Rapport de projet Java

Titre provisoire

Colors

I.A) Auteur(s) :

Erwan HAMZAOUI

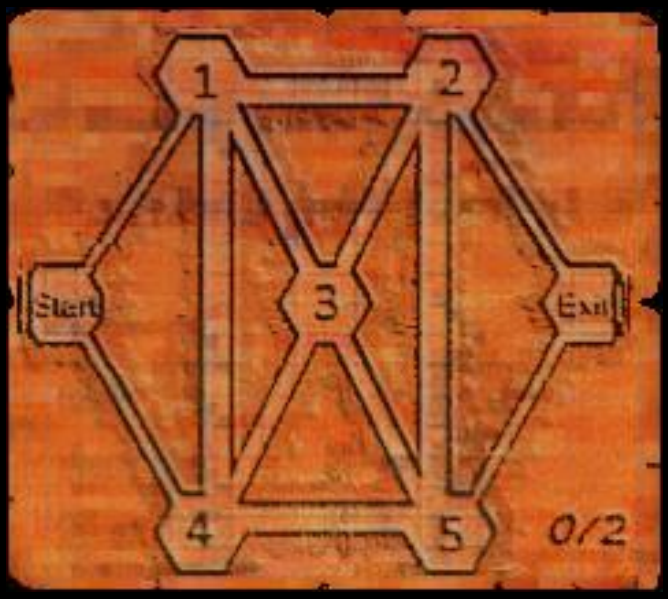
I.B) Thème (phrase-thème validée) :

Dans un donjon, le jouer doit réussir à sortir en moins de deux heures.

I.C) Résumé du scénario complet (voir exercice 7.3.1)

Pendant l'exploration d'un donjon avec votre coéquipier, vous êtes rentrés dans une nouvelle salle et la porte c'est refermée derrière vous. Vous allez devoir trouver une autre sortie mais votre temps est compté à cause de l'air toxique.

I.D) Plan (complet, avec indication de la partie "réduit" si exercice 7.3.3)



I.F) Détail des lieux, items, personnages

Personnages : Le joueur

Lieux : Entrance, Room1, Room2, Room3, Room4, Room5, Exit, Outside

Hole(sous entrance, salle de test beamer/random)

Items : 'paper', redCrystal, blueCrysal, greenCrystal, purpleCrystal, yellowCrystal, 'map', 'map2', 'map3', beamer

I.G) Situations gagnantes et perdantes

Situation gagnante : Le joueur sort du donjon

Situation perdante : Le temps est écoulé

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

Le jeu est basé uniquement sur des énigmes, la solution est à la fin du rapport

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

Je ne sais pas pourquoi mais lorsque je lance mon jeu avec BlueJ, tout fonctionne bien mais lorsque que je le lance directement avec java, le jeu fonctionne mais il n'y a plus de son et les fichiers de tests ne sont plus trouvés.

J'ai remarqué que ces deux parties étaient gérées par des try/catch mais je ne sais pas résoudre ce probleme.

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

7.2.1 La classe **Scanner** permet d'analyser un type primitif dans le flux passé en paramètre, ici **System.in** qui est le terminal java par défault.

.nextLine() renvoie lit la ligne tapé, .next() renvoie le mot suivant.

7.5 J'ai dût dissocier la création de la String avec l'affichage de la String. La classe Room est la mieux placée pour indiquer les sorties d'un objet de type room et c'est donc elle qui créé la String . La classe game se contente d'afficher la phrase retournée par getExitString ()

7.6 Il n'est pas indispensable de tester si le paramètre est null car l'absence de sortie dans une direction correspond à la valeur null donc on peut l'attribuer même dans ce cas-là.

Dans cet exercice, on remplace tout nos attributs exits par une hashmap contenant ces directions associés à la room. Les méthode peuvent maintenant être appliquées à l'ensemble des élements de la Hashmap même ceux qu'on rajoute par la suite et non plus un par un .

7.7 La classe Room est la mieux placée pour indiquer les sorties d'un objet de type room

La classe game est celle sur laquelle on appelle les procédures d'affichage donc il est logique que cette procédure soit dans Game et se contente d'afficher la chaine de caractères retournée par getExitString ()

7.8. , 7.9 ,7.10La méthode getDescription est maintenant inutile car on ne s'en sert plus en dehors de Room, dans lequel on peut de toute façon juste le remplacer par l'attribut privé qu'il return.

Les HashMap ressemblent à des tableaux à la différence qu'une valeur peut utiliser comme indice un autre type qu'une , à une String par exemple

.put(key,value) ajoute une valeur à la hashmap, key etant « l'indice »

.get(key) renvoie la valeur associé à cet indice

.containsKey(key) renvoie true si il y a un indice égal à key dans la Hashmap

.containsValue(value) renvoie true si il y a une valeur égal à Value

.remove(key,value) supprime l'association key value de la Hashmap

.keySet() renvoie l'ensemble des indices de la Hashmap

(Set <Type> allKeys =HashMapName.keySet())

for(keyType vName allKeys) {...} va parcourir l'ensembles des indices allKeys en attribuant chacune leur tour leur valeur à la variable vName dont on à précisé le type (keyType) et executer le code entre accolades

7.8.1 Il suffit d'ajouter dans la hashmap de la room une valeur d'indice « down »

7.10.2 Game à becoup moins de méthodes dans la documentation car ses méthodes ont un accés privé alors que celles de Room ont un accés publique car game doit y accéder.

7 .11 Au lieu de récupérer dans game chaque infos de room séparement, on les combinent sous forme de String directement dans Room.

7.14 Il suffit de rajouter une commande qui affiche getLongDescription()

7.16, 7.18 De la même manière que pour room, c'est à commandWords de créer les String de commands autorisés et game se contentera de l'afficher.

On ne choisit plus les commandes à afficher une à une mais on affiche directement toutes les commandes contenues dans le tableau de commandes valides. La classe Parser fait l'intermédiaire entre game et CommandWords pour éviter de créer un nouveau lien et donc une nouvelle dépendance.

7.18.6 . Il n'y a plus besoin de scanner car on écrit plus dans un terminale mais dans un champs de texte qu'on à ajouté dans lequel on peut récupérer directement avec getText()

JLabel=Image

JFrame=Titre+Fenêtre,

JTextField=Champs de texte,

JTextArea affiche le texte,

JScrollPlane=pour défiler,

JPanel=affiche les éléments à l'emplacement indiqués

addPanel() rajoute un composant au conteneur qu'on veut afficher

7.18.7 . J'ai utilisé la fonction getSource() pour différencier l'action du bouton et du champs de text

7.18.8. ActionListener rajoute les méthodes liées aux actions

addActionListener() permet d'appeler la méthode actionPerformed() quand il y a une action sur l'objet

actionPerformed() est la méthode qui est appelée lorsqu'une des actions est effectué dans la classe entre parenthèse de addActionListener( )

actionEvent est un objet qui contient des informations sur l'action effectuée

getActionCommand() donne la valeur du champ de texte pour JTextField et la valeur définie par setActionCommand() pour un JButton

getSource() donne le composant qui doit effectuer une action

7.19.2 Il suffit d'ajouter Nom du dossier/ devant le nom du fichier

7.20 J'ai créée une nouvelle classe Item avec un nom et un poid et nouvelle méthode et attribut dans room qui permet à une room d'avoir un item et de le placer dedans .

7.21 Mes items n'ayant pas besoin de déscription, je n'ai pas rajouté cet attribut.

7.22 Je remplace l'attribut de type item de room par une HashMap d'association Items String pour qu'elle puisse en contenir plus de 1

7.23 Il faut pouvoir mémoriser à chaque fois la dernière pièce donc je créer un nouvel attribut LastRoom. Cet attribut prend la valeur de la salle courante dans la procédure goRoom avant de changer la salle courante puis la commande back attribut à la salle courant la last Room.

7.26. **Stack** contient la classe pile et la classe file

pop() retire le dernier élément de la pile

Push() rajoute au sommet de la pile l'élément entre parenthèses

Empty() return vrai si la pile est vide

Peek() return l'élément au sommet de la pile

Clear() vide la pile

On rajoute à chaque déplacement l'ancienne salle courante au sommet de la pile et on retir e la salle au sommet à chaque utilisation de back .

7.28.1 J'ai créée un nouveau un nouveau scanner avec comme parametre mon document .txt, puis avec une boucle while, le scanner va prendre la valeur de chaque ligne et les executer successivement. try et catch servent à traiter les exceptions

7.29 Je créer une nouvelle classe Player avec comme attributs l'item porté et la salle courante initialements présents dans GameEngine, cela permet d'avoir plusieurs objets Player et de ne pas surcharger la classe GameEngine.

7.30 . Take supprime l'items associé au nom passé en parametre de la Room courante et l'ajoute à l'inventaire du joueur. C'est l'inverse pour drop

7.31 Comme pour Room, je remplace l'attribut item par une Hashmap d'items

7.31.1 Player et Room ayant tout deux une liste d'items sur laquelle ont peut appliquer exactement les même méthode, on créer un classe liste d'items pour éviter la dupplication de code et on remplace l'attribut hashmap par un attribut liste d'items contenant la hashmap

7.32 On ajoute udeux attribut à player avec le poid porté et le poid max portable. On ajoute avant l'utilisation de take une condition qui vérifie que le poid porté+ le poid de l'item ne dépasse pas le poid max

7.33 La commande inventaire appelle une méthode dans ItemList du joueur qui parcours la hashmap dans itemList et return une String avec le nom de tout les items

7.34 Je rajoute un modificateur de poids max du joueur et si la commande eat est utilisé sur l'item cookie, le poids du joueur augmente.

7.42.1 Pour créer un timer avec le temps réel j'ai utilisé la fonction System.currentTimeMillis() qui me donne le temps actuel en millisecondes

Je créer 1 variable double égale à System.currentTimeMillis() à mon temps au lancement du décompte et soustrait ce temps au temps actuel pour obtenir le temps écoulé . J'ai converti ce temps en secondes (/1000) , arrondi (int) et j'ai fais une procédure qui m'affiche ce temps après convertion au format heures :minute :secondes. J'ai créer ce Timer dans un nouvelle classe Timer.

7.42.2 J'ai affiché le jeu en plein écran avec la fonction Dimension vScreenSize = Toolkit.getDefaultToolkit().getScreenSize(); qui me donne la taille de mon écran

J'ai affiché des Items par dessus l'image de la salle en créant de nouveaux JLabel avec des fonds transparents

J'ai affiché le temps écoulé avec un nouveau jTextArea.

J'ai changé la couleur et police d'écriture avec setBackground(), seForeground et setFont() ;

7.35-7.41 Fait

7.43 J'ai ajouté une trap Door (entre l'entrée et hole en dessous)

J'ai créée une fonction isExit() dans Room qui prend un parametre de type room et return true si cette Room est dans la liste d'exits

Dans goRoom(), je rajoute un if() qui clear la hashmap de back si la salle précedente n'est pas une sortie de la salle courante.

7.44 J'ai rajouté une classe Beamer qui hérite de la classe items avec un attribut en plus, la salle sauvegardé.

La commande charge accede au modificateur de cet attribut et fire change la salle courante du joueur pour la salle de cet attribut, modifit l'attribut à null, vide le stack de back (si le joueur à bien l'item beamer).

7.45.1 Actuellement 4 fichiers de tests : **essai** test toutes les commandes, **langage** test toutes les commandes dans une autre langue, **court** test 4 commandes, **solution** finit le jeu de la manière la plus courte (pour un joueur qui n'a pas déjà le mot de passe)

7.46 Pour faire une Transporter Room (hole) j'ai créer une nouvelle classe qui hérite de room et j'ai redéfinit la méhode getExit() qui donne maintenant une room aléatoire. Pour cela j'ai dut lui ajouter un attribut de type hashmap<String,Room> avec toutes les salles du jeu dans lesquelles il peut être téléporté. La methode redéfinie créer une variable entier, lui attribue une valeur aléatoire en 0 et la taille de la hashmap avec =new Random().nextInt(taille), je créé un tableau que je remplie avec les rooms dans la hashmap et je return la room d'indice obtenu.

nextInt() donne un nombre « aléatoire » entre 0 et le nombre en paramètre.

La seed définit la séquence pseudo aléatoire. Avec la même seed on aura toujours les même tirages.

7.46.1 Pour la commande alea, j'ai ajouté dans transperoom un attribut de type room aAlea qui est definit par la commande « alea »+room et un attribut boolean aTest qui est mit true au debut de la commande test et remit false à la fin. Dans le cas ou aTest est true, la commande getExit renvoie directement la valeur de aAlea. Sinon fonctionne normalement.

7.47. J'ai déjà remplacé ces if par un switch

7.48 J'ai ajouté la classe Character avec une salle courante, un nom d'image et un dialogue.

7.49. J'ai ajouté une classe MovingCharacter héritant de Character avec en plus une méthode changeRoom()à l'aide d'un nouvelle méthode dans room getAllExit() qui renvoie un tableau avec toutes les sorties de cette room.

changeRoom() créer un nombre aléatoire de la taille de ce tableau et change la currentRoom pour celle dans le tableau de l'indice tiré. A la place d'un message qui indique la présence du MovingCharacter j'ai utilisé un image (qui est tiré aléatoirement pour un meilleur rendu) .

7.49.2 Tout est présent.

J'ai ajouté une musique de fond, des bruitages pour les différentes actions, un bouton pause et une commande note.

IV. Déclaration obligatoire anti-plagiat (*)

Je n'ai recopié aucune ligne de code en dehors de celles fournis

J'ai fai les images à l'aide d'images de l'artiste Reza Afshar

(https://twitter.com/rezaafsharr)

Musique utilisée
(https://www.youtube.com/watch?v=CeKfF5X5lMw&t=9s)

Bruitages libres de droits

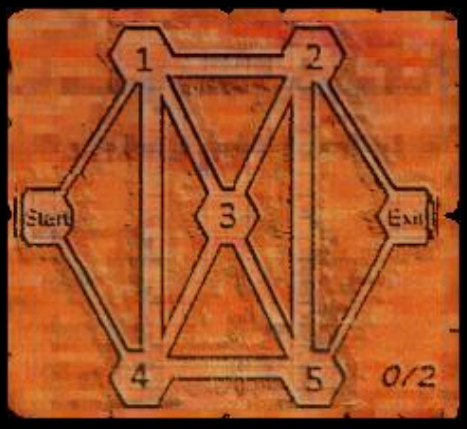
V, VI, etc... : tout ce que vous voulez en plus

(*) Cette déclaration est obligatoire :

- soit pour préciser toutes les parties de code que vous n'avez pas écrites vous-même et citez la source,

- soit pour indiquer que vous n'avez pas recopié la moindre ligne de code (sauf les fichiers zuul-*.jar qui sont fournis évidemment).

Resolution du jeu :



-Se rendre dans la salle 2 et ramasser l'item «'paper ' »

-Lire l'item avec la commande «look paper »

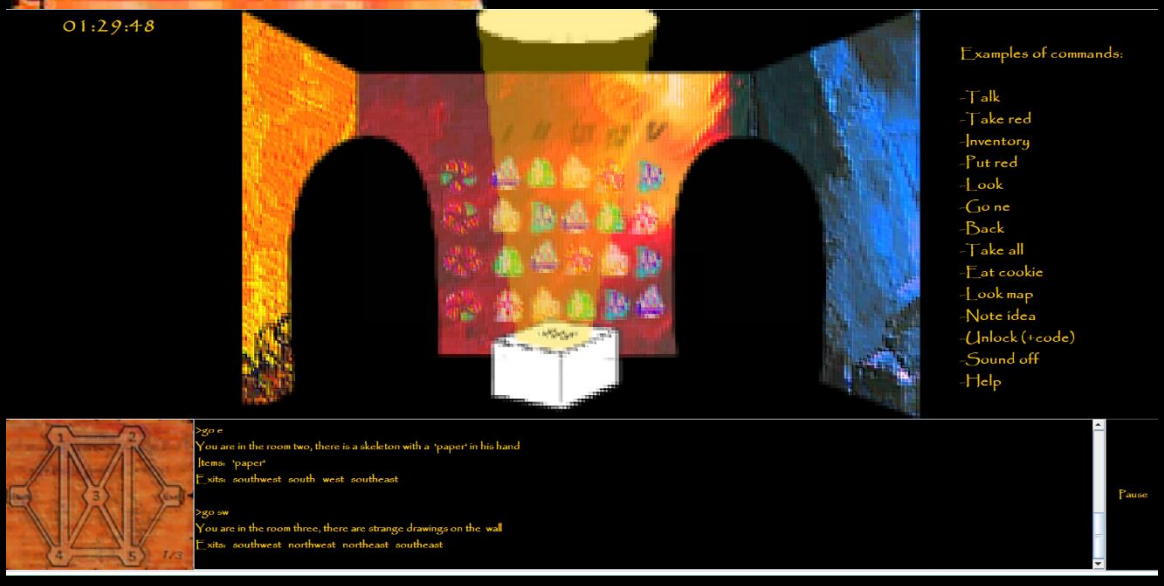
-Se rendre dans la salle 3, il y a d'étranges dessins sur le mur

Enigme 1 : Le joueur doit maintenant trouver le message caché dérrière ces deux élements. On lit sur le papier que la couleur rouge correspond à un mauvais placement et verte à un bon placement. Pour la première ligne par exemple, cela signifie que sur les 5 couleurs, 2 sont au bon emplacement et les 3 autres sont mal placés.

En testant les ordres qui fonctionne sur les 4 lignes, on obtient que 2 possibilités :

Jaune Vert Beu Rouge Violet et

Violet Bleu Vert Rouge Jaune



Les deux sont exactes et indispensables et correspondent à la couleur de crystal à placer dans chaque salle Room numérotés de 1 à 5.

-On retourn dans la salle d'entrée

-On mange le cookie pour pouvoir porter plus d'objets

-On ramasse tout les crystaux avec la commande « take all »

-Puis on va placer les crystaux trouvés dans les salles trouvés à l'énigme

-Le joueur obtient 2 nouveaux items « 'map1 ' » et « 'map2' » pour chacune des solutions de l'enigme 1.

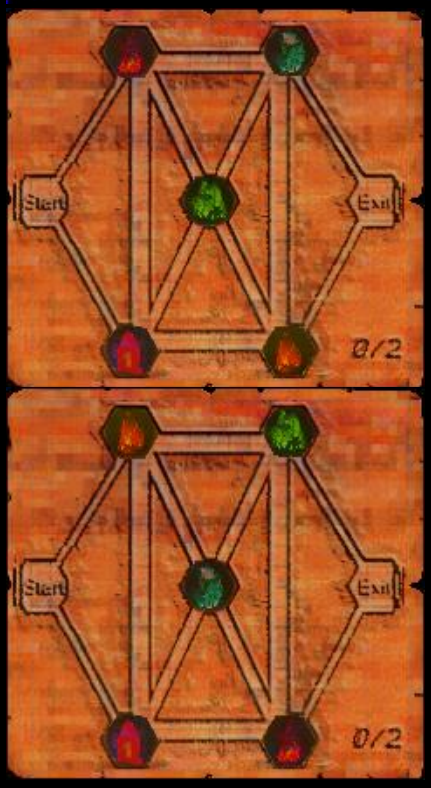
-Afficher les maps avec look map1 et look map2

-Se rendre dans la salle exit

-La porte de sortie est fermé, la commande unlock nous indique que l'on doit trouver un code avec des lettres

<u>Enigme 2</u> : Le joueur doit trouver le mot de passe à l'aide des Items map1 et map2 débloqués, l'item paper et les inscriptions sur les mur de la salle exit

Chaque lettre sur le mur correspond en fait à une couleur.



Par exemple YRGBP=Yellow Red Green Blue Purple

En reliant ces 5 couleurs sur les deux cartes on obtient respectivement le chiffre 2 et un symbole ressemblant à un k (mais n'en n'est pas un), on lit sur le papier que ce symbole correspond à la lettre E. Le chiffre correspond en fait à l'emplacement de la lettre dans le mot de passe.

En faisant de même avec tout les mots sur les murs, on obtient :

1T 2E 3R 4M 5I 6N 7U 8S

Il ne reste plus qu'à taper la commande « unlock TERMINUS »

La porte de sortie s'ouvre et le joueur sort avec la commande «go Outside » pour sortir et gagner le jeu

01:29:54