13. Exercice 7.13	7
II.14. Exercice 7.14	7
II.15. Exercice 7.15	7
II.16. Exercice 7.16	7
II.17. Exercice 7.17	7
III Itération II	8
III.1. Exercice 7.18	8
III.2. 7.18.1	8
III.3. 7.18.2	8
III.4. 7.18.3	8
III.5. 7.18.4	8
III.6. 7.18.5	8
III.7. 7.18.6	9
III.8. 7.18.7	9
III.9. 7.18.8	9
III.10. 7.19	9

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-zuul c'est un jeu d'aventure qui était développé 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. Tout en visant de récolter 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 Itération I

II.1 Exercice 7.1

7.1.1 Que fait cette application ?

Cette application est implémenté en mode textuelle permet le joueur de se balader dans des pièces à travers l'interaction en ligne de commande (ou autres).

7.1.2 Quelles commandes le jeu accepte-t-il ?

Le jeu accepte les commandes suivantes :

go quit help

7.1.3 Que fait chaque commande ?

go : permet le joueur de se déplacer dans quatre directions *north, south, east, west* et entrée dans une chambre s'il existe.

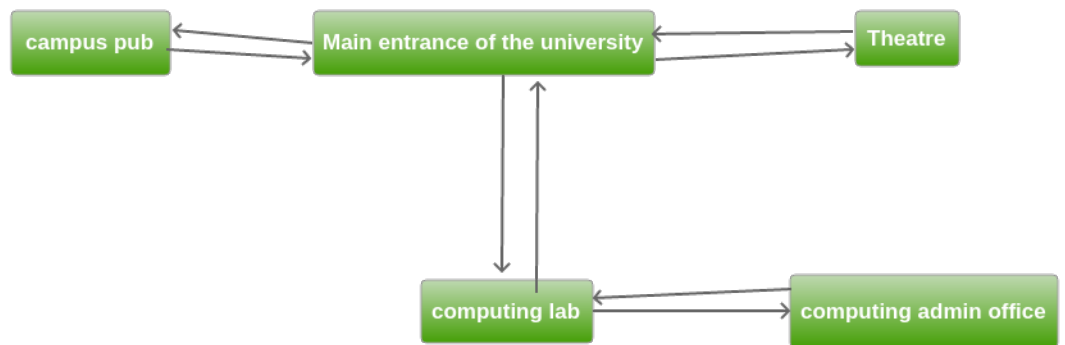
help : pour le moment cette commande permet de localiser vaguement l'endroit du joueur. *quit* : permet de quitter le jeu.

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 :



II.2 Exercice 7.2

Rôle des classes :

CommandWord : Cette classe permet tout simplement de tester si une chaîne est équivalente au tableau de commande qu'elle contient à travers la méthode `isCommand` qui renvoie `true` dans le cas les chaînes sont équivalentes sinon `false`.

Parser : Cette classe permet de récupérer la chaîne à partir de la ligne de commande en utilisant la bibliothèque `Scanner` à travers la méthode `getCommand` On récupère le premier mot et le deuxième et ignore le reste.

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

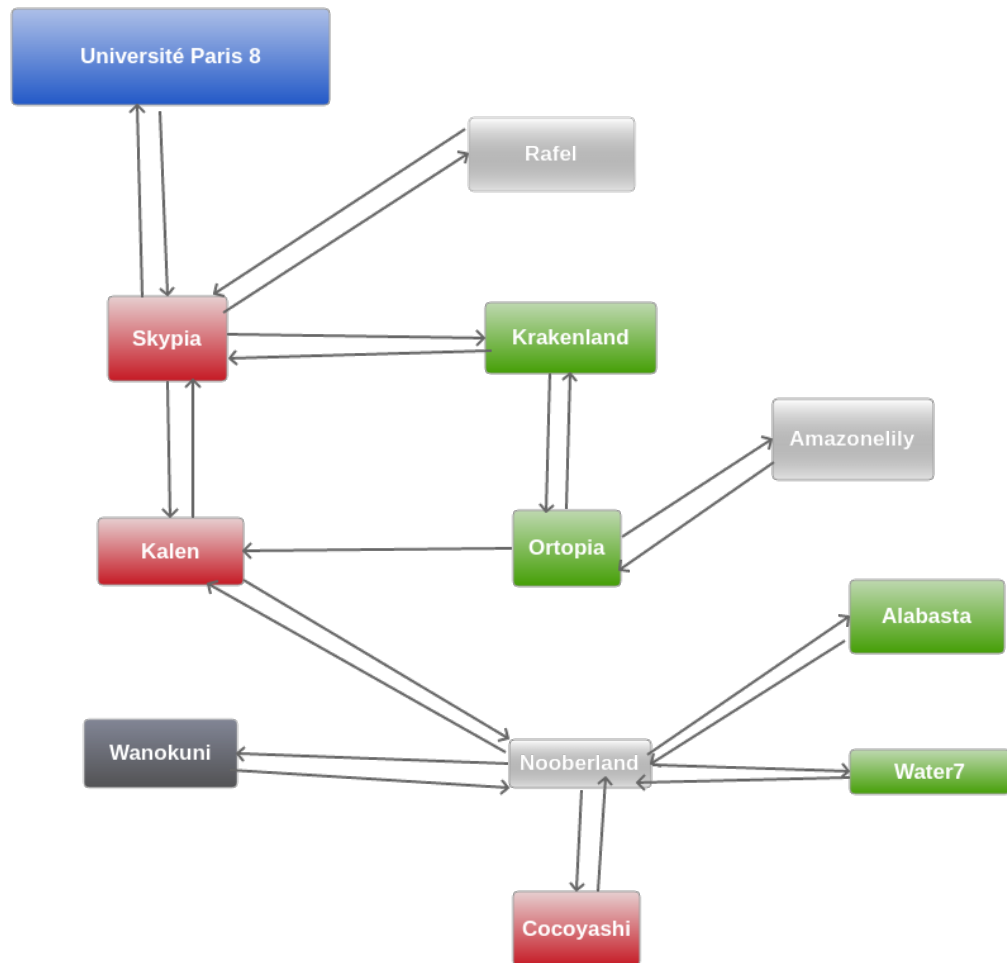
Room : Cette classe permet la création des salles avec une petite description on constate aussi la méthode `setExits` qui permet de mettre des sorties de cette salle.

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

II.3 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`.



II.4 Exercice 7.4

Voir code.

II.5 Exercice 7.5

Voir code.

II.6 Exercice 7.6

Voir code.

II.7 Exercice 7.7

Voir code.

II.8 Exercice 7.8

Voir code.

II.9 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 la méthode `keySet` permet de voir nos clés. Les modifications apportées au Map sont donc reflétées dans l'ensemble, et inversement.

II.10 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 chaîne de caractère simple.

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

Ici on déclare notre Set on obtient toutes les clés grâce à `keySet()` du `HashMap` `exits`

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

Ici on boucle sur ces clés et on ajoute nos clés à notre chaîne de départ `returnString`

```
return returnString;
```

Renvoie la chaîne avec les clés de `exits`.

II.11 Exercice 7.11

Voir code.

II.12 Exercice 7.12

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

II.13 Exercice 7.13

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

II.14 Exercice 7.14

Voir code.

II.15 Exercice 7.15

Voir code.

II.16 Exercice 7.16

Voir code.

II.17 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 permettra 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();  
}
```


III Itération II

III.1 Exercice 7.18

Voir code.

III.2 Exercice 7.18.1

Comparer son projet à celui-ci (gestion des sorties et des descriptions du projet zuul-better.

La version actuelle du projet est compatible avec zuul-better. Il y a aucun warning.

III.3 Exercice 7.18.2

StringBuilder

```
public StringBuilder getExitString(){
    StringBuilder returnStringBuilder=new StringBuilder(" Exits: ");
    Set<String> keys = exits.keySet();
    for(String exit : keys)
        returnStringBuilder.append(exit+" ");
    return returnStringBuilder;
}
```

Il existe aussi la méthode insert qui prend en paramètre la position et la chaîne à insérer.

Aussi delete qui permet de supprimer en revanche il prend deux arguments qui seront la position de départ et d'arrivée de la suppression.

III.4 Exercice 7.18.3

Voir dossier images.

III.5 Exercice 7.18.4

Le jeu est intitulé One Piece Treasure Cruise

III.6 Exercice 7.18.5

Voir code.

III.7 Exercice 7.18.6

Voir code.

III.8 Exercice 7.18.7

III.9 Exercice 7.18.8

III.10 Exercice 7.19