Rapport du jeu Zuul

Auteur : DEBELLE Hugo **Groupe :** 4

Titre : L'affaire au musée

Phrase thème : Un détective dans un musée devant résoudre des énigmes en changeant de salle pour trouver comment un objet a été volé

Résumé du scénario :

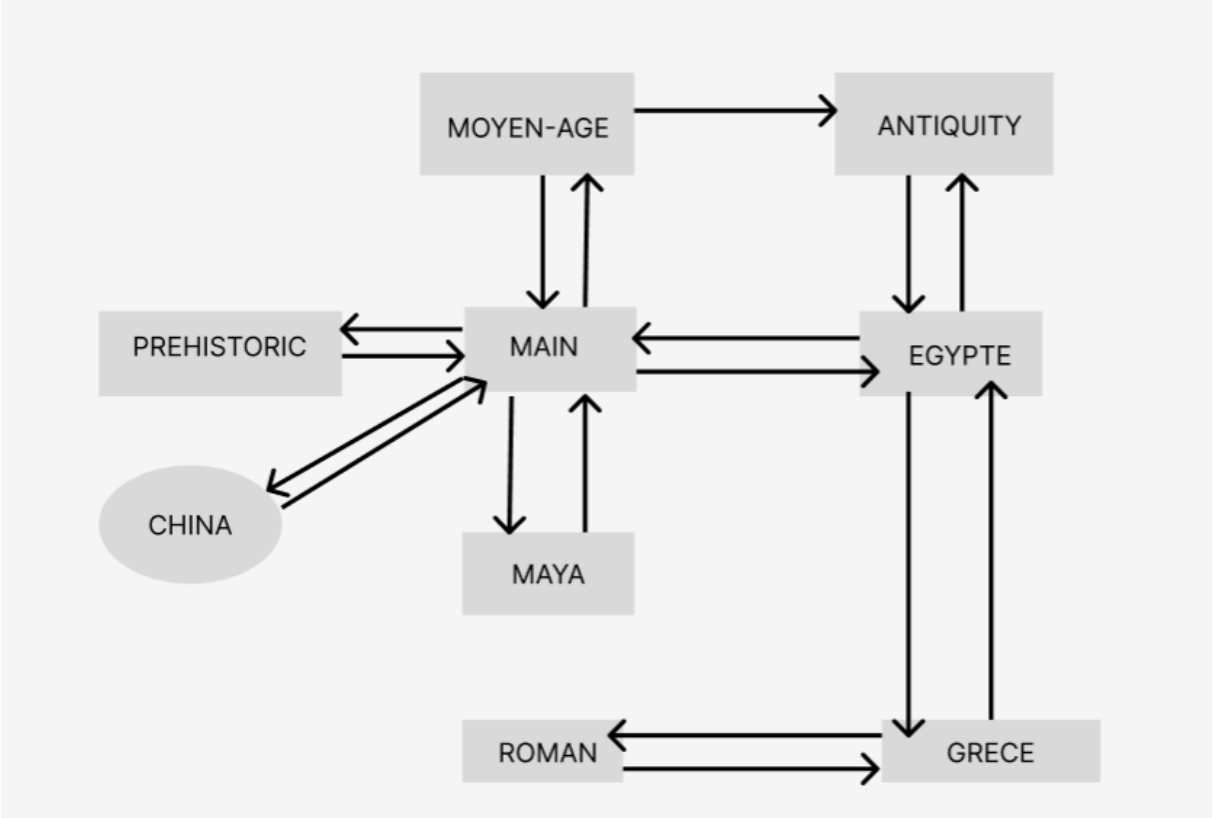
Un détective dans un musée devant résoudre des énigmes en changeant de salle pour trouver comment un objet a été volé. Pour gagner, il doit choisir la bonne phrase retraçant les événements du vol. S'il choisit la bonne phrase, il gagne sinon il perd.

Plan réduit :

Rectangle signifie salle à la même hauteur

Ovale signifie salle à une hauteur plus haut

Flèche signifie liaison disponible d'une salle vers l'autre



Scénario réduit :

TODO

Détail des lieux, items, personnages :

TODO

Situations gagnantes et perdantes :

TODO

Enigmes, mini-jeux, combats :

TODO

Commentaires :

A remplir à la fin du jeu

Réponses aux exercices

Exo 7.5 :

J'ai dû modifier la methode goRoom et printWelcome pour qu'elle appelle printLocationInfo permettant de supprimer la répétition de code

Exo 7.6 :

J'ai ajouté la fonction getExit(direction) dans Room permettant de retourner la piece en fonction de la direction et d'éviter la répétition de code. Dans la procédure printLocationInfo j'ai fait un

for de toutes les directions évitant la répétition des ifs. Plus tard, il faudrait surement créer une fonction retournant un tableau de toutes les directions possibles

Exo 7.7 :

J'ai créé la fonction getExitString() qui retourne la String avec toutes les sorties disponibles pour la room actuelle. J'ai ensuite remplacé le for de la procédure printLocationInfo par l'appelle de cette fonction.

Il faut mettre getExitString() dans Room car c'est dans room que on a toutes les sorties et qu'on sait si elles sont null ou non. Aussi, si on veut ajouter une sortie, on a juste a modifier cette fonction. On affiche dans Game car c'est dans cette procédure (printLocationInfo) qu'on s'attend à l'affichage. Room ne doit rien afficer

Exo 7.8 :

Il y a la méthode setExits qui devient inutile puisque remplacé par la méthode setExit (sans s) car elle permet de mettre une et une seule room dans une direction souhaitée

Exo 7.9 :

Ajouts du keySet dans GetExitString permettant d'itérer sur toutes les directions crées. Cela nous évite de mettre toutes les directions à la main

Exo 7.10 :

La méthode getExitString permet de retourner un String contenant les sorties disponibles de cette room. Pour cela on itère sur toutes les directions créent et ajoute ces directions dans le résultat final.

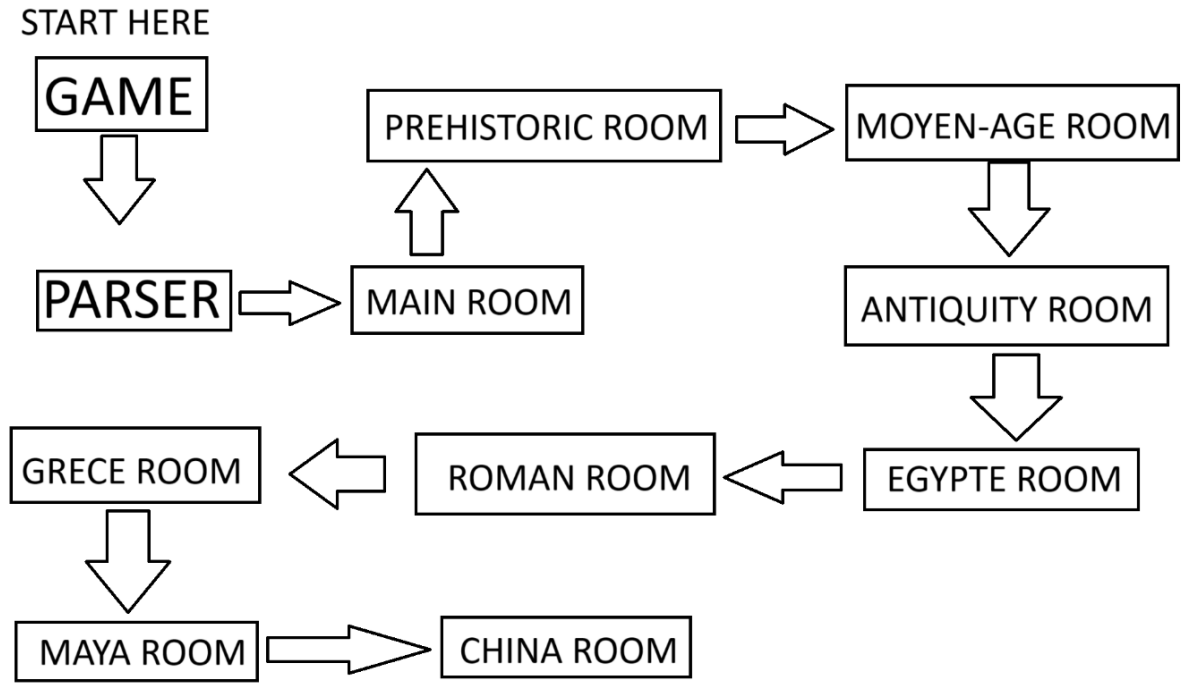
Exo 7.10.2 :

Class Game contient moins de méthode que la class Room dans la javadoc car la javadoc affiche que la doc des méthodes publiques hors la class Game possède beaucoup de méthode privée

Exo 7.11 :

On a créé une fonction getLonDescription dans la classe Room permettant de retourner la description à afficher au joueur. On créer ceci dans la classe Room car si on souhaite ajouter ders infos dans une Room on a juste cette classe à modifier et non cette classe + la classe Game

Exo 7.12 :



Exo 7.13 :

Le diagramme ne change pas durant l'exécution de go command car toutes les rooms et les objets en général sont déjà créent

Exo 7.14 :

On a ajouté la commande look dans CommandWords et dans processCommand permettant d'exécuter la commande look. Enfin, on a regardé s'il y avait un second mots et si c'est le cas, on envoie un message d'erreur

Exo 7.15 :

En suivant le même principe que la commande look, ajout de la commande eat

Exo 7.16 :

Ajout des procédures showAll et showCommands permettant de print toutes les commandes disponibles. Utiles pour help car quand on ajoute des commandes, elles sont automatiquement print et on a plus besoin de les rajouter à la main pour le help.

Exo 7.17

Si on veut ajouter une commande, on doit toujours modifier la classe Game car on doit modifier la fonction processCommand qui permet d'executer une méthode souhaiter en fonction de la commande tapé

Exo 7.18 :

Rename de la méthode showAll en getCommandList et showCommands en getCommands.

En effet cess 2 méthodes retournent maintenant la liste des commandes au lieu de les afficher directement.

Exo 7.18.2 :

Remplacer la concatenation d'un String par StringBuilder.append() dans les fonctions Room. getExitString() et CommandWords.getCommandList()

Exo 7.18.6 :

Dans Parser, nous n'utilisons plus la class Scanner car au lieu de lire l'input du joueur dans la fonction getCommand, on passe l'input du joueur en paramètre de cette fonction.

Exo 7.18.7 :

La ligne this.aEntryField.addActionListener(this); permet d'ajouter un actionListener à l'entryField c'est-à-dire écouter quand on appuie sur entrer dans le field pour prendre le texte tapé et exécuter la commande. On met this en paramètres, car on veut écouter dans cette classe. C'est pour cela qu'il y a la procédure actionPerformed(pE) car elle est appellé quand le joueur fait entrer dans le inputField et permet de faire les actions décrits ci-dessus (exécuter la commande en fonction du texte tapé)

Exo 7.18.8 :

On a ajouté un Bouton « help » avec un évent dans la classe courante. Donc vue qu'il y a field et boutton qui possède des events, il faut maintenant regarder dans actionPerformed(pE) qu'elle « source » appelle cet événement. On met : if (pE.getSource() instanceof JButton) et exécute la commande help si c'est un bouton, sinon on exécute processCommand car ça signifie que la source est l'entryField

Exo 7.19 :

MVC pattern peut être utilisé dans le jeu. En effet il permettrait de faire des parties presque indépendantes permettant moins de modifications si un changements est effectués ? Dans le jeu, on voit un début d'implémentation. En effet, Room gère que les rooms, GameEngine les commandes et UserInterface l'interface utilisateur. Néanmoins, il peut être améliorer en créent par exemple un CommandControler permettant de gérer les commandes (car c'est pas logique de la faire dans GameEngine)...

Exo 7.21 :

Les infos des items dans les Rooms doivent être produite dans la class GameEngine

C'est Item qui doit produire son String.

C'est le GameEngine qui doit afficher les infos des items car il affiche déjà toutes les autres.

Exo 7.21.1 :

On ajoute juste une condition pour regarder si le second mot correspond à un nom d'item et si c'est le cas on affiche ça description longue

Exo 7.22.1 :

J'ai utilisé une HashMap comme ça on peut retrouver un item directement avec son nom. C'est plus rapide qu'avec une List dans laquelle on aurait dût faire un for de tous les items et comparé chaque nom

Exo 7.25 :

Vue qu'on accept pas de second mot, on ne peut pas taper back plusieurs fois dans la même commande

Exo 7.27 :

Il serait intéressant de tester la fonction goRoom, la mise en place d'un Item dans une room ou les sorties des rooms

Exo 7.28 :

Un test peut être automatisé en écrivant une suite d'instruction que le test va exécuter et comparé au résultat attendu, par exemple une suite de commande dans un fichier.

Il faudrait modifier la classe UserInterface qui devrait lire chaque commande d'un fichier au lieu d'attendre que le joueur tape une commande. On peut aussi mettre ce mécanisme dans GameEngine si on veut que test soit une commande et non un fichier java externe.

Exo 7.28.1 :

On créer une commande test qui permet de lire chaque ligne du fichier. Chaque ligne appelle interpretCommand avec son contenue

Eco 7.29 :

On créer une class Player qui contient un nom, sa salle actuelle, les items qu'ils possèdent et les anciennes salles visitées. On ajoute une fonction back et goRoom qui retourne true si l'action a été effectué, sinon false. Dans le GameEngine, si les actions ont été effectués, on appelle printLocationInfo sinon on envoie un message d'erreur.

Exo 7.30 :

On ajoute take & drop dans Player et on retourne true si l'action c'est exécuté sinon false. On affiche printLocationInfo ou le message d'erreur on fonction de la valeur retournée.

Exo 7.31 :

On modifie juste drop & take pour prendre en paramètres le nom de l'item et le supprimer de la collection d'item

Exo 7.31.1 :

On créer ItemList avec addItem, removeItem, getItemByName & getItemString. On l'ajoute à Player et Room en supprimant le code devenu inutile

Exo 7.32 :

On ajoute un getWeight à ItemList et dans take, on regarde si le poids des items portés + celui qu'il veut prendre son supérieur au poids max. Si c'est le cas on retourne false.

Exo 7.33 :

Ajout de la commande inventory. On a juste à print le itemList.getItemStrring()

Exo 7.34 :

On ajoute une procédure use au joueur avec l'item à utilisé en paramètre. On lui supprime cet item et on fait l'action souhaité avec cet item.

Exo 7.35 & 7.35.1 :

On crée l'enum CommandWord dans lequel on ajoute toutes les commandes. Ensuite on modifie le Command pour qu'il accepte CommandWord au lieu d'un String. Enfin, on modifie le Parser pour qu'il appelle une fonction getCommand de commandWord qui retourne la commande en fonction de son nom. Pour cela, dans son constructeur, on a mis un boucle for permettant d'itérer sur toutes les commandes créées et de les ajouter dans la HashMap avec en key le nom tout en minuscule.

Exo 7.37 :

Pour ajouter un nouveau nom pour une commande déjà existante, il faut juste modifier la class CommandWords ce qui montre que le code respecte bien les règles précédentes permettant d'avoir à modifier le code qu'à un seul endroit

Exo 7.38 :

Quand on change le nom de la commande help, on remarque qu'il ne change pas dans le message de bienvenue. Il faudrait faire prendre dynamiquement celui renseigné dans la HashMap de CommandWords

Exo 7.40 :

On ajoute un compteur de help à 5 puis on le décrémente de 1 quand on tape help. S'il est inférieur à zero, on affiche un message disant que ne peux plus accéder à l'aide.

Exo 7.42.2 :

On se contentera de l'IHM actuelle pour les rendus intermédiaire et on changera pour le rendu final

Exo 7.43 :

J'avais déjà implémenté dans la map une trapdoor mais je n'avais pas fait le cas du back. Pour cela j'ai juste créer la fonction isExit(Room) que j'appelle et si elle retourne false, je n'exécute pas la commande back

Exo 7.44 :

Ajouter d'une classe Beamer contenant les informations s'il est chargé et la salle dans lequel il a éré chargé. Enfin, ajout des commandes charges et fire qui respectivement ajoute la salle actuelle de joueur et le téléporte à la salle chargée.

Exo 7.45 :

Ajout de la classe Door (car pas fais avant) et LockDoor. On ajoute canPass dans Door et on l'appelle dans Player.goRoom . Ensuite LockDoor extends Door donc on regarde si le joueur possède la clé, si c'est le cas on le laisse passer sinon non

Exo 7.46 :

Ajout de la classe TransporterRoom permettant de transporter une room aléatoire. Les seuls ajouts sont dans cette class montrant la qualité du code.

Exo 7.46.1 :

Ajout de la commande alea. Ajout de la fonction interceptCommand avec en 2ème paramètre un boolean pour savoir si on est en test ou non. Cette fonction est appellé que quand on est en test sinon c'est la fonction avec 1 seul paramètre qu'il l'appel avec le boolean à false.

Exo 7.46.2 :

J'ai déjà utilisé l'héritage pour les exercices 7.43 et 7.45 et cela améliore la qualité du code puisqu'on doit juste modifie une classe au lieu de plusieurs.

Déclaration obligatoire anti-plagiat

Aucun code n'a été repris. Seul le code fournis dans les exercices a été utilisés.