Donjon

L'objectif est la réalisation d'un « jeu de donjon » simple. Dans ce jeu le joueur parcourt les différentes salles d'un donjon. Lorsqu'il arrive dans une salle le joueur peut choisir une action à exécuter parmi la liste des actions possibles à ce moment là dans cette salle. Une salle peut contenir des objets et être habitée par un ou plusieurs monstres. Les objets peuvent être utilisés par le joueur. Si le joueur rencontre un monstre, il faudra alors le tuer pour pouvoir quitter la salle qu'il habite. Evidemmment le monstre peut riposter s'il est attaqué. L'objectif du joueur est d'atteindre (vivant) la sortie du donjon, une salle particulière. À l'inverse le joueur perd le jeu s'il meurt avant d'atteindre la sortie.
L'objectif de ce projet est la réalisation de ce jeu conformément au cahier des charges précisés ci-dessous.

Cahier des charges

Les personnages
Dans le jeu les personnages sont les monstres ou le joueur. Un personnage est caractérisé par son nombre de points de vie, sa force et le nombre de pièces d'or qu'il porte. Tous les personnages peuvent attaquer un autre personnage. La gestion des combats est simpliste : le joueur peut attaquer un monstre et, s'il n'est pas mort, le monstre attaqué riposte alors en attaquant à son tour le joueur. Le résultat d'une attaque se gère très simplement : le personnage attaqué perd autant de points de vie que les points de force de l'attaquant.
En dehors de la riposte après l'attaque du joueur, les monstres n'ont pas de comportement (il n'attaque pas le joueur spontanément). Lorsqu'un monstre meurt il laisse tomber sur le sol de la salle une bourse qui contient les pièces d'or qu'il possédait.
À la différence des monstres qui attendent passivement dans une salle de se faire taper dessus, le joueur a la capacité d'agir. C'est-à-dire qu'il doit choisir une action parmi les actions possibles au moment de ce choix et l'exécuter. Le joueur peut ainsi se déplacer, regarder autour de lui, attaquer, utiliser des objets, etc.
D'une manière générale,lorsque le joueur est amené à faire une choix, seuls ceux effectivement possibles au moment du choix lui sont proposés. Ce doit être en particulier le cas pour les actions.

Les actions
Les actions que peut exécuter le joueur peuvent n'être possibles que sous certaines conditions.
Voici une rapide description de ces actions :

- *regarder* : cette action est toujours possible et permet au joueur d'obtenir une description sur la salle où il se trouve, les issues, les monstres ou les objets qu'elle contient.
- *se déplacer* : cette action n'est possible que s'il n'y a pas de monstre (vivant) dans la salle, elle consiste pour le joueur à choisir une direction à emprunter pour passer à la salle suivante.
- *combattre* : cette action n'est possible que s'il y a au moins un monstre (vivant) dans la salle, le joueur doit alors choisir le monstre à attaquer.
- *utiliser* : cette action n'est possible que s'il y a au moins un objet dans la salle, le joueur peut alors choisir l'objet à utiliser parmi les objets présents. Pour simplifier, cette actions consiste à la fois à ramasser et à utiliser (immédiatement) l'objet. Il n'y a donc pas d'inventaire à gérer au niveau du joueur.

D'autres actions doivent pouvoir être envisagées pour enrichir les possibilités du jeu (par exemple « se reposer » pourrait permettre au joueur de récupérer des points de vie, sous réserve qu'il n'y ait pas de monstre dans la salle).

Les objets
Différents objets peuvent être trouvés dans les salles et utilisés. Tous les objets sont à usage unique, ils disparaissent une fois utilisés. Il s'agit par exemple de :

- *bourse d'or*, l'utiliser permet d'augmenter son nombre de pièces d'or,
- *potion de soin*, l'utiliser permet d'augmenter son nombre de points de vie,
- *potion de force*, l'utiliser permet d'augmenter son nombre de points de force,
- « *bandit manchot* » (*one-armed bandit*), le joueur doit donner un certain nombre de pièces d'or fixé à l'avance pour l'utiliser. Si le joueur n'a pas assez d'or, le bandit manchot disparaît sans que rien ne se passe. Dans le cas contraire, le bandit manchot produit aléatoirement un autre objet que le joueur utilise dès son obtention.

L'ajout et la prise en compte d'autres objets doit être possible.

Le donjon
Comme déjà indiqué, l'environnement dans lequel évolue le joueur est appelé *donjon*. Un donjon est constitué de salles reliées par des couloirs. Ces couloirs sont identifiés par des directions que le joueur doit indiquer pour emprunter les couloirs. Toutes les directions ne sont pas forcément disponibles dans toutes les salles.
Les personnages se trouvent toujours dans une salle, la notion de couloir n'existe que pour permettre les transitions et les couloirs n'ont donc pas d'existence propre. Les déplacements sont donc supposés « instantanés » d'une salle à sa voisine.
Une salle peut contenir des objets et être habitée ou non par un ou plusieurs monstres. Certaines salles (au moins une par donjon) sont particulières, elles représentent des sorties du donjon. Les atteindre permet au joueur de gagner le jeu.

Réalisation

On suppose que la classe permettant de représenter un jeu est de la forme :

AdventureGame
- currentRoom : Room - player : Player
+ AdventureGame(startingRoom : Room) + currentRoom() : Room + getPlayer() : Player + play(player : Player) + addMonster(monster : Monster, room : Room) + addItem(item : Item, room : Room) + isFinished() : boolean + playerMoveTo(direction : Direction)

La méthode `play` permet de démarrer le jeu avec le joueur passé en paramètre.
Vous pouvez être amené à compléter cette classe en fonction de l'analyse menée par la suite.

Q 1 . Pour représenter le joueur (*game player*) un programmeur fait la proposition de code suivante :

```java
package adventure.entities;
import java.util.Scanner;
import adventure.AdventureGame;

/** This class represents players in the adventure game */
public class BadDesignPlayer extends Character {

  protected AdventureGame game;
  public BadDesignPlayer() {  }

  // ... CODE HERE ...

  /**
   * the player acts in the game
   */
  public void act() {
    Scanner in = new Scanner(System.in);
    System.out.println("what do you want to do ?");
    System.out.println(" - move ");
    System.out.println(" - attack ");
    System.out.println(" - look ");
    String choice = in.nextLine();
    in.close();

    switch (choice) {
    case "move":
      if (this.canMove()) {
        System.out.println(" player moves...");
        // code for move here ...
      } else {
        System.out.println(" action not possible...");
      }

      break;
    case "attack":
      if (this.canAttack()) {
        System.out.println(" player attacks...");
        // code for attack, here ...
      } else {
        System.out.println(" action not possible...");
      }
```

```
        break;
      case "look":
        // to look is always possible
        System.out.println(" player looks around...");
        // code for look here ...
        break;
      default:
        System.out.println(" i don't understand...");
      }
    }

    // ============================================================
    public static void main(String args[]) {
      BadDesignPlayer player = new BadDesignPlayer();
      while (! player.game.currentRoom().isExit()) {
        player.act();
      }
    }

}
```

Critiquez cette proposition.

Q 2 . Sous la forme de diagrammes UML, faites une (rapide) proposition pour représenter les différentes entités du jeu :

- les salles (*room*) et leurs liens entre elles,
- les personnages (*game character*),
- les objets (*item*).

Q 3 . Faites une proposition pour mieux gérer les actions du joueur. Donnez ensuite un code pour la méthode `act` du joueur et pour les actions d'attaque ou de déplacement.

Q 4 . Que ce soit le choix de l'action à effectuer, de la direction à prendre, du monstre à attaquer, le joueur doit à de nombreuses occasions réaliser un choix parmi une liste de possibilités.

Comment gérer « proprement » ces différentes situations de choix par le joueur ?

C'est-à-dire en évitant la duplication de code et en permettant la gestion d'autres choix (comme celui de l'objet à utiliser).