L’Occultance (rapport)

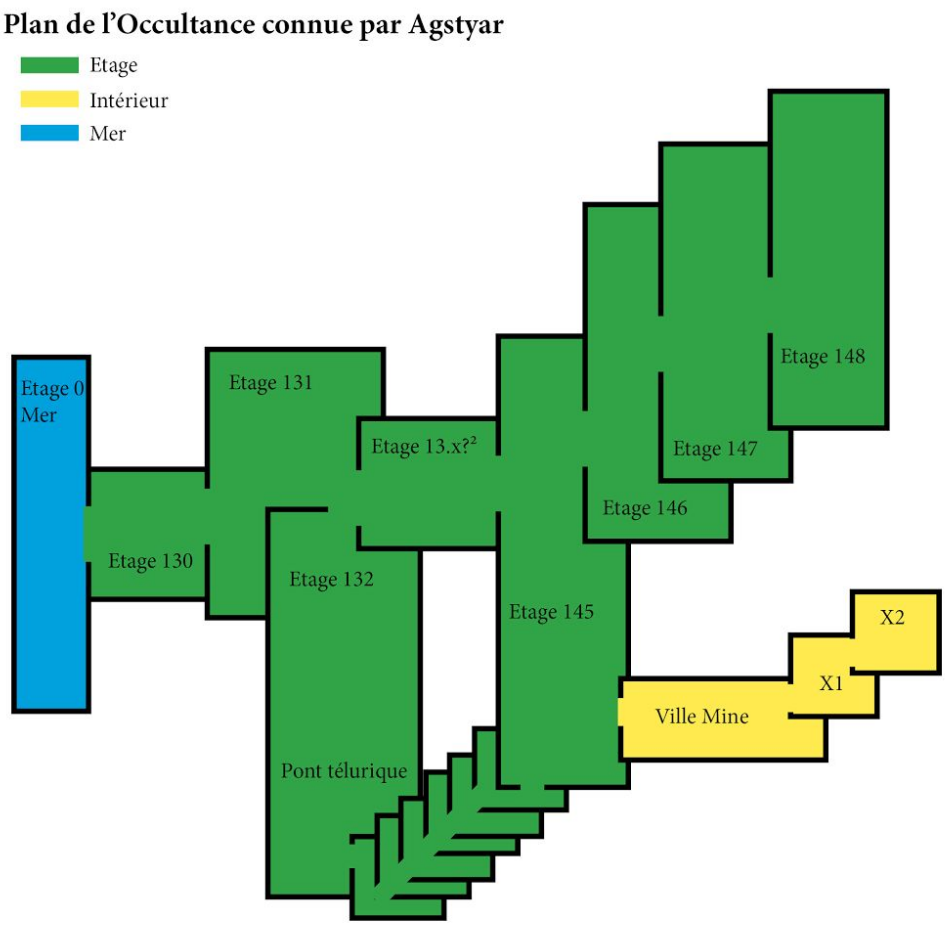
(A3P(AL) 2015/2016 G8)

I.A) **Auteurs**: Mr Jules Neghnagh--Chenavas.

I.B) **Phrase thème:** Un homme doit parler à un dinausaure.

I.C) **Résumé du scénario:** Un jeune homme prénommé Agastya Astar Cherche à atteindre le sommet de son monde. Son parcours sera semé d'embûches. Et il parlera à un dinausaure.

I.D) **Plan:**



I.E) **Résumé:** Agstya Astar est un jeune homme vivant dans un étage paysan du 83ème étage. Oppressé par ce monde qu’il ne peut pas quitter, il a rejoins une expédition ayant pour but d’arriver au sommet de ce mur. Il va donc participer, lui et 40 autres aventuriers, à cette épopée. Tout commence lorsqu’il arrive au 134ème étage (dernier étage connu). Une maladie due à un problème de conservation des denrées va entraîner la mort d’une grande partie du groupe. Il va donc devoir se débrouiller pour trouver un moyen d’atteindre un étage connu avant de nourrir de faim. Et parler à un dinausaure.

I.F) **Détail des lieux:** Tout se déroule dans un monde constitué d'une falaise immense, si grande que personne n'est jamais arrivé au sommet. Cette falaise est néanmoins quelque peux particulière; en effet, tous les 800m environ, il existe un renfoncement d'environ 60m horizontal. Ce monde est donc nivelé. deux problèmes se posent donc: le déplacement et la nourriture. La surface arable est très limitée et pousse à même la faible couche de terre peu riche en nutriment par dessus la pierre, ce qui oblige les paysans à utiliser des plantes nécessitant peu d'énergie pour grandir. De plus, l'eau est une ressource très rare en altitude. Le mur stoppant les nuages, il pleut très fréquemment. Mais les nuages n'existent pas au dessus de 4000m. Monter étant très épuisant, une hiérarchie s'est donc organisée et est fonction des étages. Les étages inférieurs, ou il pleut en permanence, sont spécialisés dans la culture du bois (les UV passent à travers les couches successives de nuages, et au fil des ans, les arbres ont évolués et ressemblent plus a des arbres de mangroves qu'autre chose, tout en ayant le haut d'un dragonnier de Socotra), dans la recherche et l'exploration, ainsi que la technologie (ils reçoivent toutes les denrées). Plus haut, l'univers est rempli de champs, à perte de vue. Des escaliers géants fabriqués en bois servent à relier les étages entre eux. Il existe aussi un système d'irrigation. Le système de chauffage fonctionna au charbon extrait dans les mines présentes à certains étages (les mines ne prennent pas de place, il y en a donc régulièrement). Une des choses les plus incongrues de ce monde est sûrement la livraison de colis aux étages inférieurs. En effet, les colis sont… lancés d'étages en étages, parachutés pour être exacts, grâce à de grands filets de laine. La laine, voila aussi un gros problème. La laine est essentielle pour s'habiller et pour moult applications industrielles. Il existe donc dans trois étages situés à environ 80 000m des élevages géants alimentant une grande partie de la population en nourriture. L'énergie provient des exploitations minières. Le RDC est situé en bord de mer, la pêche est donc une activité très très commune. Dans les premiers étages, des industries telles que les colles… existent. L'électricité a été découverte mais créer des ascenseurs est un projet très long, complexe, risqué et cher.

L'argent a son influence ici. les endroits situés près du niveau de la mer sont très cotés, et une catégorie de population assez élevée gouverne ces étages. Bien entendu, il existe une certaine mixité sociale due aux activités tertiaires. Il y a aussi un dinausaure.

Items: Trois items. Une Epee, un Beamer et une vasque trop lourde.
Tous les items sont dans la salle de départ.

Personnages: jean, un dinausaure.

I.G) **Situatons gagnante et perdantes:**
Situation gagnante: parler à Jean.
Situation perdante: faire 15 déplacements.

I.H) **Objectifs divers:** Ramasser des items et se ballader et regarder les images.

II) **Réponse aux exercices.**

Exercice 7.7) GetExitString retourne sous la forme d'une String les différentes sorties d'une pièce.

Exercice 7.8) La HashMap a été implantée ici. C'est un tableau associatif, contenant en clé le nom de la sortie et liée à une pièce.

Exercice 7.9) keySet retourne un Set des clés contenues dans cette Map

Exercice 7.10) Idem que la 7.7. (même question)

Exercice 7.16) La boucle for étendue (aussi appelée for each) est une boucle qui agit comme ceci:
On a 2 termes séparés par un ":". Le premier prendra successivement la valeur de chaque élément du deuxième (le deuxième est une collection ou comporte plusieurs valeurs).

Exercice 7.18) Pour créer un bouton, il faut l'insérer dans un Component, celui qu'on veut. Si on veut lui faire écouter une action faut que la classe extends ActionListener. on peut faire pleins de choses avec cette classe, c'est marqué ici: https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html

Exercice 7.20-7.22) On va créer un HashMap d'items (String key, item élément) dans les salles et dans la classe Player si elle est crée. On le crée vide.
On va ensuite ajouter des objets dans nos salles via une méthode setItem(String nom, Item item).

Exercice 7.23) Pour la fonction back(), il faut tout d'abord créer un attribut pour enregistrer la salle précédente. Si on appelle back() on change la salle courante et on affiche l'image de celle ci.

INCONVÉNIENT: On ne peut back qu'une seule fois (jeu de 2 salles).

Exercice 7.26) avec un stack, on peut dévier ce problème. A chaque fois qu'on va changer de salle, on enregistre cette salle dans un Stack, et la méthode back() nous permet de retrouver la salle précédente et de l'afficher.

Exercice 7.28) Création de la commande test (Pour d'abord 1 fichier et après 2)
On va demander à test un paramètre (une String) qui sera le chemin relatif à la racine du fichier.
On va ensuite créer un parser, ainsi qu'une nouvelle variable de type File permettant de récupérer un fichier.

On va ensuite (Si le fichier a été trouvé) lire ligne par ligne le fichier jusqu'à la dernière ligne et exécuter la commande correspondant à la ligne (renvoyée sous forme de String).
Sinon, on va renvoyer un String qui dira que le fichier n'a pas été trouvé.
(Au début, le fichier est déjà mis en place).

Exercice 7.29) On va extraire de GameEngine les paramètres liées au joueur. Cad L'inventaire, la pièce courante ainsi que les pièces visitées précédemment et le poids total des items.
On peut migrer toutes les méthodes du type eat(), look(),take(),drop() qui utilisaient la pièce courante précédemment.

Exercice 7.30) On ajoute une HashMap du type String en key et Item pour stocker les Items.
On va ensuite créer les commandes take et drop qui vont respectivement prendre un objet présent dans la salle actuelle pour le mettre dans l'inventaire du joueur et déposer l'item du joueur dans l'inventaire de la salle. On renvoie un message de non erreur ou d'erreur en fonction de si l'action a été ou non réussie.

7.31) Dans cet exercice , il faut permettre au player de porter plusieurs items. c'est fait.

7.31.1) Dans cet exercice, on crée une classe ItemList qui a pour but de gérer les inventaires des rooms et du joueur.
Les méthodes qui existent en double dans room et dans player sont donc transféré dans cette classe, cela inclut les méthodes de transfert, d'ajout, de vérification.

7.32) Dans cet exercice, il faut limiter le poids que peut porter le joueur.
Un nouvel attribut est créé dans player : le pdsMax.
Quand on utilise la commande take, on vérifie si le poids actuel de l'inventaire + le poids de l'item est inférieur au Carry Max. Il a fallu créer une nouvelle méthode dans ItemList pour renvoyer le poids de l'inventaire.

7.33) Ajout d'une commande inventaire, qui affiche tous les objets de l'inventaire, avec la méthode getItemString() de ItemList.

7.34) Pour cet exercice, il faut ajouter un item mangeable, qui permet d'augmenter le CarryMax.
les commandes prennent un player en attribut donc il n'ets pas difficile de créer une méthode ds Player augmentant le carry max.

7.35) Dans cet exercice, il faut intégrer les enums de zuul-with-enums. Le plus complexe dans cet exercice était d'abord de comprendre les enums, puis de les intégrer sans faire de régression.

7.35.1) Dans cet exercice, il faut intéger le switch dans processCommand(). On remplace le grand if(...equals(...)) par un switch(vWord).
vWord étant d'un type énuméré , on peut comparer sa valeur directement avec ==, d'où la pertinence du switch, qui permet de simplifier l'écriture. Par exemple, case : LOOK ; permet de décrire le code a exécuter dans le cas ou vWord == LOOK. Ce code s'arrête au break ;.
default : permet de décrire le cas où aucune des valeurs ne correspond, mais pas utilisé.

7.41.1) Dans cet exercice, les enums sont modifiés pour intégrer des constructeurs.

7.42) Dans cet exercice, il faut intégrer une limite de temps ou de déplacement. J'ai donc ajouté un attribut au player correspondant au nombre de déplacement restant ; Lors de l'utilisation de la commande Go, cet attribut est décrémenté, et s'il atteint n, le player est gameoveré
Elle est de 15.

7.42.2) Début et fin de la programmation de l'IHM. j'ai décidé de ne rien faire de plus.

7.43) Dans cet exercice, il faut intégrer une porte à sens unique.
j'ai mis un attribut boolean pour savoir si on pouvait revenir en arrière. Ca marche.

7.44) Dans cet exercice il faut intégrer un beamer. Le beamer est une nouvelle classe qui hérite de Item, et qui possède un attribut aSavedRoom.

Une nouvelle commande charge est créée . Lors de son utilisation, on vérifie si le beamer est dans l'inventaire, puis on remplace la aSavedRoom par la CurrentRoom.
Une autre Commande Use est créée : elle vérifie la présence du beamer dans l'inventaire, vérifie s'il est chargé, et si c'est le cas, le player et téléporté à l'endroit sauvegardé.

Il ne peut plus se retéléporter tant qu'iln'a pas chargé le beamer.

7.46) Dans cet exercice, il faut créer une Room qui renvoie dans une room aléatoire du jeu. Pour cela j'ai créé deux nouvelles classes : TransporterRoom, qui hérite de Room, et RoomRandomizer.

TransporterRoom ne possède qu'une seule porte, mais une multitude de sorties. On Override le getExit pour faire appel au RoomRandomizer, qui renvoie une sortie aléatoire parmis toutes les sorties de la transporterRoom.

RoomRandomizer est une classe qui ne possède que des méthodes static : la plus importante.

GetRandomRoom accepte une ArrayList en paramètre, génère un nombre aléatoire compris entre 0 et le nombre d'élément de la liste, et renvoie l'élément correspondant du tableau.

7.46.1) Dans cet exercice, il faut ajouter une commande alea, qui permet de forcer la transporter room à renvoyer une room spécifique. Pour cela j'ai créé un attribut static dans RoomRandomizer, un entier.

Si cet entier a une valeur différente de null, alors RoomRandomizer renvoie forcément la Room correspondant à cet entier du tableau.

Cet entier est initialisé avec le second mot de la commande, et si la commande est réutilisée sans second mot, l'entier redevient null.

7.47) Dans cet exercice, il faut intégrer l'organisation de zuul-even-better avec les abstract command. Cet exercice est très long: en dehors des modifications à faire aux enums, et au parser , il suffit juste de déplacer le code depuis Player et GameEngine dans de nouvelles classes, et de changer les accesseurs utilisés.

7.47.1) Dans cet exercice il faut utiliser des packages dans notre jeu.
La javadoc montre le résultat

7.53) Création de la méthode main dans la classe Game.

7.54) Exécution du jeu avec le terminal de Linux et de Windows.

7.58) Création d'un fichier .jar exécutable. Il faut modifier toutes les URL des images.

7.53) Création de la méthode main dans la classe Game.

7.54) Exécution du jeu avec le terminal de Linux et de Windows.

7.58) Création d'un fichier .jar exécutable. Il faut modifier toutes les URL des images.

Déclaration anti plagiat)
L'intégralité du code, à l'exception des parties fournies dans les versions de Zuul, ont été rédigées par moi-même.
Toutes les autres images utilisées sont Creatives Commons, de créateurs indiqués sur les images, et n'étant pas utilisés à des fins commerciales, sans modification graphiques de celles ci (le rendimensionnement n'entrant bien sur pas en compte).