Titre : sauver sa famille du covid

Auteur : Firas Miladi, Groupe 8

Table des matières :

- Présentation du jeu

Auteur :
Je suis Miladi Firas du Groupe 8

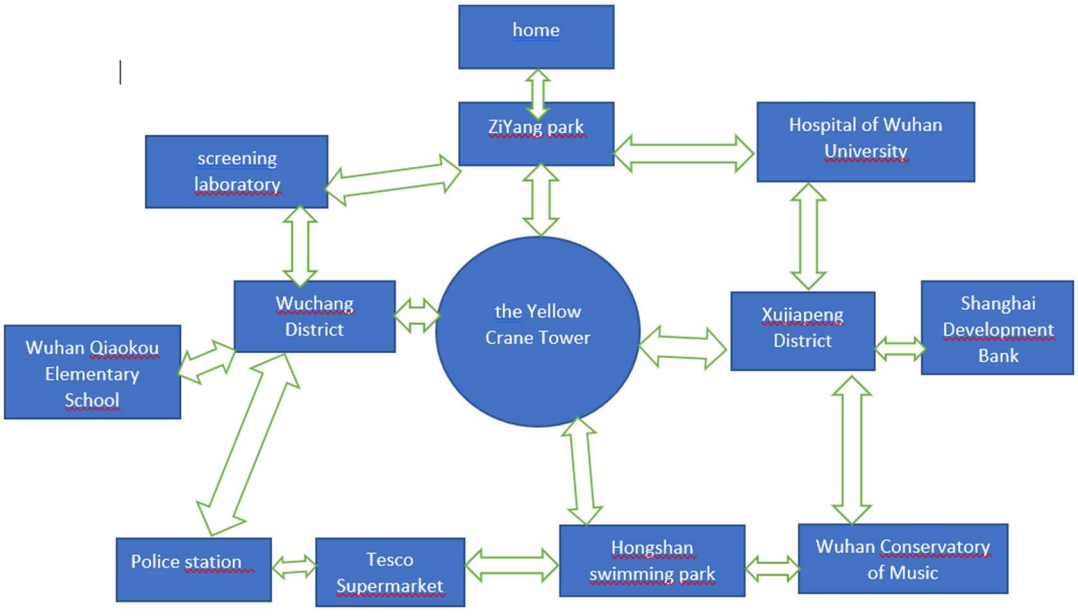
Thème :
A Wuhan Un père cherche à sauver sa famille du virus mortel .

Résumé du scénario :

Dans ce jeu, vous débutez la partie à Wuhan, et vous contrôlez un homme qui doit récupérer sa femme et ses enfants et retournez chez lui pour les sauver du covid .

Arriver chez vous, vous devez aller faire les courses puisqu'un confinement total est annoncé pour le lendemain. De plus, vous n'avez que 30 minutes de jeu pour récupérer ta famille, faire les courses et rentrer chez vous avant le couvre-feu.

Plan



Scénario détaillé

Au cours d'une journée comme toutes les autres, Kong est au travail il pense à faire une surprise à sa famille. Soudain, il aperçoit un mouvement et des bruits inhabituels dans l'entreprise. Les médias viennent de lancer un appel d'urgence a toute la population de regagner leurs domiciles en toute urgence. Un virus mortel vient de se propager dans la ville en faisant déjà des milliers de mort et de contaminés. Kong doit récupérer sa femme Hong qui travaille à l'administration de l'Hospital de Wuhan, sa fille qui suit des cours a l'école de musique et son fils élève à l'école Wuhan Qiaokou. Tout au long de son parcours, il va trouver des objets qu'ils vont l'aider dans sa mission tel que des masques, du gel hydroalcoolique, des gants, des visières…et il va rencontrer des personnes porteuses du virus qu'il doit les éviter. Arriver chez lui, il doit aller faire les courses puisqu'un confinement total est annoncé. De plus, il n'a que 30 minutes de jeu pour récupérer sa famille, faire les courses et rentrer chez lui avant le couvre-feu.

Détail des lieux, items, personnages
Personnage Kong banquier le père Hong la mère
13 lieux

1/home

2/Ziyang park

3/screening laboratory

4/Hospital of Wuhan University

5/the Yellow Crane Tower

6/Xujiapeng District

7/Shanghai Development Bank

8/Wuchang District

9/Wuhan Qiaokou Elementary School

10/Police station

11/Tesco Supermarket

12/ Hongshan swimming park

13/ Wuhan Conservatory of Music

Items:
-un téléphone
-des masques
-du gel hydroalcoolique

- des gants
- des visières

Situations gagnantes et perdantes

Il doit sauver sa famille en rentrant chez lui sans que quelqu'un d'entre eux n'attrape le covid et faire les courses en respectant le temps imposé le couvre-feu.
Si l'un des membres de sa famille attrape le covid, ou si vous vous faites embarquer par la police (ne respecte pas le couvre-feu) la partie est perdue.

Eventuellement énigmes, mini-jeux, combats, etc.

Le père doit éviter les personnes infectées et notamment les affronter

Réponses aux exercices :

Le rôle de chaque classe est :

La class Game : cette class c'est la classe principale de du programme notamment avec la méthode play() qui permet de commencer le jeu. Cette class permet de lire et d'exécuter les commandes.

La class Parser : cette classe lit le terminal où l'utilisateur tape ses commandes. Elle essaye d'interpréter les mots comme des commandes, et retourne des objets de la classe Command.

La class CommandWord : cette class définie les commandes valides pour le jeu

La class Command : cette class produit une commande de l'utilisateur

La class Room : cette class créer les lieux du jeu, et les connexions entre eux

Ex 7.5 printLocationInfo :

La méthode printLocationInfo a été créée dans la classe Game pour éviter la duplication de code, comme indiquée dans le livre. Elle affiche la position actuelle du joueur et également les différentes sorties.

Quand on modifie une partie du code, il faut apporter plusieurs changements dans le code pour éviter les incohérences notamment, les méthodes goRoom et printWelcome.

Après la création de printLocationInfo, Les méthodes printWelcome et goRoom pourront donc appeler la méthode printLocationInfo quand elles auront besoin d'afficher des informations sur la situation pour faciliter les futures modifications qui pourraient arriver,

```
private void printLocationInfo(){
   System.out.println("You are " + this.aCurrentRoom.getDescription());
   System.out.print("Exits: ");
   if(this.aCurrentRoom.northExit != null)
      System.out.print("north ");
   if(this.aCurrentRoom.eastExit != null)
      System.out.print(" east ");
   if(this.aCurrentRoom.southExit != null)
      System.out.print(" south ");
   if(this.aCurrentRoom.westExit != null)
      System.out.print(" west ");
   System.out.println();}
```

Ex 7.6 getExit()

Le guetteur getExit() de la classe Room permet de réduire le niveau de couplage

```
public Room getExit(String direction){
      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 ;}
```

Ce guetteur va être utiliser dans la classe Game (méthode goRoom()) en rendant le code beaucoup plus court
On passe de ce code :

```
Room vNextRoom = null;
if(vDirection.equals("north")) { vNextRoom = this.aCurrentRoom.
aNorthExit;}

if(vDirection.equals("east")) { vNextRoom = this.aCurrentRoom.
aEastExit;}

if(vDirection.equals("south")) { vNextRoom = this.aCurrentRoom.
aSouthExit;}

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

a cela :

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

Ex 7.7 getExitString() :

Cette méthode se situe dans la class Room. Son rôle est de simplifier la méthode printLocationInfo. Cette méthode va simplifier les modifications futures du code.

```
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();
   }
```

Ex7.8 (HashMap)

Dans cet exercice on doit intégrer une HashMap pour faciliter la création des rooms et les directions

Le HashMap est un tableau ou les indices ne sont pas des entiers, mais des objets que l'on nommera "key" (ici les keys sont les String).

Pour utiliser ce paquetage, il faut importer la java.util.HashMap;

```
import 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
```

Ex 7.9 (keySet)
Il faut modifier également la méthode getExitString()

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

Ex 7.10

Les directions de sortie générées par la méthode getExitString() sont les "key" de la table de hachage aExits.

L'interface Set est une collection d'objets sans ordre avec des éléments uniques sans doublant.

La méthode keySet() créer un ensemble d' objet de type set<>

En faisant «java Set<String> vKeys = this.aExits.keySet();»
On stock toute les "key" (direction de sortie de l'objet courant) dans la collection de types Set<String> vKeys.

Ex 7.11 (getLongDescription)

on ajoute la méthode getLongDescription()dans la classe Room ,puis on modifie la méthode printLocationInfo().

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

Ex 7.14 look
Dans cet exercice, il faut ajouter une nouvelle commande « look ».

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

Le problème c'est si on ajoute look seulement dans la class game l'utilisateur ne peut jamais l'utiliser car elle n'est pas connu donc on doit l'ajouter dans la class CommandWords

```
private static final String[] aValidCommands=
{"go","help","quit","look","eat"};
```

et egalement la méthode processCommand qui exécute les commandes dans la class game

```
private boolean processCommand  (final  Command pCom){
```

```
        boolean vReturn = false ;
        if(pCom.isUnknown())
        {
            System.out.println("I don't know what you mean...");
            return false;
        }
        //the command is unknown

        String vcomtostring = pCom.getCommandWord();

        if(vcomtostring.equals("quit"))
        {
            vReturn=true;
        }
        else if(vcomtostring.equals("help"))
        {
            printHelp();
        }
        else if(vcomtostring.equals("go"))
        {
            goRoom (pCom);
        }else if(vcomtostring.equals("look"))
        {
            look();
        }

        return vReturn;
        // execute the commands
    }
```

Exercice 7.15 (eat)

Pour ajouter la commande eat() , il faut faire les mêmes étapes que l'exercice 7.14

```
private void eat()
    {
        System.out.println("You have eaten now and you are not hungry
any more.");
    }
```

```
private boolean processCommand  (final  Command pCom){
        boolean vReturn = false ;
        if(pCom.isUnknown())
        {
            System.out.println("I don't know what you mean...");
            return false;
        }
        //the command is unknown

        String vcomtostring = pCom.getCommandWord();

        if(vcomtostring.equals("quit"))
        {
            vReturn=true;
        }
        else if(vcomtostring.equals("help"))
        {
            printHelp();
        }
        else if(vcomtostring.equals("go"))
        {
            goRoom (pCom);
        }else if(vcomtostring.equals("look"))
        {
            look();
        }
        else if(vcomtostring.equals("eat"))
        {
            eat();
        }

        return vReturn;
        // execute the commands
    }
```

Exercice 7.16

Si on utilise la méthode help on se rend compte que les commandes look et eat ne font pas partie des commandes proposées

Pour régler ce problème on doit ajouter une méthode showAll() pour afficher toutes les commandes sans avoir besoin de les ajouter manuellement une par une .

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

Et puisqu'il n'y a pas de couplage entre la class Game et CommandWords on doit passer de CommandWords a Parser et de Parser a Game

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

//Game
private void printHelp ()
   {
       System.out.println("You are lost. You are alone. You wander
around at the university.");
       System.out.println("");
       System.out.println("Your command words are:");
       aParser.showCommands();
  }
```