TRAN Léo & DIEYE Papa Amadou

Groupe 111

Rapport

PROJET BPO

Table des matières

[Introduction 2](#_Toc10025525)

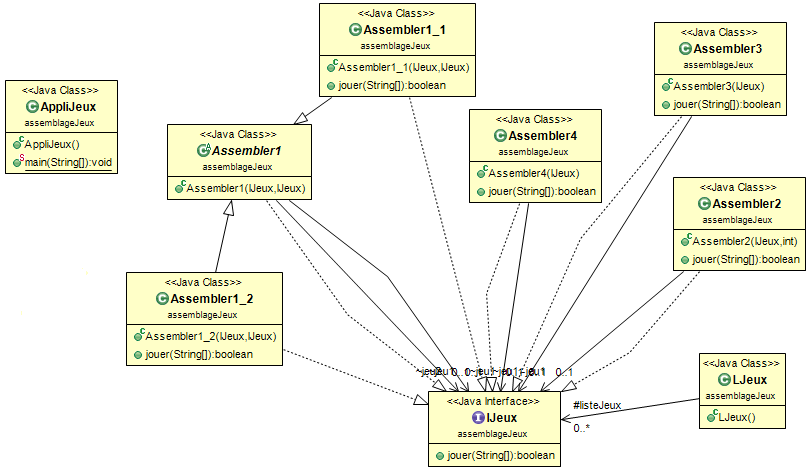
Diagramme UML des classes………………………………………………………………………………………….3

[Code des classes java 4](#_Toc10025526)

[Bilan de projet ………………………………………………………………………………………………………………12](#_Toc10025527)

# Introduction

Nous devions développer une bibliothèque de jeux en java. Notre bibliothèque devait permettre de pouvoir assembler n’importe quels jeux des quatre manières différentes et puis d’y jouer. La première méthode d’assemblage était de jouer à un jeu 1 et 2 en remportant les deux jeux. La deuxième était de jouer à un jeu 1 et 2 en remportant l’un des deux jeux. La troisième était de jouer à un jeu 1 en remportant ce jeu dans un nombre d’essais donné. Le dernier assemblage devait permettre de rejouer à un jeu 1 jusqu’à le remporter.

Diagramme UML des classes

# Code des classes java

AppliJeux.java

/\*@author TRAN Léo, DIEYE Papa Amadou

\* @version 1.0

\*/

package assemblageJeux;

public class AppliJeux {

public static void main(String[] args) {

// LJeux Jeux = new LJeux();

// IJeux jeu = new Assembler1\_1(new IJeux(), new IJeux());

// jeu.jouer(args);

// Jeux.listeJeux.add(jeu);

// Jeux.listeJeux.add(new Assembler1\_1(Jeux.listeJeux.get(x),Jeux.listeJeux.get(y))

// Jeux.listeJeux.get(i).jouer(args);

}

}

IJeux.java

/\*@author TRAN Léo, DIEYE Papa Amadou

\* @version 1.0

\*/

package assemblageJeux;

/\*@brief Interface des jeux

\*/

public interface IJeux {

/\*@brief

\* @return retourne si la partie est gagnée ou non

\*/

boolean jouer(String[] args);

}

LJeux.java

/\*@author TRAN Léo, DIEYE Papa Amadou

\* @version 1.0

\*/

package assemblageJeux;

import java.util.ArrayList;

import java.util.List;

import jeux.crazy.AppliCrazy;

import jeux.devinettes.AppliDevinettes;

import jeux.intrus.AppliIntrus;

import jeux.pendu.AppliPendu;

import jeux.pfc.AppliPierreFeuilleCiseaux;

import jeux.pppg.AppliPlusPetitPlusGrand;

import jeux.suites.AppliSuite;

import jeux.tictactoe.AppliTicTacToe;

/\*@brief Liste où les jeux seront stockés pour créer de nouveau jeux

\*/

public class LJeux {

protected List<IJeux> listeJeux =new ArrayList <IJeux>();

public LJeux () {

// listeJeux.add(new IJeux());

}

}

Assembler1.java

/\*@author TRAN Léo, DIEYE Papa Amadou

\* @version 1.0

\*/

package assemblageJeux;

/\*@brief Classe abstraite des Assembler1\_1 et Assembler1\_2 \*/

public abstract class Assembler1 implements IJeux {

IJeux jeu1;

IJeux jeu2;

public Assembler1 (IJeux jeu1,IJeux jeu2) {

this.jeu1= jeu1;

this.jeu2= jeu2;

}

}

Assembler1\_1.java

/\*@author TRAN Léo, DIEYE Papa Amadou

\* @version 1.0

\*/

package assemblageJeux;

/\*@brief Assemble les deux jeux en paramètre pour créer un nouveau jeu où il faut gagner les deux jeux pour gagner le nouveau jeu

\*/

public class Assembler1\_1 extends Assembler1 implements IJeux {

public Assembler1\_1(IJeux jeu1, IJeux jeu2) {

super(jeu1, jeu2);

}

@Override

public boolean jouer(String[] args) {

try {

if(jeu1.jouer(args)&&jeu2.jouer(args)) {

System.out.println("Bravo");

return true;

}

else {

System.out.println("Dommage");

return false;

}

}

catch (Exception e) {

}

return true;

}

}

Assembler1\_2.java

/\*@author TRAN Léo, DIEYE Papa Amadou

\* @version 1.0

\*/

package assemblageJeux;

/\*@brief Assemble les deux jeux en paramètre pour créer un nouveau jeu où il faut gagner un des deux jeux pour gagner le nouveau jeu

\*/

public class Assembler1\_2 extends Assembler1 implements IJeux{

public Assembler1\_2 (IJeux jeu1,IJeux jeu2) {

super(jeu1, jeu2);

}

@Override

public boolean jouer(String[] args) {

try {

if(jeu1.jouer(args))

return true;

else if(jeu2.jouer(args)) {

System.out.println("Bravo");

return true;

}

else {

System.out.println("Dommage");

return false;

}

}

catch (Exception e) {

return true;

}

}

}

Assembler2.java

/\*@author TRAN Léo, DIEYE Papa Amadou

\* @version 1.0

\*/

package assemblageJeux;

/\*@brief Créer un nouveau jeu avec le jeu en paramètre où il faut gagner le jeu en moins de n tentatives pour gagner le nouveau jeu

\*/

public class Assembler2 implements IJeux{

private IJeux jeu1;

private int n;

public Assembler2 (IJeux jeu1, int n) {

this.jeu1= jeu1;

this.n = n;

}

@Override

public boolean jouer(String[] args) {

try {

for(int i=0;i<this.n;++i) {

if(jeu1.jouer(args)) {

System.out.println("Bravo");

return true;

}

}

System.out.println("Dommage");

return false;

}

catch (Exception e) {

return true;

}

}

}

Assembler3.java

/\*@author TRAN Léo, DIEYE Papa Amadou

\* @version 1.0

\*/

package assemblageJeux;

/\*@brief Créer un nouveau jeu avec le jeu en paramètre où l'on joue tant qu'on a pas gagné pour gagner le nouveau jeu

\*/

public class Assembler3 implements IJeux{

IJeux jeu1;

public Assembler3 (IJeux jeu1) {

this.jeu1= jeu1;

}

@Override

public boolean jouer(String[] args) {

try {

boolean b=false;

while (b!=true) {

b=jeu1.jouer(args);

}

System.out.println("Bravo");

return true;

}

catch (Exception e) {

return true;

}

}

}

## Bilan de projet

Dans notre groupe, on avait des difficultés dans la compréhension du sujet et un peu sur la programmation en java. Mais le travail à 2 fut bénéfique pour nous, car cela nous a donné l’opportunité de progresser, de trouver des solutions sur nos problèmes et de pouvoir ainsi échanger nos opinions et idées sur le projet.

Nous avons aussi remarqué lors du codage qu’il y avait des tâches répétitives et que l’on comprenait mieux le développement de la bibliothèque ensembles et que l’ambiance de travail a fait ressortir un travail d’équipe efficace dans notre binôme. Ainsi que dans la réflexion et propositions de solutions, que l’on a adoptées.

On peut en conclure que ce projet était intéressant et nous a permis d’appliquer nos connaissances et ainsi nous permettre de gagner en expérience.