Lita FIAHAU

L3 Informatique T7

Grâce FILITIKA

L3 Informatique T7

UNIVERSITÉ de la NOUVELLE-CALÉDONIE

Rapport de projet POO Java

Jeu “Serious Game”

Professeur: **Claire LESCHI**

Sommaire

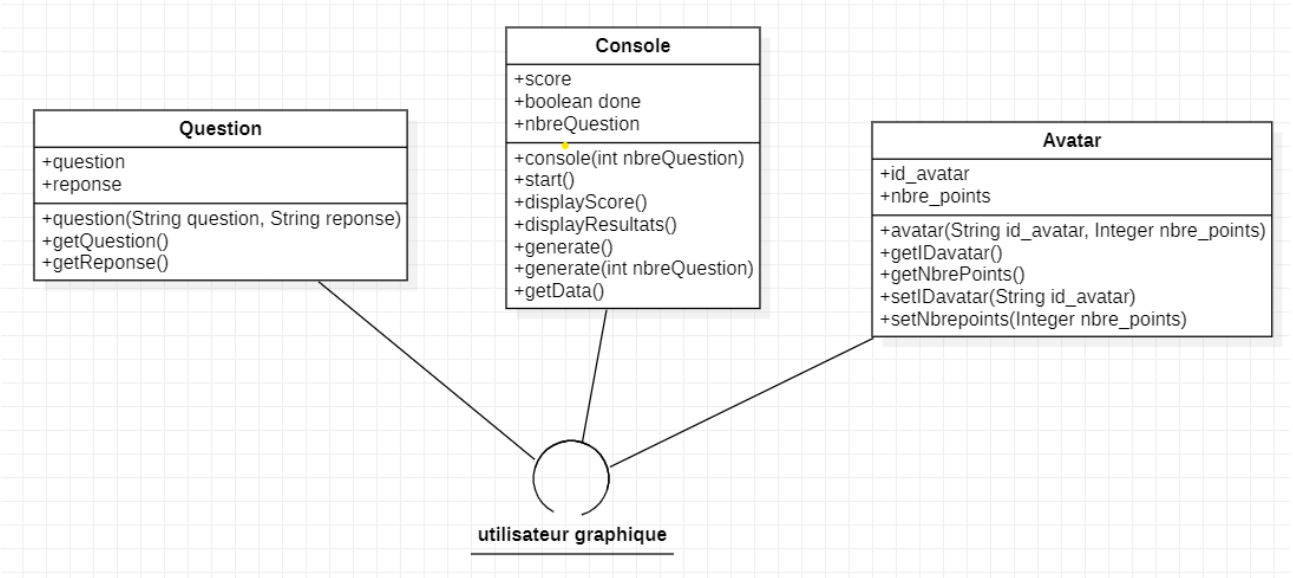
Introduction

Dans le cadre de notre formation Licence Informatique TREC 7, nous avons effectué un projet dans le module Programmation Orientée Objet en Java.

L'objectif de notre projet est de réaliser un "serious game" autour de la notion d'avatars, de questions et de défis. Chaque étudiant est associé à un avatar dont les caractéristiques dépendent de ses bulletins de notes et de sa capacité à répondre à des questions. Un avatar est caractérisé par un nombre de points de vie.
Dans ce jeu, les joueurs peuvent se défier par l'intermédiaire de leurs avatars. Lors du lancement du jeu, un joueur peut envoyer une ou plusieurs questions à son adversaire. Chaque question doit être associée à un nombre de points. Ce dernier peut diminuer ou augmenter soit en fonction de ses notes, ou de ses réponses aux questions qui lui sont envoyées ou en effectuant des tests en dehors du jeu.

Dans ce rapport, il s'agira de vous présenter les différentes fonctionnalités que nous avons implémenter ainsi que la partie technique.

Diagramme de classe



Pour le diagramme de classe, nous avons réalisé trois classes:

- la classe “Avatar” qui permet la gestion d’un avatar en lui attribuant deux paramètres qui sont un identifiant ainsi que le nombre de points
- la classe “Question” qui permet de créer une question ainsi qu’une réponse à cette dernière
- la classe “Console” qui va permettre de lancer un défi en posant une ou plusieurs questions, vérifiant les réponses et comptant le score.

Ensuite pour tester le jeu et toutes les classes nous avons créé une classe “main”.

Étapes du projet

Description des fonctionnalités (Javadoc)

Dans un premier temps, nous avons créé une classe avatar pour la création d'un avatar par un joueur. Cette classe contient deux attributs (id_avatar et nbre_points) et 5 méthodes:

- un constructeur avec en paramètres l'ID de l'avatar et un entier nbre_points pour le nombre de points pour l'avatar.

```java
import java.util.List;

public class avatar {
    protected String id_avatar;
    protected Integer nbre_points;

    /**
     * constructeur
     *
     * @param id_avatar
     * @param nbre_points
     */
    public avatar(String id_avatar, Integer nbre_points) {
        this.id_avatar = id_avatar;
        this.nbre_points = nbre_points;
    }
```

- une méthode getIDavatar() qui va nous permettre de retourner l'IDavatar de l'avatar créé.

```java
/**
 * retourne l'ID Avatar
 *
 * @return String
 */
public String getIDavatar() {
    return this.id_avatar;
}
```

- une méthode getNbrePoints() qui nous permet de retourner le nombre de points de l'avatar.

```java
/**
 * retourne le nombre de points de l'avatar
 *
 * @return Integer
 */
public Integer getNbrePoints() {
    return this.nbre_points;
}
```

- une méthode setIDavatar(String id_avatar) avec en paramètre un ID avatar qui permet à un joueur de modifier l'ID de son avatar.

```java
/**
 * modifier l'ID avatar
 *
 * @param id_avatar
 */
public void setIDavatar(String id_avatar) {
    this.id_avatar = id_avatar;
}
```

- Et une méthode setNbrepoints(Integer nbre_points) avec en paramètres le nombre de points que l'avatar souhaite modifier.

```java
/**
 * modifier le nombre de points de l'avatar
 *
 * @param nbre_points
 */
public void setNbrepoints(Integer nbre_points) {
    this.nbre_points = nbre_points;
}
```

Ensuite, nous avons implémenté une classe Question pour la gestion des questions des avatars. Cette dernière est composée de deux attributs (question et reponse) et de 3 méthodes:

- un constructeur avec une question et une réponse en paramètres

```java
/**
 * modifier le nombre de points de l'avatar
 *
 * @param nbre_points
 */
public void setNbrepoints(Integer nbre_points) {
    this.nbre_points = nbre_points;
}
```

- une méthode qui permet de retourner une question

```java
/**
 * retourne la question
 *
 * @return String
 */
// retourne la question
public String getQuestion() {
    return question;
}
```

- Et une méthode qui permet de retourner la réponse à la question.

```java
/**
 * retourne la reponse de la question
 *
 * @return reponse
 */
public String getReponse() {
    return reponse;
}
```

Après la création de ces deux classes, nous avons implémenté une classe Console avec en paramètres le score et le nombre de questions. Cette dernière va gérer le lancement du jeu et des défis. Elle contient 7 méthodes:

- Un constructeur avec en paramètre un entier qui est le nombre de questions qu'un joueur peut poser:

```java
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;

public class console {
    private int score;
    private boolean done = false;
    private int nbreQuestion;
    Scanner clavier = new Scanner(System.in);

    /**
     * Constructeur de la classe ConsoleQuizz
     *
     * @param nbreQuestion : Le nombre de questions à poser au joueur
     */
    public console(int nbreQuestion) {
        this.nbreQuestion = nbreQuestion;
    }
```

- Une méthode start() qui va permettre de poser une question, vérifier les réponses du joueur et compter le score:

```java
public void start() {
    try {
        ArrayList<question> questions = generate(nbreQuestion);
        for (int i = 0; i < nbreQuestion; i++) {
            System.out.println(questions.get(i).getQuestion());
            String userAnswer = clavier.nextLine();

            if (userAnswer.equalsIgnoreCase(questions.get(i).getReponse())) {
                score++;
                System.out.println(x:"Bonne Reponse");
            } else {
                System.out.println(x:"Mauvaise Reponse");
                System.out.println("La bonne reponse etait: " + questions.get(i).getReponse());
            }
        }
        done = true;
    } catch (IllegalArgumentException e) {
        done = false;
        System.out.println(e.getMessage());
    }

}
```

- Une méthode displayScore() qui affichera le score du joueur après avoir répondu aux questions:

```java
/**
 * Affiche le score final
 */
private void displayScore() {
    System.out.printf(format:"Votre score final est de: %d/%d.\n ", score, nbreQuestion);

}
```

- Une méthode displayResultats() qui affiche le résultat au joueur ou un message d'erreur:

```java
/**
 * Affiche le resultat au joueur, un message d'erreur sinon
 */
public void displayResultats() {
    if (done) {
        displayScore();

    } else {
        System.out.println(x:"Vous n'avez rien repondu.");
    }
}
```

- Une méthode generate() qui stocke les questions et les réponses dans un tableau en fonction du nombre de questions choisit par le joueur:

```java
public ArrayList<question> generate() {
    String[][] data = getData();
    int index = 0;
    ArrayList<question> questions = new ArrayList<question>();
    ArrayList<Integer> indexesAlreadyTaken = new ArrayList<Integer>();

    if (nbreQuestion > data.length) {
        throw new IllegalArgumentException("On ne peut generer que: " + data.length + " questions maximun");
    }
    indexesAlreadyTaken.clear();

    for (int i = 0; i < nbreQuestion; i++) {
        do {
            Random random = new Random();
            index = random.nextInt(data.length);
        } while (indexesAlreadyTaken.contains(index));
        indexesAlreadyTaken.add(index);
        String question = data[index][0];
        String reponse = data[index][1];
    }
    return questions;
}
```

- Une deuxième méthode generate(int nbreQuestion) avec en paramètre le nombre de question que peut poser le joueur:

```java
public ArrayList<question> generate(int nbreQuestion) {
    String[][] data = getData();
    int index = 0;
    ArrayList<question> questions = new ArrayList<question>();
    ArrayList<Integer> indexesAlreadyTaken = new ArrayList<Integer>();

    if (nbreQuestion > data.length) {
        throw new IllegalArgumentException("On ne peut generer que: " + data.length + " questions maximun");
    }
    indexesAlreadyTaken.clear();

    for (int i = 0; i < nbreQuestion; i++) {
        do {
            Random random = new Random();
            index = random.nextInt(data.length);
        } while (indexesAlreadyTaken.contains(index));
        indexesAlreadyTaken.add(index);
        String pays = data[index][0];
        String president = data[index][1];
        String questionText = String.format(format:"Quel est le nom du Président de : %s? ", pays);
        questions.add(new question(questionText, president));
    }
    return questions;
}
```

- Et une dernière méthode getData() qui crée un tableau contenant des couples “question-réponse”, pour notre jeu nous avons créé des questions que sur les différents présidents donc dans ce tableau nous avons des couples composés du pays et du nom du Président:

```
private static String[][] getData() {
    String[][] data = { { "France", "Macron" }, { "Etats Unis", "Biden" }, { "Russie", "Poutine" },
            { "Ukraine", "Zelensky" } };
    return data;
}
```

Ensuite pour tester toutes ces classes crées nous avons implémenté une classe Main:

```java
public class main {
    /**
     * @param args
     */
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println(x:"Entrez votre nom d'avatar");
        String nom = scanner.nextLine();
        System.out.println(x:"Entrez vos notes");
        List<Double> valeurs = new ArrayList<>();

        // Demander à l'utilisateur de saisir des nombres entiers
        System.out.print(s:"Combien de notes voulez-vous saisir ? ");
        Double n = scanner.nextDouble();

        for (int i = 0; i < n; i++) {
            System.out.print(s:"Saisissez une note : ");
            Double valeur = scanner.nextDouble();
            valeurs.add(valeur);
        }

        // Calculer la somme des valeurs dans la liste
        Double somme = 0.0;
        for (Double valeur : valeurs) {
            somme += valeur;
        }

        // avatar avatar1 = new avatar(nom, somme);
        // avatar avatar2 = new avatar(nom, somme);

        // Afficher la somme
        System.out.println("Votre nombre de point de vie est de : " + somme);

        if (somme == 0) {
            System.out.println(x:"Vous ne pouvez pas lancer de défis.");
        } else {
            System.out.println(x:"Entrez le nombre de question à envoyer : ");
            Integer nbreQuestion = scanner.nextInt();

            console qcm = new console(nbreQuestion);
            qcm.start();
            qcm.displayResultats();
```

Problèmes rencontrés

Au début de notre projet, nous avons eu des problèmes d'environnement car nous n'avons pas réussi à exécuter du code java sur un de nos ordinateurs.
Ensuite, en ce qui concerne le projet la première partie était simple sauf que pour la gestion de 2 avatars qui se défient nous avions eu du mal à voir comment réaliser cela.
Pour la partie question aussi, nous n'avions pas trop su comment gérer les questions ainsi que les réponses.

Continuité

Pour la continuité, la chose que l'on pourrait faire par la suite serait de réaliser toutes les fonctionnalités avancées qui ont été proposées dans le sujet et ainsi améliorer le prototype de notre "serious game". La gestion de notre temps aussi est important pour pouvoir finir à temps ce projet.

Conclusion

Pour conclure, malgré le retard pris par les problèmes d'environnement nous avons tout de même pu réaliser un jeu basique.

Ce projet nous a quand même permis de mettre en pratique les notions que nous avons apprises tout le long du semestre.

Nous avons su communiquer et nous entraider en équipe, afin de rendre, le mieux possible ce qui a été demandé.