

# Rapport Projet

Benjamin Singabrayen, L3 informatique

13 novembre 2018

## Résumé

Dans ce rapport, je décris le code d'un programme choisi pour un projet et je donne également mes ressentis.

## 1 Introduction

Ceci est le Latex que Mr.Etienne Payet nous a donné à rédiger qui conclut l'aboutissement de notre Projet. Nous avons eu au départ le choix entre l'exercice 2.3 et 3.2 de notre TP Programmation Concurrente. J'ai choisis l'exercice 3.2 qui consiste à écrire une programme,soit en java soit en python, qui affiche des balles en mouvements dans une fenêtre.J'ai bien sur choisis de programmer en java car je suis plus à l'aise avec ce langage de programmation que python, c'est celui que je maîtrise le plus. Pourquoi ce choix ? Tout simplement car l'exercice sur les balles m'a parue dans un premier temps plus intéressant que l'exercice 2.3 qui consiste à créer deux threads producteurs et consommateurs qui se partagent une file.

Mon projet se compose de 5 fichiers java :

- Ball.java
- Fenetre.java
- GamePanel.java
- Game.java
- Tiimer.java

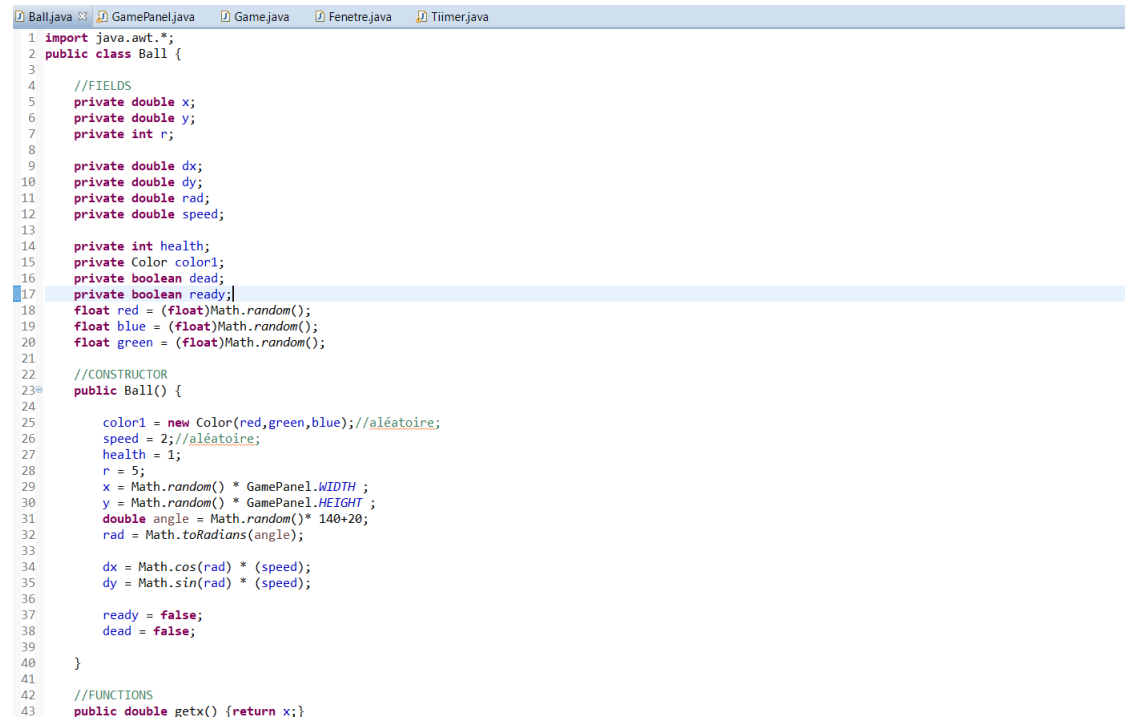


## 2 Focus sur la partie codage

### 2.1 Class Ball

Je vais commencer par vous présenter ma classe Ball, car c'est aussi la première par laquelle j'ai commencé. À mon avis c'est celle qui contient le code le plus basique et c'est surtout l'objet de base de notre programme, sans elle notre affichage graphique serait tout banalement une simple fenêtre blanche.

Voici un aperçu de ma classe Ball :



```
1 import java.awt.*;
2 public class Ball {
3
4     //FIELDS
5     private double x;
6     private double y;
7     private int r;
8
9     private double dx;
10    private double dy;
11    private double rad;
12    private double speed;
13
14    private int health;
15    private Color color1;
16    private boolean dead;
17    private boolean ready;
18    float red = (float)Math.random();
19    float blue = (float)Math.random();
20    float green = (float)Math.random();
21
22    //CONSTRUCTOR
23    public Ball() {
24
25        color1 = new Color(red,green,blue);//aléatoire;
26        speed = 2;//aléatoire;
27        health = 1;
28        r = 5;
29        x = Math.random() * GamePanel.WIDTH ;
30        y = Math.random() * GamePanel.HEIGHT ;
31        double angle = Math.random()* 140+20;
32        rad = Math.toRadians(angle);
33
34        dx = Math.cos(rad) * (speed);
35        dy = Math.sin(rad) * (speed);
36
37        ready = false;
38        dead = false;
39    }
40
41    //FUNCTIONS
42    public double getX() {return x;}
43
```

Ma classe balle possède plusieurs paramètres :

1. Les paramètres de positions :
  - double x : coordonnée x de l'objet Ball
  - double y : coordonnée y de l'objet Ball
2. les paramètres de déplacements :
  - double dx : la coordonnée x d'une balle sera incrémentée de la valeur dx pour les déplacements horizontaux
  - double dy : la coordonnée y d'une balle sera incrémentée de la valeur dy pour les déplacements horizontaux
  - double speed : vitesse de déplacement de la balle
3. les paramètres qui gèrent la forme de la balle :
  - int r : représente le rayon de l'objet Ball
  - Color color : couleur de notre objet Ball

Les parties intéressantes et plutôt délicates à aborder selon moi dans cette classe Ball sont : la fonction public void update() et l'initialisation de la couleur de la balle avec une couleur aléatoire dans le constructeur de Ball.

Le code pour la couleur de la balle :

```
float red = (float)Math.random();
float blue = (float)Math.random();
float green = (float)Math.random();

public Ball(){
    color1 = new Color(red,green,blue);
}
```

Les trois variables red, blue, green initialisées par un nombre flottant aléatoire entre 0 et 1( (float)Math.random(); ) permet d'obtenir des teints de couleur rouge,vert et bleu différents pour chaque paramètre lors de la création de l'objet Color(red,green,blue) dans notre constructeur Ball.

Le code qui permet de dessiner une balle en tant qu'élément graphique ci-dessous est la fonction qui sera appelée dans le panneau pour dessiner chaque balle de ma liste de balles :

```
public void draw(Graphics2D g){
    g.setColor(color1);
    g.fillOval((int) (x-r), (int)(y-r), 2*r, 2*r);
    g.setStroke(new BasicStroke(3));
    g.setColor(color1.darker());
    g.drawOval((int) (x-r), (int)(y-r), 2*r , 2*r);
    g.setStroke(new BasicStroke(1));
}
```

Enfin la dernière partie et la plus difficile de cette classe est celle qui gère les mouvements, est la fonction public void update() son code est le suivant :

```
public void update() {
    x += dx;
    y += dy;

    if(x < r && dx < 0) dx = -dx;
    if(y < r && dy < 0) dy = -dy;
    if(x > GamePanel.WIDTH -r && dx > 0) dx = -dx;
    if(y > GamePanel.HEIGHT -r && dy > 0) dy = -dy;
}
```

Dans cette fonction on change les coordonnées pour faire bouger la balle, en additionnant dx et dy respectivement à x et y. Ce changement de coordonnées permet de déplacer la balle dans le panneau;la balle bouge des coordonnées x en x+dx et des coordonnées y en y+dy. Ensuite dans la deuxième partie de ce code, on a crée des conditions pour prendre en compte les rebonds des balles sur les bords de notre panneau.

1. ième condition if
  - test de la condition : la balle se déplace à gauche et sa coordonnée x est inférieure à son rayon
  - conséquence : la balle se déplace à droite
2. ième condition if
  - test de la condition : la balle se déplace vers le haut et sa coordonnée y est inférieure à son rayon
  - conséquence : la balle se déplace vers le bas

3. ième condition if
  - test de la condition : la balle se déplace à droite et sa coordonnée x est supérieure à (la largeur du panneau - le rayon de la balle)
  - conséquence : la balle se déplace à gauche en x
4. ième condition if
  - test de la condition : la balle se déplace vers le bas et sa coordonnée y est supérieure à (la hauteur du panneau - le rayon de la balle)
  - conséquence : la balle se déplace vers le haut

## 2.2 Class Timer

```

1 import java.util.Timer;
2
3 public class Tiimer implements Runnable {
4
5     public int second;
6     public boolean state;
7
8     public Tiimer() {
9         super();
10        this.state = true;
11        this.second = 0;
12    }
13    public void run() {
14        while(true) {
15            while(state == true) {
16                try {
17                    this.second++;
18                    Thread.sleep(1000);
19                    System.out.println(""+this.second);
20                } catch (InterruptedException e) {
21                    // TODO Auto-generated catch block
22                    e.printStackTrace();
23                }
24            } while(state == false) {
25                try {
26                    Thread.sleep(0);
27                } catch (InterruptedException e) {
28                    // TODO Auto-generated catch block
29                    e.printStackTrace();
30                }
31            }
32        }
33    }
34 }
35
36 }
37

```

Mon objet timer possède deux paramètres :

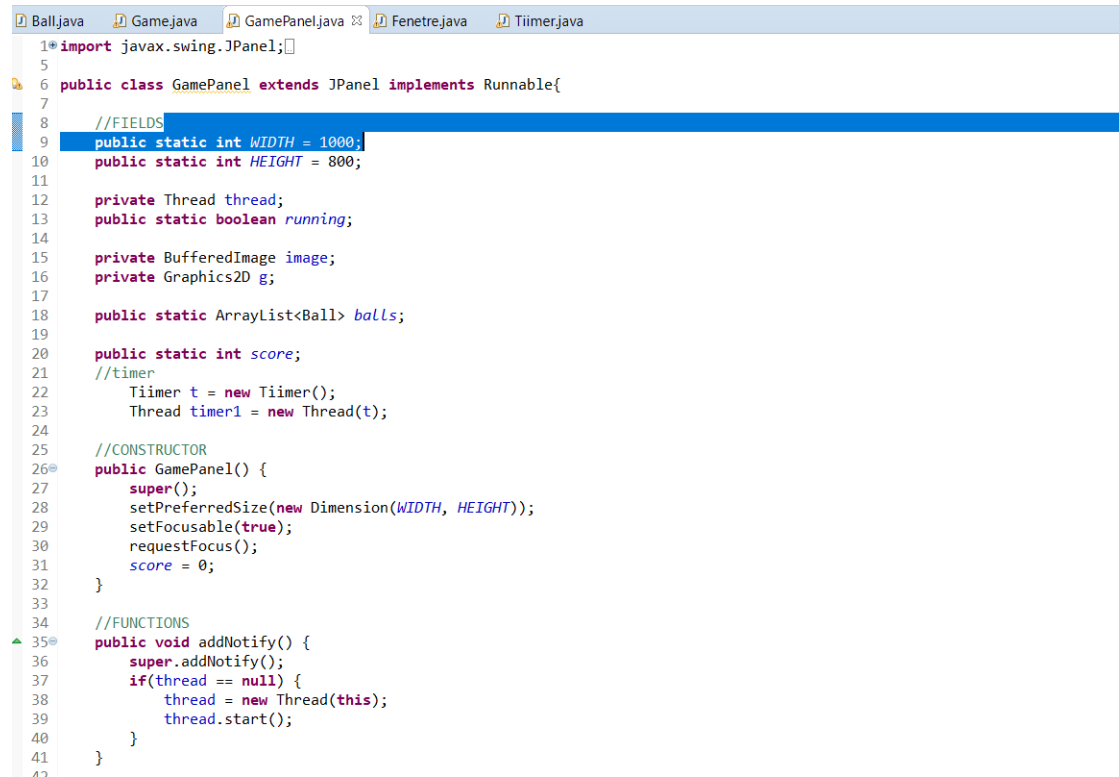
- int second;
- boolean state;

Ma classe Timer implémente l'interface Runnable. Elle possède une fonction run(). Sa fonction run() contient une boucle infini qui contient une boucle qui est exécutée si l'état du Tiimer est vrai, dans cette boucle les secondes s'incrémentent de 1 en 1 et mon Thread Tiimer dort pendant 1000 millisecondes. Si l'état du timer passe à faux une autre boucle dans la boucle while(state = true) est exécutée et le timer dort pendant 0 milliseconde. Faire cela permet de mettre le timer en pause quand l'état est à false.

## 2.3 Class GamePanel

Ma classe GamePanel est la plus importante, on dira que c'est le cerveau de mon code, C'est elle qui contient la majorité du code et qui gère l'affichage de mon Panel, là où mes balles se créent et se déplacent.

Aperçu de ma classe GamePanel :



```
1 import javax.swing.JPanel;
2
3
4 public class GamePanel extends JPanel implements Runnable{
5
6     //FIELDS
7     public static int WIDTH = 1000;
8     public static int HEIGHT = 800;
9
10    private Thread thread;
11    public static boolean running;
12
13    private BufferedImage image;
14    private Graphics2D g;
15
16    public static ArrayList<Ball> balls;
17
18    public static int score;
19    //timer
20    Tiimer t = new Tiimer();
21    Thread timer1 = new Thread(t);
22
23    //CONSTRUCTOR
24    public GamePanel() {
25        super();
26        setPreferredSize(new Dimension(WIDTH, HEIGHT));
27        setFocusable(true);
28        requestFocus();
29        score = 0;
30    }
31
32    //FUNCTIONS
33    public void addNotify() {
34        super.addNotify();
35        if(thread == null) {
36            thread = new Thread(this);
37            thread.start();
38        }
39    }
40
41 }
```

On va commencer tout d'abord par la fonction run() du thread. Cette fonction est basée sur le même principe que celui du timer.

```
public void run() {
    running=true;
    //lancer chrono
    timer1.start();
    image = new BufferedImage(WIDTH, HEIGHT, BufferedImage.TYPE_INT_RGB);
    g = (Graphics2D) image.getGraphics();
    balls = new ArrayList<Ball>();
    // balls number instance
    for(int i = 0; i <30; i++) {
        balls.add(new Ball());
    }
    while(running) {
        gameUpdate();
        gameRender();
        gameDraw();
        //System.out.println(running);
        try {
            Thread.sleep(3);
        }catch(InterruptedException e) {
```



Lorsqu'on appuie sur le bouton auquel on ajoute cette écoute, dans notre Fenêtre c'est le bouton startstop, Si la variable running de notre GamePanel est à true, elle passe à false et les balles s'arrêtent de bouger et le timer arrête aussi le décompte du temps. Vice versa si il est à false il passe à true et les balles reprennent leur mouvement et le timer relance son décompte.

La classe pour mon écoute de bouton add :

```
class BoutonListener2 implements ActionListener{ //ajouter un timer
//Redéfinition de la méthode actionPerformed()
    public void actionPerformed(ActionEvent arg0) {
        GamePanel.balls.add(new Ball());
        if (GamePanel.running == false){
            for(int i = 0; i < GamePanel.balls.size(); i++) {
                pan.gameRender();
                pan.gameDraw();
            }
        }
    }
}
```

Lors de l'appui sur le bouton add, il y a un ajout de balle dans notre tableau de balle qui a été instancié dans mon GamePanel. Pour gérer l'affichage des balles lorsque mon thread GamePanel est en pause, j'ai rajouté les fonctions gameRender() et gameDraw() de ma classe GamePanel(vu dans la sous section 2.3.

La classe pour mon écoute de bouton remove :

```
class BoutonListener3 implements ActionListener{
    public void actionPerformed(ActionEvent arg0) {
        if (GamePanel.balls.size()!=0) {
            GamePanel.balls.remove(0);
            if (GamePanel.running == false){
                pan.gameRender();
                pan.gameDraw();
            }
        }
    }
}
```

Le bouton remove fonctionne identiquement à celui de add, au lieu d'ajouter les balles on en enlève. Mais il reste une petite subtilité, si on enlève une balle alors que le tableau de balles est vide alors il y a une message d'erreur qui apparaît. J'ai réglé cette erreur en ajoutant une condition qui exécute la suppression d'une balle si et seulement si la longueur du tableau de balles est différente de zero.

### 3 Conclusion

Pour conclure il était intéressant de faire ce projet, il abordait toutes les notions qu'on a vu depuis la L2 sur java et également les Threads qu'on a étudiés en L3.

Le Thread étaient la principale difficulté rencontrée dans ce projet. Je n'étais pas très à l'aise au début avec cette notion, on ne l'a vu que très récemment. Mais ce fut une réelle expérience de pratiquer cela. Pour m'aider dans la programmation, j'ai utilisé la documentation java et le site OpenClassroom.

Pour améliorer mon code, je pourrais rajouter une entrée clavier pour que l'utilisateur puisse spécifier combien de balle il veut créer au lancement du programme. J'avais pensé aussi à rajouter un bouton qui permet à l'utilisateur de changer la taille des balles, mais par respect de la consigne et du cahier des charges je ne l'ai pas fait.

### 4 Bibliographie

#### Références

- [1] Herby Cyrille. Openclassroom. <https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java>.
- [2] Swing. <https://docs.oracle.com/javase/tutorial/uiswing/index.html>.