Pink Pong Game

TECH GIRLS TEAM
Layla de Souza Barbosa
Lizandra Esterque
Programming Fundamentals - Gurleen Kaur
Computer Programming Diploma
Georgian@Ilac College

I. Project Overview

The PinkPongGame Application is a ping pong game simulator program, aiming to fulfill requirements for the final project in Programming Fundamentals course for Computer Programming Diploma. The amusement of the game is to prevent the ball from escaping from your side of the field, which leads to increase a point for your rival's score. Each interaction of the ball with the field/player increases its speed and changes its direction, escalating the difficulty level of the game.

The development environment used to create the application was JAVA 18, along with 4 external libraries that made an easy path for an easier graphic design and smoother game play. The list of external sources included the following libraries:

- Java AWT: contains all of the classes for creating user interfaces and for painting graphics and images;

- Java AWT Event: provides interfaces and classes for dealing with different types of events(inputs) fired by AWT components;

- JAVA UTIL: accommodates the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes. Mostly. This application used the Random, Math, String and some other functions from this library;

- JAVAX SWING: allows the use of components implement in the easiest possible way the creation of a GUI project.

II. Functional Requirements

The project was organized into six different classes, and outlined as follows:

1. *PinkPongGame Class*

- **Main method:**
  It is used to create an object from the GameFrame class, where the game starts.

2. *Paddle Class that extends Rectangle Class*

- **Instantiate 3 variables:**
  - Integer id;
  - Integer yVelocity;
  - Integer speed.
- **A constructor that takes 5 integer arguments:**
  It operates the super method to invoke the awt.Rectangle constructor, then overwrite it to assign id value as an additional parameter.

- **keyPressed method that take a KeyEvent argument:**

Using a switch controller that takes the id variable as a parameter, it selects which of the paddles will execute certain function. Later, nested if controllers vary the code block taking into consideration the pressed keyboard key.

Therefore, resulting in the possible combination pairs (key == value attribute to variable speed &paddleid), namely: W == negative value & paddle1; S == positive value & paddle1; UP == negative value & paddle2; and DOWN == positive value & paddle2. Subsequently, the respective value is attributed to speed variable, which is used as a parameter to setYDirection method that is called. Move method is also called to improve smoothness of paddle movement.

- **KeyReleased method that take a KeyEvent argument:**

The method applies the same structure of code as keyPressed method. Only, when keyboard key is released, instead of using speed as a parameter, it passes the 0 digit to setYDirection method. It aims to prevent the paddle from moving for an undetermined time.

- **setYDirection method that takes an integer argument:**
  It attributes the integer parameter to yVelocity variable.

- **move method that take no argument:**
  This compound the value of y with yVelocity.

- **draw method that take a Graphics argument:**
  It defines the color and format of the paddle using methods from awt.Graphics Class.

3. Ball Class Extends Rectangle Class

- **4 variables:**
  - Random random;
  - Integer xVeloccity;
  - Integer yVelocity;
  - Integer initialSpeed.

- **A constructor that takes 4 integer arguments:**
  It avails of the super method to invoke the awt.Rectangle constructor, then overwrite it using a randomly generated integer (0 or 1) to assign value to xVelocity. Furthermore, utilizing an if controller, reassigns the value of 0 to -1 (minus one). Thus, it passes the xVelocity value as a parameter calling the setXDirection method to define the ball direction in a x line. Same methodology is applied to y line, using the respective variables and method.

- **SetYDirection method that takes an integer arguments**
  This one attributes the integer parameter to yVelocity variable.

- **SetXDirection method that takes an integer arguments**
  It attributes the integer parameter to xVelocity variable.

- **Move method that takes no argument**
  This compound the value of x and y variables with xVelocity and yVelocity, respectively.

- **Draw method that takes a Graphic argument**
  It defines the color and format of the ball using methods from awt.Graphics Class.

4. Score Class extends Rectangle Class

- **4 static variables:**
  - Integer GAME_WIDTH;
  - Integer GAME_HEIGHT;
  - Integer player1;
  - Integer player2.
- **A constructor that takes 2 integer arguments:**
  It creates the score object with the pre-defined integers equal to the size of the game panel.

- **Draw method that takes a Graphic argument:**
  It defines the color, font and size, and format of the score. Also, it draws a Line separating each player's side of field and writes the players score and game name on the screen, all using methods from awt.Graphics Class.

5. GameFrame Class extends JFrame

- **An empty constructor:**

It creates an object from the GamePanel Class, then uses methods from the awt.Frame class to structure the game frame, iterating with it in some ways such as: defining name of the game, and background color (our choice was pink), preventing the game to be resized, configuring the game to exit when closed, making the game visible on the user's screen, and adjusting the frame to the size of the game.

6. GamePanel Class extends JPanel implements Runnable

- **6 static final variables:**

The reason we chose to apply the keyword final to the variables was to prevent undesired changes to them in any way, considering that they are extremely important for the game structure.

- Integer GAME_WIDTH;
- Integer GAME_HEIGHT;
- Dimension SCREEN_SIZE;
- Integer BALL_DIAMETER;
- Integer PADDLE_WIDTH;
- Integer PADDLE_HEIGHT.

- **8 class variables:**
  - Thread gameThread;
  - Image image;
  - Graphics graphics;
  - Random radom;
  - Paddle paddle1;
  - Paddle paddle2;
  - Ball ball;
  - Score score.

- **An empty constructor:**

It calls the methods newBall and newPaddles and creates a score object with the aim of beginning the game. In addition, utilizing methods from awt.Component and javax.swing Classes, it opens the game in the user's screen, the door to receive input from the user's keyboard, and define the size of the game on the screen using the SCREEN_SIZE variable as a parameter. Thus, it instantiates a Thread object to uses its method start to initiate the game.

- **newBall method that takes no arguments:**

It creates a Ball object, passing the following parameters:

- x= half of GAME_WIDTH minus half of BALL_DIAMETER, so the ball appears on the exactly middle x of the field;

- y= a random chosen integer using as a parameter half of GAME_HEIGHT minus half of BALL_DIAMETER, so the ball appears on the anywhere on the y line of the field;

- Width and height =BALL_DIAMETER, so the ball has a round format.

-

- **newPaddles method that takes no arguments:**
  It creates 2 Paddle objects, passing the following parameters:

  - paddle1 x = 0; so the paddle appears on the left side of the field;

  - paddle1 y = half of GAME_HEIGHT minus half of PADDLE_HEIGHT; so the paddle appears on the the exactly middle y of the field;

  - paddle1 id = 1;

  - paddle2 x = GAME_WIDTH minus PADDLE_WIDTH; so the paddle appears on the right side of the field;

  - paddle1 y = half of GAME_HEIGHT minus half of PADDLE_HEIGHT; so the paddle appears on the the exactly middle y of the field;

  - Paddle2 id = 2;

  - paddle1 and paddle2 width = PADDLE_WIDTH;

  - paddle1 and paddle2 height = PADDLE_HEIGHT.

- **Paint method that takes a Graphics argument:**
  It uses methods from awt.Componet and awt.Image Classes to make possible generating images on user's screen.

- **Draw method that takes no argument:**
  It is called to draw the paddles, ball and score on the screen.

- **Move method that takes no argument:**
  Everytime it is called, it creates motion and action on the game.

- **Collision method that takes no argument:**

This is the most complex method of the project, having a handful of vital functions for the game.

A. It utilizes if controllers to prevent paddles to move out of frame edges in a y direction;

B. It uses if controllers to prevent the ball from moving out of the frame in a y direction. In addition, when the ball bounces on the top/bottom frame, it reverses the ball direction and increases its velocity, making the game more difficult;

C. It employs if controllers to reverse ball direction and increase its velocity when it intersects with paddles;

D. When the ball moves out of the frame in x direction, it operates if controllers to increase rival player's score. Next, it creates a new ball and new paddles, so the game carries on with everything on its initial position on the field. Additionally, it prints out the points for the scoring player on the console.

- **Run method that takes no arguments:**

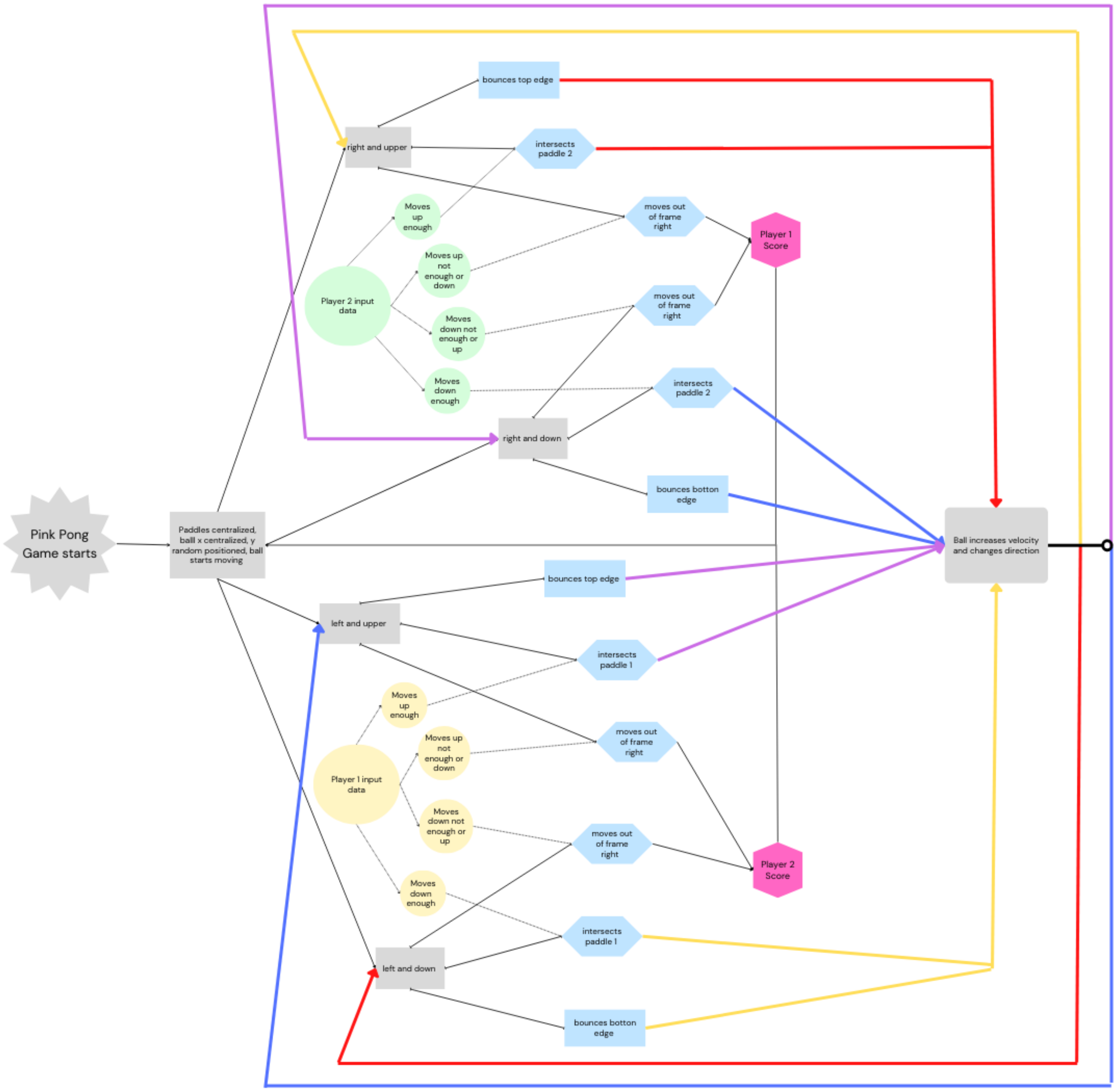
Taking advantage of a while controller with an always true boolean parameter, it keeps the game lopping on screen. Nested inside the while, there is a if controller that calls move, collision and repaint methods every nanosecond, providing the game a fluid motion.

- **SubClass AL extends KeyAdapter:**

It includes two methods that receive inputs from the keyboard and attributes them to the paddles. Each method attributes the input of on (keyPressed) or off (keyReleased).

III. Project Design

A little context to the diagram, square boxes means only code programming applied to that result, round boxes mean user's input data, and hexagon boxes means a mix of user's input and code programming lead to that action. You can follow the actin until it gets back to the code programming flow by the code coloured arrows.



IV. Work Assignments

Team member	Functional Requirements	Java Classes
Layla	**1, 2, 3, 4, 5, 6**	1. Score Class 2. GameFrame Class 3. GamePanel Class 4. PinkPongGame
Lizandra	**1, 2, 3,4, 5, 6**	1. Ball Class 2. Paddle Class 3. GamePanel Class 4. PinkPongGame