Itération 3 groupe 1

Exercice7.28.1 :

création de la commande ***test*** acceptant un second mot qui représente un nom de fichier, et exécute toutes les commandes lues dans ce fichier texte, les classes à modifier sont « CommandWord » et « GameEngine » :

```
230     private void test(Command command){
231         if (!command.hasSecondWord()) {
232             // if there is no second word, we don't know where to go...
233             gui.println("Whate is the name of the file ?");
234             return;
235         }
236         try
237         {
238           String nameFile = "test.txt";
239             nameFile = command.getSecondWord();
240             Scanner sc = new Scanner(new File(nameFile));
241             while (sc.hasNext()) {
242                 String line = sc.nextLine();
243                 //gui.println(line);
244                 interpretCommand(line);
245             }
246             sc.close();
247         }
248         catch (FileNotFoundException ex)
249         {
250             gui.println("File not fond.");
251             ex.printStackTrace();
252         }
253     }
```

Apprentissage lecture de fichiers de text, hasNext(), next(), exceptions : Fait.

https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html
https://docs.oracle.com/javase/8/docs/api/java/io/File.html

Exercice 7.28.2:

exercice fait par rapport au scénario développé dans ce jeu.

Exercice 7.29:

L'ajout de la classe **Player** qui contient un nom, le poid maximum qu'il supporte, le poid qu'il porte actuellement, un gain et les objets que le joueur a ramassé ou mangé.

```java
public class Player {
    private String name;
    private int weightMax;
    private int weight;
    private int gain;
    private ItemList items;

    public Player(String name) {
        this.name = name;
        this.weightMax = 60;
        this.weight = 0;
        this.gain = 0;
        this.items = new ItemList();
    }
```

Exercice 7.30:

L'ajout de deux commandes take et drop qui s'occupe de la prise ou du dépôt d'un objet déjà prit respectivement, les classes modifiées sont « CommandWord » et « GameEngine » :

```java
private void take(Command command) {
    if (!command.hasSecondWord()) {
        gui.println("vous voullez manger quoi?");
        return;
    }

    String ob = command.getSecondWord();

    // Try to leave current room.
    Objet objet = currentRoom.getObjet(ob);

    if (objet == null) {
        gui.println("tu vas rien manger!");
    } else if(currentPlayer.addObjets(objet)) {
        currentRoom.deleteObjetRoom(ob);
        gui.println("take :");
        gui.println(objet.toString());
    }else{
        gui.println("weight Exceed! choose an other object.");
    }
}
```

```java
private void drop(Command command) {
    if (!command.hasSecondWord()) {
        gui.println("vous voullez manger quoi?");
        return;
    }

    String ob = command.getSecondWord();

    // Try to leave current room.
    Objet objet = currentPlayer.getObjet(ob);

    if (objet == null) {
        gui.println("tu vas rien manger!");
    } else{
        currentRoom.addObjetsRoom(objet);
        currentPlayer.deleteObjetPlayer(ob);
        gui.println("drop :");
        gui.println(objet.toString());
    }
}
```

Exercice 7.31 & exercice 7.31.1::

porter items fait.

Création de la classe ItemList qui gére une liste d'items et mutualiser la gestion des items qui ont été dupliqué dans Room et dans Player.

```java
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

public class ItemList{
    private HashMap<String, Objet> objets;
    private int nombreObjets;
    public ItemList(){
        this.objets = new HashMap<String, Objet>();
        this.nombreObjets = 0;
    }

    public Objet getObjet(String ob) {
        return objets.get(ob);
    }

    public void createObjets()
    {
        int min = 2;
        int max = 5;
        nombreObjets = min + (int)(Math.random() * ((max - min) + 1));// entre 2 à 5
        for(int  i = 0;i < nombreObjets ; i++){
            Objet obj = new Objet();
            objets.put(""+i, obj);
        }
    }
```

```java
    public String lookObjets(){
        String strObjets = "Nombre de objets: " + nombreObjets + "\n";
        for(Map.Entry<String, Objet> entry : objets.entrySet()){
            String key = entry.getKey();
            Objet obj = entry.getValue();
            strObjets += key;
            strObjets += obj;
        }
        return strObjets;
    }

    public void deleteObjet(String ob){
        objets.remove(ob);
        --nombreObjets;
    }

    public void addObjet(Objet obj){
        objets.put(""+nombreObjets,obj);
        nombreObjets++;
    }

    public boolean hasCles(){
        int i = 0;
        for(Map.Entry<String, Objet> entry : objets.entrySet()){
            Objet obj = entry.getValue();
            if(obj.getDescription().equals("Clé de la chambre à trésor"))
                i++;
        }
        if (i == 5)
            return true;
        return false;
    }
    public boolean magic(Objet ob){
        if(ob.getDescription().equals("Un cookie"))
            return true;
        return false;
    }
}
```

Exercice 7.32 & exercice 7.33 :

Poid max et inventaire c'est fait dans dans la classe Player et GameEngine avec une nouvelle commande ***items.***

```
299     private void items() {
300         gui.println(currentPlayer.lookObjetsPlayer());
301     }
```

```
43      public String lookObjetsPlayer(){
44          return items.lookObjets();
45      }
```

Et cette dernière appelle la méthode lookObjets dans la classe ItemsList.

Exercice 7.34:

l'ajout d'un objet magic cookie qui augmente le poid maximum d'un joueur, qui appelle la méthode magic de la classe ItemsList.

```
16      public boolean addObjets(Objet ob){
17          if(!weightExceed(ob)){
18              this.items.addObjet(ob);
19              if(items.magic(ob)){
20                  this.weightMax += 10;
21              }
22              this.weight += ob.getWeight();
23              this.gain += ob.getGain();
24              System.out.println(this.weight);
25              System.out.println(this.weightMax);
26              return true;
27          }
28          int w = this.weight + ob.getWeight();
29          System.out.println("ta depasser"+w);
30          return false;
31      }
```

Apprentisage de enum et values : fait :

https://docs.oracle.com/javase/8/docs/api/java/lang/Enum.html

Exercice 7.35 & exercice 7.35.1:

Consultation du code source du projet zuul-with-enums-v1 pour voir comment le type CommandWord est utilisé. Les classes Command, CommandWords, Game, et Parser sont toutes adaptées dans cette version de notre projet, pour prendre en charge ce changement. le programme fonctionne toujours comme prévu.

Exercice 7.35.2:

En s'inspirant de ***processCommand()*** de la classe ***Game*** dans zuul-withenums-v1, on a remplacé dans ***interpreteCommand()*** de la classe ***GameEngine*** la suite de if else par un ***switch*** (autorisé sur un type énuméré !) .

```
146             switch(commandWord){
147                 case HELP:
148                     printHelp();
149                     break;
150                 case LOOK:
151                     look();
152                     break;
153                 case EAT:
154                     eat(command);
155                     break;
156                 case BACK:
157                     back(command);
158                     break;
159                 case GO:
160                     goRoom(command);
161                     break;
162                 case TEST:
163                     test(command);
164                     break;
165                 case TAKE:
166                     take(command);
167                     break;
168                 case DROP:
169                     drop(command);
170                     break;
171                 case ITEMS:
172                     items();
173                     break;
174                 case QUIT:
175                     wantToQuit = quit(command);
176                     break;
177                 default : {
178                     gui.println("I don't know what you mean...");
179                     return false;
180                 }
181             }
```

Exercice 7.36, exercice 7.37, exercice 7.38, exercice 7.39, exercice 7.40 et exercice 7.41:

déjà fait.