GROUPE S2E : CROCHET Rémi & CINAR Erdal

Pour rappel il a fallu écrire les classes

- Elements
- Perso
- Caisse
- Depot
- ListeElements
- Labyrinthe
- Jeu
- Chargement
- MainJeu
- TestChargement
- TestJeu

Les classes Elements et ses héritiers étaient plutôt simple à écrire. Nous avons décidés que leurs attributs seraient en privés ce qui a induit l'écriture de 2 getter et 2 setter (pour x et y)

Pour la classe ListeElements il y avait peu d'indications sur comment la représenter alors nous avons décidés de la faire avec un ArrayList pour disposer de nombreuses méthodes déjà écrites. Nous avons toutefois rajouter quelques méthodes par exemple getElement(int,int) qui permet de trouver si un élément aux bonnes coordonnées est présent dans la liste. Nous avons aussi rajouter la méthode getElement qui convertit l'ArrayList en tableau ce qui en facilite sa manipulation (notamment pour l'accès au premier élément sans avoir à réécrire d'autres méthodes).

La classe Labyrinthe est essentiellement composé des constantes caractères, nous avons toutefois rajouter quelques méthodes pour pouvoir la manipuler plus facilement. Nous avons un getter pour récupérer le tableau murs. Nous avons 2 setters différents. L'un qui prend un tableau en paramètre et le met en attribut du labyrinthe, l'autre qui pour une case donnée la transforme en murs. Nous avons aussi la méthode etreMur qui renvoie le boolean à la position [x][y] du tableau.

La classe Jeu a été la première vraie difficulté de ce projet. Nous avons décidé de rajouter une méthode getChar qui permet de faire le lien entre un caractère rencontré plus tard lors du chargement du jeu dans une String et ce qu'il représente. Nous avons par ailleurs rajouter un caractère spécial pour une caisse posée sur un dépot (ce qui n'était pas défini dans les constantes). Cette méthode getChar nous a permi une création plus simple de la méthode jeuToString car nous pouvions à l'aide d'un StringBuilder faire le lien entre l'état du jeu et l'état de la String attendue.
Pour la méthode deplacerPerso, la difficulté a été de comprendre comment bien décomposer le problème et les différents cas. Pour nous aider nous avons rajouter la méthode getSuivant qui permet d'obtenir les coordonées du personnage après un déplacement (cela nous permet de gérer séparement le comportement du personnage en fonction des actions et la réaction du déplacement au sein du jeu)
La méthode etreFini a essentiellement consisté en une double boucle qui pour chaque caisse vérifie si ces coordonnées correspondent à un dépôt

La classe Chargement avec sa méthode chargerJeu était également une difficulté de comprendre l'algorithme que de l'implémenter. Bien qu'il fallait là aussi décomposer le cheminement de la méthode, cela a été plus compliqué que deplacerPerso. Il nous a fallu plusieurs tentatives avant de bien comprendre toutes les étapes nécessaires pour faire fonctionner la méthode chargerJeu, nous y sommes cependant parvenus.

La classe MainJeu n'a pas posée de soucis en tant que telle, cependant il nous a fallu au moins une heure pour comprendre qu'elle n'était pas sensé figurer dans le package Jeu, ce qui nous a bloqué pendant un certain temps.

Les deux classes de Tests étaient plus ou moins facile a écrire. La seule difficulté a été de pouvoir faire que les classes localisent les fichiers. Pour une raison que nous ignorons encore, pour les classes de tests si nous écrivions les chemins vers les fichiers, les classes étaient incapables de le retrouver c'est pourquoi nous avons du procéder à une recherche par URL.