Dans le jeu, vous débuterez la partie en Martinique, et vous contrôlerez un homme qui cherche sa planche de wakeboard pour aller à sa compétition. Cependant, le temps est limité pour récupérer la planche et vous rendre à la compétition.

Table des matières :

I. Informations sur le jeu :

Auteur et Thème :

Je m'appelle Sainte-Rose Clément et je suis en classe 12 promo B.
En Martinique, un jeune wakeboardeur a perdu sa planche 5 jours avant sa compétition.

Résumé du scénario :

Vous débuterez donc la partie en Martinique, et vous contrôlerez un jeune homme de 17ans s'appelant Victor qui cherche sa planche de wakeboard pour aller à sa compétition. Cependant, vous n'avez que 3 jours soit 3 minutes de jeu pour récupérer la planche et vous rendre à la compétition.

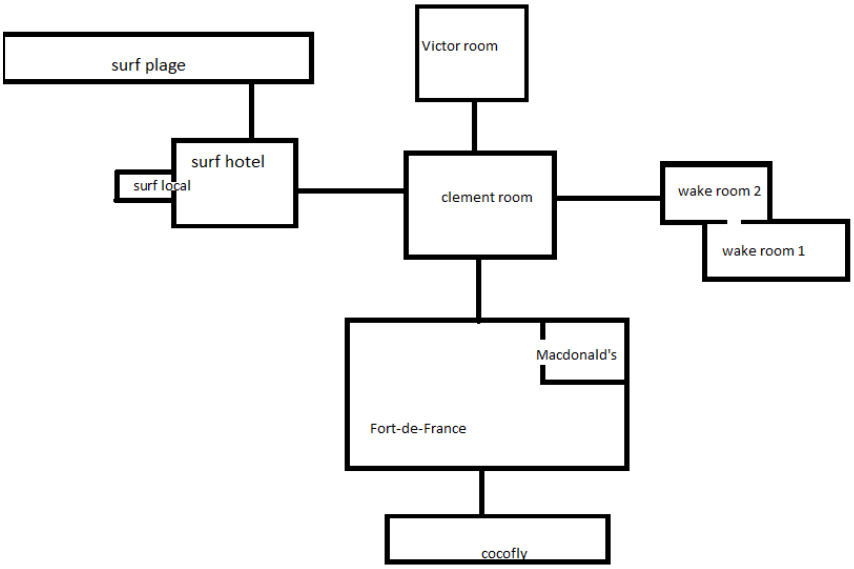
Scénario détaillé :

Aujourd'hui, jeudi matin, Victor est en vacances depuis déjà 1semaine et à une compétition samedi après-midi. Il décide de partir s'entrainer aux Trois-Ilets (emplacement de son club), cependant il ne trouve pas sa planche et a 3jours (3minutes) pour la retrouver. Il part donc avec son argent de poche fouiller les endroits où il a pu l'utiliser. En raison du fort prix de cette dernière, la mère de Victor ne doit pas l'apprendre, il devra donc rentrer tous les soirs avant 18h (10 dernières secondes) sous peine d'avoir 30s de jeu de moins (2 soirs de retard=défaite). De nombreux autres malus et bonus sont applicables : bonne chance à vous !

Différentes rooms et items :

Rooms :

1) Victor home
2) Clément home
3) Surf hotel
4) Surf black beach
5) Local surf
6) Wake room 1
7) Wake room 2
8) Cocofly
9) MacDonald's
10) Fort-de-France



Items :

-argents de poche (acheter des MacDo).

-jeton de défi (permet de faire tourner une roue de bonus (+ 15s, passe un coup de fil et sais si il y a la planche dans la classe choisie (2 par parties), recharge la barre de vie 100%) et de malus (déplacement ralentit, - 30s, perte de 50%de vie) 40s entre l'apparition et les réapparitions.

- planche (items gagnant).
- Happy meal (remplit de 40% les HP).

Situations gagnantes et perdantes :

Situations gagnantes :

- Victor retrouve sa planche et la ramène.

Situations perdantes :

- Victor ne ramène pas la planche dans le temps imparti.
- Victor n'est pas rentré à 00:55 et 01:55 (couvre-feu de 18h).
- Victor n'a pas rechargé sa barre de vie.

II. Réponses aux exercices :

Exercice n° 7.5 printLocationInfo :

La méthode printLocationInfo est dans la classe Game et elle permet d'éviter la duplication de code.
Elle affiche les différentes sorties et la position du joueur.
Il faut aussi changer le code des classes goRoom et printWelcome car grâce à la méthode printLocationInfo les méthodes printWelcome et goRoom pourront afficher les informations sur la situation.

Le code de la méthode printLocationInfo est :

```
private void printLocationInfo(){
        System.out.println("You are " + this.aCurrentRoom.getDescription()); System.out.print("Exits: ");
        if (this.aCurrentRoom.aNorthExit != null) System.out.print("north ");
        if (this.aCurrentRoom.aEastExit != null) System.out.print(" east ");
        if (this.aCurrentRoom.aSouthExit != null) System.out.print(" south ");
        if (this.aCurrentRoom.aWestExit != null) System.out.print(" west ");
        System.out.println() ;
}
```

Exercice 7.6 getExit ()

Le guetteur getExit () a pour fonction de diminuer le niveau de couplage dans la classe Room.

```
public Room getExit (String pdirection) {
                if (pdirection.equals(« north »)) {
                return aNorthExit ;
        }
                if (pdirection.equals(« east »)) {
                return aEastExit ;
        }
                if (pdirection.equals(« south »)) {
                return aSouthExit ;
        }
                if (pdirection.equals(« west »)) {
                return aWestExit ;
        }
        return null ;
}
```

On s'en sert aussi dans la classe Game en réduisant le code.
Ainsi on passe de ce code :

```
Room vNextRoom = null; {
        if(pdirection.equals("north")) {
        vNextRoom = this.aCurrentRoom. aNorthExit;
        }
        if(pdirection.equals("east")) {
        vNextRoom = this.aCurrentRoom. aEastExit;
        }
        if(pdirection.equals("south")) {
```

```
            vNextRoom = this.aCurrentRoom. aSouthExit;
            }

            if(pdirection.equals("west"))
            vNextRoom = this.aCurrentRoom. aWestExit;
            }
}
```

A celui là :

Room vNextRoom = this.aCurrentRoom.getExit(vDirection);

Exercice 7.7 getExitString() :

Cette méthode qui se situe dans la class Room permet de simplifier la méthode printLocationInfo.

```
Public Room getExitString() {
            String vExits = "Exits : ";
            If (this.aNorthExit != null ){
            vExits +=" north ");
            }
            If (this.aEastExit!= null){
            vExits +=" east ");
            }
            If (this.aSouthExit!= null){
            vExits += "south ");
            }
            If (this.aWestExit!= null){
            vExits +=" west ");
            }
            returrn vExits
            }
}
private void printLocationInfo() {
            System.out.println ("You are "+this.aCurrentRoom.getDescription());
            System.out.print (this.aCurrentRoom.getExitString());
            System.out.println ();
}
```

Exercice 7.8 (HashMap)

Pour faciliter la création des rooms et des directions il faut ajouter une HashMap.
L'HashMap contenant des indices nommés key qui seront dans notre des cas des Strings. Pour l'utiliser il faut importer la java.util.HashMap.

```
public class Room{
        private String aDescription ;
        private HashMap<String, Room> aExits;

        public Room (String pdescription){
        this.aDescription = pdescription;
        aExits= new HashMap<String, Room> ();
        }
        public String getDescription(){
        return this.aDescription;
        }
        public void setExits(final String pDirection,final Room pNeighbor ) {
        aExits.put(pDirection, pNeighbor);
        }
        public Room getExit(final String pDirection) {
        return this.aExits.get(pDirection);
        }
        public String getExitString() {
        String vExits = "Exits : ";
        if(this.aNorthExit != null ){
        vExits +=" north ");
        }
        if(this.aEastExit!= null){
        vExits +=" east ");
        }
        if(this.aSouthExit!= null){
        vExits += "south ");
        }
        if(this.aWestExit!= null){
        vExits +=" west ");
        }
        returrn vExits ;
        }
}
} // Room
```

Exercice 7.9 (keySet)

La méthode getExitString() doit aussi être modifiée :

```
public String getExitString(){
String vExits = "Exits: ";
Set<String> keys = aExits.keySet();
        for(String exit : keys){
```

```
            vExits+=" " +vexit;
}
return vExits;
}
```

Exercice 7.10

Les sorties générées par la méthode getExitString() sont les key d'aExits.
Set est une collection sans ordre avec des éléments sans doublant.
La méthode keySet() crée un ensemble d'objets de type set<> grâce à «java Set<String> vKeys = this.aExits.keySet();»
On stocke toute les key dans la collection Set<String> vKeys.

Exercice 7.11 (getLongDescription)

En ajoutant getLongDescription( ) dans Room on modifie aussi la méthode printLocationInfo().

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

Exercice 7.14 look

Maintenant on doit rajouter la commande « look ».

```
private void look() {
System.out.println(this.aCurrentRoom.getLongDescription());
}
```

On doit ajouter look dans la classe CommandWords aussi comma ca le joueur peut l'utiliser.

```
public CommandWords(){
        this.aValidCommands= new String[5];
        this.aValidCommands[0]= "go";
        this.aValidCommands[1]= "help";
        this.aValidCommands[2]= "quit";
        this.aValidCommands[3]= "look";
        this.aValidCommands[4]= "eat";
    }
```

la méthode processCommand qui exécute les commandes dans la classe game doit aussi être crée.

```
private boolean processCommand(final Command pCommand){
        String vCommand=pCommand.getCommandWord();
        boolean vR=false;
```

```
        if (vCommand==null){
            System.out.println("I don't know what do you mean...");
            return false;
        }
        else if (vCommand.equals("go")){
            this.goRoom(pCommand);
        }
        else if (vCommand.equals("help")){
            this.printHelp();
        }
        else if (vCommand.equals("quit")){
            this.quit(pCommand);
            return true;
        }
        else if (vCommand.equals("look")){
            this.look();
        }
        else if (vCommand.equals("eat")){
            this.eat();
        }
        return vR;
    }
```

Exercice 7.15 (eat)

Pour ajouter eat() , il faut s'inspirer de la méthode de l'exercice 7.14 :

```
private void eat(){
        System.out.println("Vous venez de manger, vous n'avez plus faim");
    }
private boolean processCommand(final Command pCommand){
        String vCommand=pCommand.getCommandWord();
        boolean vR=false;
        if (vCommand==null){
            System.out.println("I don't know what do you mean...");
            return false;
        }
        else if (vCommand.equals("go")){
            this.goRoom(pCommand);
        }
        else if (vCommand.equals("help")){
            this.printHelp();
        }
        else if (vCommand.equals("quit")){
            this.quit(pCommand);
            return true;
```

```
        }
        else if (vCommand.equals("look")){
            this.look();
        }
        else if (vCommand.equals("eat")){
            this.eat();
        }
        return vR;
    }
```

Exercice 7.16

look et eat ne sont proposées dans le menu help. On ajoute donc showAll() pour afficher toutes les commandes sans les ajouter une par une.

```
public void showAll(){
        for (String command:aValidCommands){
            System.out.println(command + " ");
        }
        System.out.println();
    }
```

Il n'y'a pas de couplage entre la class Game et CommandWords la classe Parser fera donc le pont.
Dans Parser :

```
public void showCommands (){
        aValidCommands.showAll();
    }
```

dans game :

```
private void printHelp(){

        System.out.println("you are lost. You are alone."+'₩n'+"You wander around at the university.");

        System.out.println("Vos commandes sont : ");

        aParser.showCommands();

        return;

    }
```