RAPPORT

Surfman (https://git.esiee.fr/surf-game/surf-man.git)

(A3P 2015/2016 J4)

auteur

- Lehmoudi Kamel

Thème

- Un surfeur dans une plage doit récuperer tout les planches de surfeur perdues pour gagner .

Résumé du scenario

Plan

Scénario détaillé

Lieux, items, personnages

Situation gagnante et perdantes

commentaire

Réponses aux exercices

- 7.5 : printinfo location : on régle le problème de cohesion en ajoutant une methode qui retourne l'information sur la salle courante.

- 7.6:getexit: on modifie les attribues de la classe room en private et on les encapsules ensuite on fait un accesseur pour decoupler la classe Room et Game

- 7.7: getexitstring: on ameliore la cohesion en separant l'info produite et l'affichage , l'info est produite dans Room et l'affichage se fait dans game

Réponses aux exercices

- 7.8: on remplace les 4 parametre de sortie par HashMap pour par la suite pouvoir rajouter d'autre direction plus facilement

- La méthode devenue maintenant inutile est getExit() car on peut accéder a la description d'une room dans le hashmap .

- 7.8.1:deux nouvelles directions ont était ajouté "bas" "haut" et une pièce utilisant ses deux directions vHole
- 7.9: la méthode keyset renvoie l'ensemble des clés que l'on a défini pour le hashmap donc ici les directions .

Réponses aux exercices

- 7.10: la méthode GetExitString() du code 7.7 doit retourner une string,
- Ensuite la première ligne initialise une string returnString,après on définit le type des clés du hashmap exits qui sont des string,on fait par la suite une boucle for qui parcours tout le hashmap et qui va ajouter au string returnString tout les directions en forme de string de la Room sur laquelle on a appelé la méthode , enfin on retourne returnString.

- 7.10.2: la java doc est ajouté , la classe game ne sert qu'a afficher de l'information alors que la classe room doit manipuler des données d'où la différence de nombre de méthodes.

Réponses aux exercices

- 7.14: on ajoute un tableau avec les commandes valides qu'on pourra modifier plustard pour ajouter d'autre commandes.
- On ajoute des instructions a la méthode processCommand() pourra effectuer un affichage lorsque une commande valide est reconnu.

- (je me suis arrêté a cette exercice,j'ai un problème avec la méthode processCommande.)
- 7.15: on rajoute une commande eat dans le tableau de string qui prend les commandes valides

Réponses aux exercices

- 7.16: il y a un problèmes dans la méthode print help car quand on rajoute des commandes il faut les rajouter dans cette méthode aussi.
- On modifie cette méthode.

- 7.18: on modifie la méthode showAll() pour respecter la règle de responsabilité .

- 7.18.5: on crée une nouvelle hashmap où est stocker toutes les pièce du jeux. On met dans createRooms() la méthode qui permet d'ajouter un par un les Rooms.

Réponses aux exercices

- 7.18.6: changement de la conception du jeux avec un interface graphique(nouvelle classe UserInterface)

- 7.18.8: ajout d'un bouton en rajoutant un listener sur un bouton que l'on crée et implemente dans la procédure createGui()
- 7.20: on crée une classe item avec des accesseurs et une hashmap pour ranger les items et pouvoir les rajouter dans nos différentes Rooms

Réponses aux exercices

- 7.21: on améliore la cohésion dans la classe Room

- 7.22: on modifie la classe Room pour pouvoir ajouter des items dans les pièces qu'on veut

- 7.26:
- Au lieu d'un attribut on crée une pile (stack) où l'on stock tout les pièces traversées, puis on les retourne quand la commande back est tapée,

Réponses aux exercices

- 7.26.1:nous avons généré la javadoc et créée un fichier .BAT .
- 7.28.1: nous implémentons une nouvelle commande test pour pouvoir lire un fichier depuis BlueJ
- 7.28.2:nous avons crées 3 fichiers texte (map;court;ideal) . Map.txt contient les chemins possibles et ideal.txt contient le chemin le plus long.

Réponses aux exercices

- 7.26.1:nous avons généré la javadoc et créée un fichier .BAT .
- 7.28.1: nous implémentons une nouvelle commande test pour pouvoir lire un fichier depuis BlueJ
- 7.28.2:nous avons crées 3 fichiers texte (map;court;ideal) . Map.txt contient les chemins possibles et ideal.txt contient le chemin le plus long.

Réponses aux exercices

- 7.29: ici on change la conception de la classe GameEngine pour séparer le déplacement du joueur des autres fonctions. Dans une autre classe (Player). Une classe = un rôle

- 7.30: on ajoute les fonction take et drop pour prendre les items se trouvant dans les rooms
- 7.31:on stock les items dans une hashmap,
- 7.31.1:on créée une classe ItemListe pour pouvoir bien gérer la gestion des items et ne pas créée des hashmaps chaque fois

Réponses aux exercices

- 7.32:on ajoute un attribut aMax et une condition dans la fonction take pour interdire de prendre des objets avec une valeur supérieure a aMax.
- 7.33: on créée une commande pour afficher l'inventaire du joueur
- 7.34:on créée un item qui nous permet de prendre d'autre item de poids plus grand
- 7.42:on ajoute une limite de déplacement avec un nouvelle attribut et des modifications
- 7.42.2:
- 7.43:on crée une fonction isExit() et un test pour interdire le retour en arrière de plus on vide la pile pour empêcher le retour via la commande back.

Réponses aux exercices

Déclaration anti-plagiat