Basket-Ball Training + lien page web

(A3P(AL) 2014/2015 Gα)

I.A) Auteur(s)

-POISSON Valentin, Etudiant à ESIEE Paris

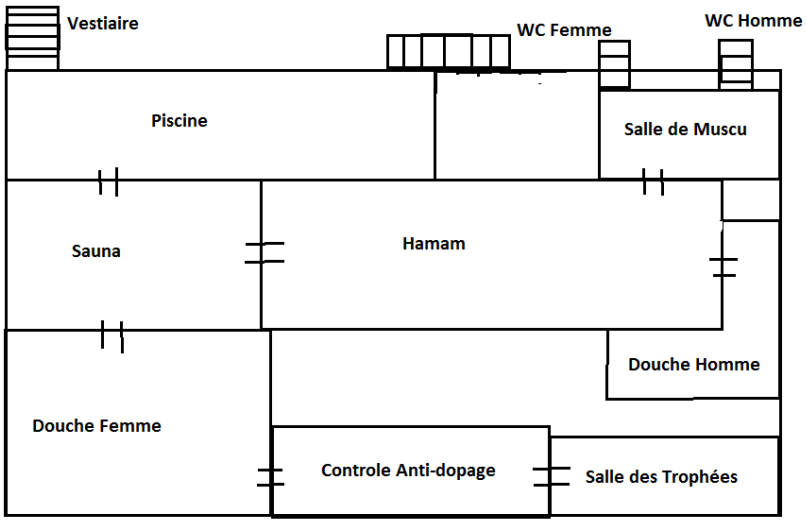
I.B) Thème (phrase-thème validée)

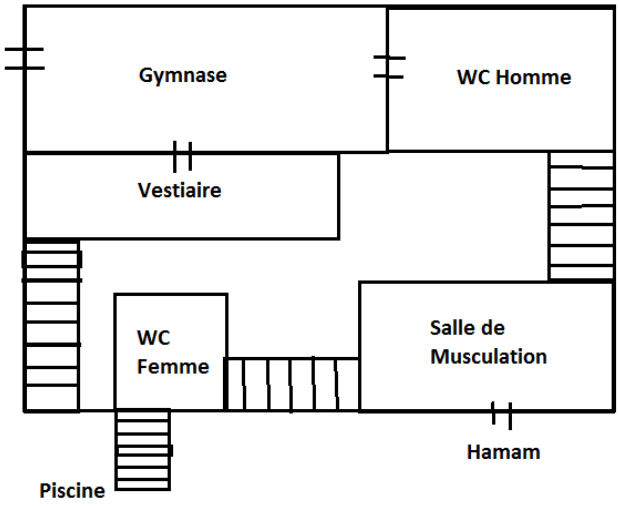
-Un basketteur doit retrouver toutes ses affaires avant que le match ne commence.

I.C) Résumé du scénario (complet)

-Il devra trouver des affaires pour le match, par exemple une balle, des chaussure,… des outils, et son objet (trophée) pour finir. L'objectif sera de trouver le trophée.

I.D) Plan





I.E) Scénario détaillé (complet, avec indication de la partie "réduit" si exercice 7.3.3)

- -Nous somme le 11 Juin 2015, et se déroule aujourd'hui le plus grand match de l'histoire de la NBA. Il se déroule à Chicago aux Etats Unis. Vous devez participer à ce très grand match et toute votre équipe vous attend ! Vous venez d'arriver dans le gymnase et le Coach vous demande où sont vos affaires car il ne les a pas trouvé dans le sac avec celle de vos coéquipiers… Avant de pouvoir commencer votre match, il vous faut récupérer toutes vos affaires nécessaires pour votre match, dans les différents lieux du Centre Sportif. Le temps sera bien évidemment compté.

I.F) Détail des lieux, items, personnages

- Le jeux se déroule au états unis dans le fameux gymnase des Chicago Bulls. Ce gymnase est composé de 12 salles : Gymnase, WC homme et Femme, salle de muscu, vestiaire, hamam, douche Homme et Femme, Sauna, Piscine, Salle des trophées, salle de contrôle anti dopage.

Dans chaque lieu le joueur pourra rencontrer des coéquipiers ou d'autre intervenant qui pourront l'aider à retrouver ses affaires.

I.G) Situations gagnantes et perdantes

- Situation Gagnantes : le joueur aura gagner lorsqu'il sera prêt pour son match et qu'il aura également trouvé le trophée.

-Situation perdantes : le joueur n'aura pas réussis à rassembler toutes ses affaires dans le temps impartie.

Exercices

7.5 :

```
public void printLocationInfo()
  {
          System.out.println("Vous êtes " + aCurrentRoom.getDescription());
          System.out.println("Les sorties :");
           if(aCurrentRoom.aNorthExit != null) System.out.println("North");
           if(aCurrentRoom.aSouthExit != null) System.out.println("South");
           if(aCurrentRoom.aEastExit != null) System.out.println("East");
           if(aCurrentRoom.aWestExit != null) System.out.println("West");
  }
```

7.6 :

```
public Room getExits(String pDirection)
  {

          if (direction.equals("North")) return aNorthExit;
          if (direction.equals("South")) return aSouthExit;
          if (direction.equals("East")) return aEastExit;
          if (direction.equals("West")) return aWestExit;
  }
```

7.7 :

```
public String getExitString()
  {
           String vSorties="";
           System.out.println("Les sorties :");
           if(this.aNorthExit != null) vSorties=vSorties+" North";
           if(this.aSouthExit != null) vSorties=vSorties+" South";
           if(this.aEastExit != null) vSorties=vSorties+" East";
           if(this.aWestExit != null) vSorties=vSorties+" West";
```

```
        if(this.aDownExit != null) vSorties=vSorties+" Down";
        return vSorties;
}
```

7.8 :

```
vGymnase.setExit("South",vVestiaire);

    vVestiaire.setExit("North",vGymnase);

    vPiscine.setExit("Down",vVestiaire);

    vSauna.setExit("North",vPiscine);

    vDoucheFemme.setExit("North",vSauna);

    vHamam.setExit( "North",vSalleDeMuscu);

    vControle.setExit("East",vSalleDesTrophees);

    vSalleDesTrophees.setExit("West",vControle);

    vDoucheHomme.setExit("West", vHamam);

    vWcFemme.setExit("Up", vSalleDeMuscu);

    vWcHomme.setExit("West", vGymnase);

    vSalleDeMuscu.setExit("Down",vWcHomme);


    vGymnase.setExit("East",vWcHomme);

    vVestiaire.setExit("Up", vPiscine);

    vPiscine.setExit("Down", vWcFemme);

    vSauna.setExit("East", vHamam);

    vDoucheFemme.setExit("East", vControle);

    vHamam.setExit("East", vDoucheHomme);

    vControle.setExit( "West", vDoucheFemme);

    vWcFemme.setExit("Up", vPiscine);

    vWcHomme.setExit("Up", vSalleDeMuscu);

    vSalleDeMuscu.setExit( "Down", vWcFemme);
```

```
        vPiscine.setExit("South", vSauna);

        vSauna.setExit("South", vDoucheFemme);

        vHamam.setExit( "West", vSauna);

        vSalleDeMuscu.setExit( "South", vHamam);
```

```
import java.util.HashMap;
public class Room
{
...
public void setExit(final String pDirection, final Room pNeighbor)
{
this.aExits.put(pDirection, pNeighbor);
}
...
} // class Room
```

7.8.1 :

```
public class Game
{
...
private void createRooms()
{
...
vPlaceMousquetaire.setExit("Down", vParking);...
vParking.setExit("Up", vPlaceMousquetaire);...
}
...
}
// class Game
```

7.9: La HashMap<String, Room> est une collection d'objets qui associe une clé (String) à une valeur (Room).

7.10 : Fonctionnement de getExitString() :

```
public String getExitString()
  {
     String vReturnString = "Exits: ";
     Set <String> keys =aExits.keySet();
     for(String aExit: keys)
     {
        vReturnString += " " + aExit;
     }
```

```
    return vReturnString;
}
```

-HashMap: Une HashMap est un dictionnaire, ou tableau d'association qui associe une clé à une valeur correspondante.

-Set: Un ensemble a une différence essentielle avec une liste : il ne peut pas comporter deux fois le même élément.

-keySet(): fonction qui parcourt la HashMap et retourne l'ensemble des clés de la hashmap

7.10.1:

7.11 :

```
public String getLongDescription()
  {
    return "You are " + this.aDescription + ".\n" + getExitString();
  }
} // Room
```

7.14 :

```
public class Game
{
...
private boolean processCommand(Command pCommand)
  {
    boolean vQuit = false;
    else if (pCommand.getCommandWord().equals("look"))
    {
       this.look() ;
return false;



public class CommandWords
{
...
private static final String[] sValidCommands = {
    "go", "quit", "help", "look", "shoot"
}// class CommandWords
```

7.15 :

```
public class Game
{
...
else if (pCommand.getCommandWord().equals("shoot"))
      {
         this.shoot();
         return false;
      }
...
}
...
}
private void shoot()
{
System.out.println(“You have to shoot for getting your clothes.”);
}
...
}
// class Game



public class CommandWords
{
...
private static final String[] sValidCommands = {
      "go", "quit", "help", "look", "shoot"
  } // class CommandWords
```

7.16 :

```
public class Game
{
...
private void printHelp()
{
System.out.println("You are lost. You are a
lone." + "\n" + “Your commands are : “);
this.aParser.showCommands();
}
...
}
// class Game



public class Parser
```

```
{
...
public void showCommands()
  {
    this.aValidCommands.showAll();
  }
} // class Parser



public class CommandWords
{
public void showAll()
  {
    for(String vCommand : sValidCommands)
    { System.out.print(vCommand + " "); }
    System.out.println();
  }//showAll()
} // CommandWords
```

A savoir expliquer :

-for (typeElement vElement : tableau): La boucle for each parcours tous les éléments du tableau, et pour chaque élément de ce même tableau, elle applique l'instruction donnée par le programmeur.

-for (typeElement vElement: ensemble): La boucle for each d'un ensemble agit de la même manière que la boucle for each d'un tableau, ce pendant l'ensemble est remplacé par un ensemble, il peut s'agir d'une HashMap par exemple.