

Searching for the slime breeder

<https://perso.esiee.fr/~victoirb/zuul/index.html>

(IPO 2020/2021 G12)

I.A) Auteur(s)

Brendan VICTOIRE groupe 12

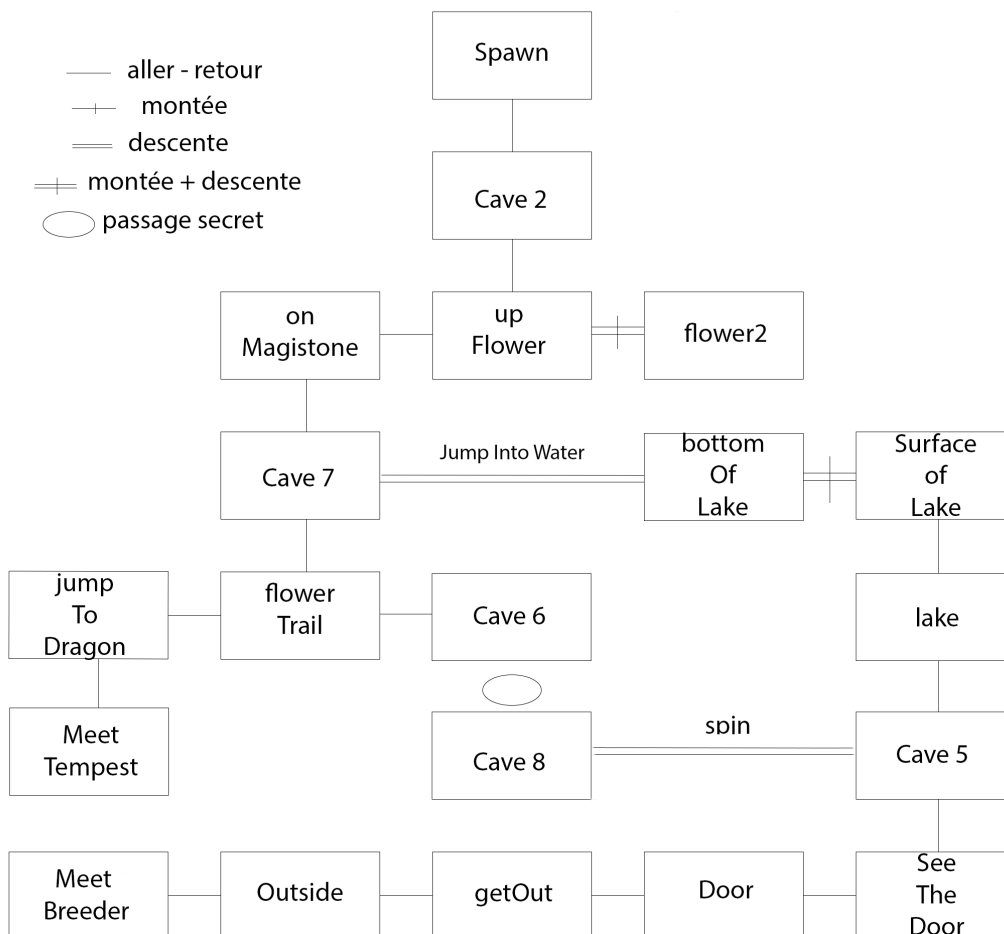
I.B) Thème

Un slime à la recherche de son éleveuse.

I.C) Résumé du scénario

Démarrage dans une grotte et essayer de trouver la sortie pour retrouver l'éleveuse.

I.D) Plan



I.E) Scénario détaillé

Vous êtes un slime perdu dans une grotte à la recherche de son éleveuse. Vous devez rencontrer Veldora pour accéder à la porte de sortie.

I.F) Détail des lieux, items, personnages

Lieux : - la grotte et ses différentes salles, l'extérieur de la grotte

Items : Magistone, une fleur, l'âme du dragon, une pierre

Personnages : Rimuru (le slime/le joueur), Veldora (le dragon), Beatrix LeBeau (l'éleveuse) et enfin Rock, Pink, Tabby et Phosphor (des slimes qui accompagne Beatrix)

I.G) Situations gagnantes et perdantes

Situation gagnante : retrouver Beatrix;

Situations perdantes : - ne pas prendre l'âme de Veldora ou la manger ;

- ne pas rencontrer Veldora;
- prendre trop de temps / trop de commandes ;
- sauter dans le lac sans voir absorbé l'âme de Veldora

I.H) Éventuellement énigmes, mini-jeux, combats, etc.

Dans la cave6 avoir l'âme de Veldora pour découvrir une porte cachée.

I.I) Commentaires (ce qui manque, reste à faire, ...)

II. Réponses aux exercices (à partir de l'exercice 7.5 inclus)

7.5 Remplacement des parties de code qui étaient identiques dans `goRoom()` et `printWelcome()` par l'appel d'une procédure `printLocationInfo()` créée pendant l'exercice.

7.6 Création d'un accesseur pour les sorties qui est ensuite appelé dans `goRoom()` et `printLocationInfo()`.

7.7 Création d'un accesseur pour récupérer les sorties de la salle actuelle sous forme de string ce qui amène une modification de `printLocationInfo()`.

7.8 Utilisation de `HashMap` pour créer une fonction pour modifier les sorties des salles + création des sorties des salles.

7.8.1 Rajout des directions "up" et "down".

7.9 Création d'une variable mémoire des directions dans `getExitString()` pour pouvoir afficher les sorties disponibles.

7.10 On a créé une variable de type `Set` qui contient l'ensemble des directions puis grâce à une boucle `for Each` on compose le `String` avec les sorties possible.

7.10.1 Ajustement des commentaires.

7.10.2 Génération du Javadoc.

7.11 Ajout d'une méthode `getLongDescription()` pour avoir une description détaillée et modification de `printLocationInfo()` grâce à cette méthode.

7.14 Ajout d'une méthode `look()` qui affiche la description longue de la pièce actuelle si la commande n'a pas d'instruction.

7.15 Ajout d'une méthode `eat()` qui affiche le fait d'avoir mangé si la commande a une instruction.

7.16 Ajout de méthodes dans les classes `CommandWords` et `Parser` pour pouvoir afficher toutes les commandes dans le message d'aide la classe `Game`.

7.17 La classe `Game` n'a plus besoin d'être modifié vu que l'on a créé des méthodes dans l'exercice d'avant qui affiche les commande de l'attribut de la classe `CommandWords`.

7.18 modification des procédures `showCommands()` et `showCommandList()` par des méthodes renvoyant des `String`.

7.18.1 Rien à modifier.

7.18.2 Remplacement des `String` par des `StringBuilder` dans `Room.getExitString()`

7.18.3 Photo déjà cherché et trouvé dès le début du projet.

7.18.4 Titre déjà choisi dès le début du projet.

7.18.5 Création d'une `HashMap` pour contenir l'ensemble des pièces.

7.18.6 Modification de classes pour pouvoir rajouter des images pour chaque salle et changer l'interface graphique du jeu.

7.18.8 Ajout de 13 boutons dans la classe `UserInterface` avec modification des procédures `actionPerformed()` - pour que les boutons exécutent la commande qui leur est associée - et de `createGUI()` - pour afficher le bouton et le rendre utilisable.

7.19.2 Ajout d'image pour le jeu et affichage de celle-ci.

7.20 Ajout d'une classe `item` pour créer des items avec leur nom, description et poids, ainsi que des accesseurs pour chacun des attributs et une fonction `toString()` pour potentiellement afficher l'item

7.21 Ajout d'une méthode `getItemString()` dans `Room` pour pouvoir afficher le nom de tous les potentiels items de la room actuelle avec la méthode

Ajout d'une méthode `setItem()` pour pouvoir ajouter une item a l'ensemble des items de la room

`getLongDescription()` et ajout d'une méthode `hasItem()` dans `Room` pour pouvoir ne rien afficher avec `getItemString()` si la room actuelle n'a pas d'item.

7.21.1 Modification de `look()` pour que si l' instruction de la commande correspond à un item, l'item `toString()` soit affiché.

7.22 Déjà fait durant l'exercice 7.21 (juste renommage de `setItem()` en `addItem()`).

7.22.1 - `LinkedList` n'est pas pratique, car les objets sont rangés et donc si un objet est supprimé de la liste alors tous ceux qui suivent verront leur index changer ce qui n'est pas très pratique. De plus s'il y a beaucoup d'objets le temps de réponse augmente proportionnellement à l'index de l'objet demandé. Et enfin on ne peut vérifier que la référence de l'objet et non l'existence exacte de celui-ci

- `ArrayList` se redimensionne à chaque fois que l'on ajoute un objet si la taille prévue de base n'est pas suffisante ce qui entraîne des ralentissements.
- `HashTable` renvoie un entier, ce n'est pas ce que nous voulons
- `HashSet` obligation de créer un `Iterator` pour parcourir le `Set`

Ce qui explique le choix de `HashMap` qui comble les problèmes listés ci-dessus.

7.22.2 Ajout d'items à des rooms.

7.23 Ajout d'une procédure `goBack()` pour retourner à la salle précédente.

7.24 Fait dans l'exercice 7.26 grâce à `stack`.

7.25 Fait dans l'exercice 7.26 grâce à `stack`.

7.26 Modification de l'attribut correspondant à la salle précédente pour qu'il corresponde à l'ensemble des salles précédentes et par conséquent de `goBack()` pour prendre en compte le `stack`.

7.26.1 Génération du Javadoc.

7.28.1 Création d'une procédure `test()` qui entraîne la modification de `interpretCommand()` et de l'attribut de `CommandWords` et qui permettra de vérifier qu'après les modifications du code dans les futurs exercices les commandes fonctionnent toujours correctement. Ces tests se font grâce à l'exécution d'un d'un fichier `.txt` dans lequel nous avons écrit au préalable les commandes à tester.

7.28.2 Création de fichier `.txt` pour tester les commandes et les salles.

7.29 Création d'une classe `Player` dans laquelle nous avons déplacé `aCreateRoom` et `aPreviousRooms` et ajouté des accesseurs et modificateurs pour ceux-ci. Mais aussi dans `Player` nous avons déplacé un bout des procédures qui permettent d'exécuter une commande.

7.30 Création de procédures `take()`, `drop()` et de procédures permettant le fonctionnement de celles-ci et d'éviter le doublon de code dans `Player`.

7.31 Déjà fait précédemment grâce à `hashMap`.

7.31.1 Création d'une classe `ItemList` pour gérer les items de `room` et `player`. Ceci implique la suppression des méthodes dans `room` et `player` qui s'appelle sur un item pour les réécrire dans `ItemList`.

7.32 Création d'un attribut `MAX_WEIGHT` pour limiter la charge que peut transporter le personnage et création par conséquent d'une méthode pour savoir connaître le poids de la charge transporter par le personnage.

7.33 Création d'une commande `items` qui affiche la liste des items transportés par le joueur.

7.34 Modification de la méthode `eat()` pour pouvoir augmenter le poids de la charge que peut transporter le personnage après avoir mangé `"stone"`.

7.34.1 Modification des fichiers de test pour prendre en compte les nouvelles commandes et possibilités.

7.34.2 Génération du Javadoc

7.35 Création d'une enum `CommandWord` pour implémenter la possibilité de jouer dans différentes langues. Et modification par conséquent de `InterpretCommand()`, du `Parser`, de `CommandWords` pour inclure les `Command` sous forme d'enum.

7.35.1 Modification de la suite de `if else{} dans interpretCommand()` par un `switch{}`.

7.37 Le jeu reste en anglais donc rien à modifier.

7.38 Le jeu reste en anglais donc rien à modifier.

7.40 Programmation de la `Command look`.

7.41 Changement de `help`.

7.41.1 Incorporation de `zuul-with-enums-v2`.

7.41.2 Génération du Javadoc.

7.42 Ajout dans `Player` d'un attribut `aCommandEntered` pour compter le nombre de commandes effectuées par le joueur et le faire quitter le jeu de force si la limite a été dépassée.

7.42.1 Limitation du jeu a une durée de 30min grâce un calcul de temps grâce a `System.currentTimeMillis()`.

7.42.2 Pour le moment, je me contente de l'interface graphique actuelle (en ayant rajouté quelques boutons).

7.43 Déjà pris en compte depuis le début du jeu donc pas de modification à faire.

7.44 Ajout d'une classe `Beamer` qui est une extension de `Item` et qui permet de mémoriser (`memorize()`) une room pour s'y téléporter (`teleport()`) par la suite mais à usage unique. Implémentation de boutons correspondant à ces deux commandes. Et enfin mis à jour des fichiers test pour tester le bon fonctionnement de `Beamer`.

7.45 Création d'une classe `Door` pour permettre de bloquer des passages et implémentation de `createRooms()` et de `goRoom()` grâce cette classe. Et mis à jour des fichiers test pour le bon fonctionnement des tests.

7.45.1 Déjà fait dans les exercices précédents.

7.45.2 Génération du Javadoc.

7.46 Création d'une procédure utilisant les classes importés `Random` et `ArrayList` pour que lorsqu'on arrive dans la Room "somewhere" si l'on essaye d'en sortir on soit téléporter dans une Room aléatoire du jeu.

7.46.1 Création d'une Command alea pour rendre la `transportRoom` non-aléatoire

7.46.2 Déjà fait lors de la création de la classe `Beamer` en la faisant hériter de `Item`.

7.46.3 Commentaires Javadoc déjà fait.

7.46.4 Génération du Javadoc.

7.47.1 Implémentation de package.

7.47.2 Génération du Javadoc.

7.48 Création d'une Classe `Character` pour mettre un PNJ dans les Room.

7.49 Création de 4 `MovingCharacter` qui suivront le joueur une fois qu'ils ont été rencontré.

7.49.2 Éléments incorporés pendant la création du jeu.

7.49.3 Génération du Javadoc.

7.50 La signature de la méthode qui calcule la valeur maximale de deux valeurs entières est : `Math.max(int a, int b)`.

7.51 Ces méthodes sont `static` on ne veux pas à avoir besoin d'un objet `Math` pour utiliser ces méthodes, car ça n'as pas de sens. Ils n'est possible de les écrire sous forme de méthodes d'instance que si la méthode est complètement réécrite par le programmeur.

```
7.52 public class test
{
    public static long count()
    {
        long vTime = System.currentTimeMillis();
        for ( int i = 1; i<=100; i++){
            vTime -= System.currentTimeMillis();
        }
        return vTime;
    }
} // test
```

7.53 Déjà créé depuis presque le début de projet pour pouvoir utiliser VS Code

7.54 Exécuté avec succès grâce à la commande à la racine du jeu : `java Game`

7.54.1 Compilé avec succès grâce à la commande à la racine du jeu : `javac Game.java -encoding utf8`. Pour modifier le code, il suffit d'ouvrir dans un éditeur de texte le dossier `.java` de la class qu'on veut modifier

7.57 il faut rajouter un attribut (`private static aInstanceCreated = 0;`) et le faire augmenter de 1 a chaque fois qu'une instance a été créé en rajoutant `this.aInstanceCreated ++;` dans le.s constructeur.s.

7.58 Création du fichier `.jar` et test de la commande `java -jar`.

7.58.1 Command fonctionnel sur Windows pas pu tester sous Linux.

7.58.2 Jeu téléchargeable sur la page perso.

7.58.3 Pas d'ajout sur l'interface Homme-machine.

7.60.2 Pas d'ajout sur l'interface Homme-machine.

7.60.3 Commentaires Javadoc déjà fait.

7.60.4 Génération du Javadoc.

7.63 Fichiers test fonctionnel.

7.63.1 Compilation fonctionnelle.

7.63.2 Scénarios fonctionnels.

7.63.3 Éléments déjà incorporés.

7.63.4 Génération du Javadoc et mis à jour du site web.

III. Mode d'emploi (si nécessaire, instructions d'installation ou pour démarrer le jeu)

IV. Déclaration obligatoire anti-plagiat (préciser toutes les parties de code que vous n'avez pas écrites vous-même et citez la source, sauf les fichiers zuul-*.jar qui sont fournis évidemment)

Toutes les images sont tirées des deux premiers épisodes de *Tensei shitara Slime Datta Ken* sauf la rencontre avec l'élèveuse tirée du jeu *Slime rancher* + le slime de *Tensei shitara Slime Datta Ken*.