`

BRICK BREAKER GAME IN JAVA

A PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

Under the guidance of

MR. JAGBINDER SIR

BY

ADARSH BARDHAN

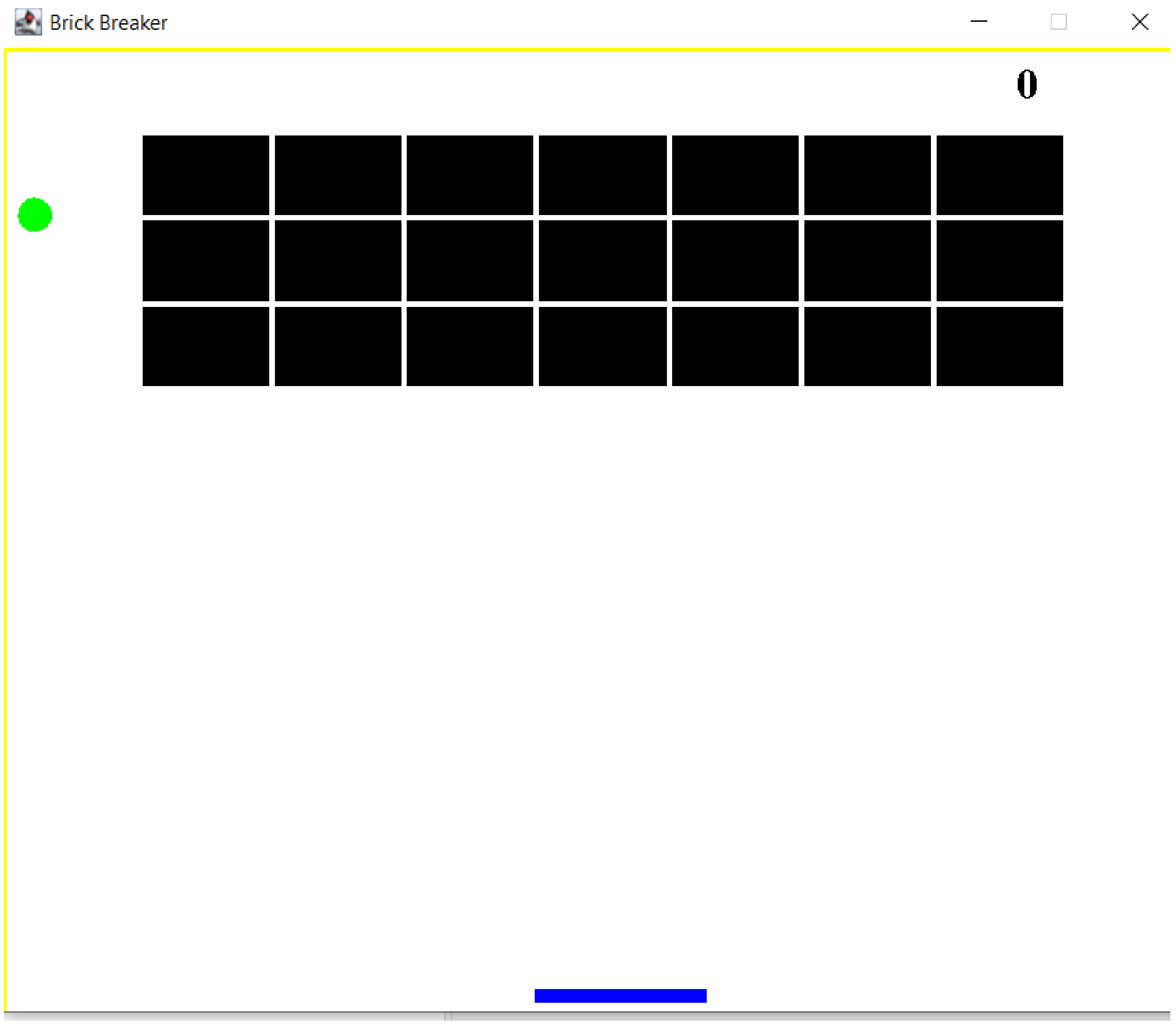
MILIND UDBHAV

PRACHI JAIN

YASH JAIN

Introduction

We made an interactive game based upon the classic game brick breaker In java. The object of brick breaker is to break the bricks that are distributed around the top of the game screen. The bricks are broken after coming in contact with a ball that bounces around the screen. At the bottom is a paddle that in the classic game moves based on user input. The user has to make sure the ball bounces off the paddle without going off the bottom of the screen. In our implementation, we have used the arrow keys to track the position of the paddle.



BRICK BREAKER GAME OVERVIEW

IMPLEMENTATION

A. Paddle control

The paddle will be controlled by a mouse. The mouse's moving to left or right will correspond to the paddle's movement. The moving speed will also relate to how fast we move the mouse. By clicking on the left key, the paddle can launch extra balls as bonus in higher level rounds. Software will be used to keep track of the current status of the paddle, to see if it's under special status, like elongated, shortened, multi-ball launching, or fireball launching (fireball can destroy all bricks in its trajectory).

B. Bouncing ball

Software will be used to assigning new locations of the ball as the ball bounce around walls (edges of the screen), bricks and paddle. When the angle of incidence changes, the angle of reflection changes too. We will also need a status tracker to see whether the ball is in regular status or fireball status. Also, we implement the game with player have multiple lives.

C. Bricks

For each brick, the software will have a status tracker. The tracker will record how many breaks needed to break a specific brick (different types of brick may need 1 or 2 or 3 times of breaking before they are destroyed), what special effect they have and when to release, and to display or not (remaining or destroyed).

D. Score counting

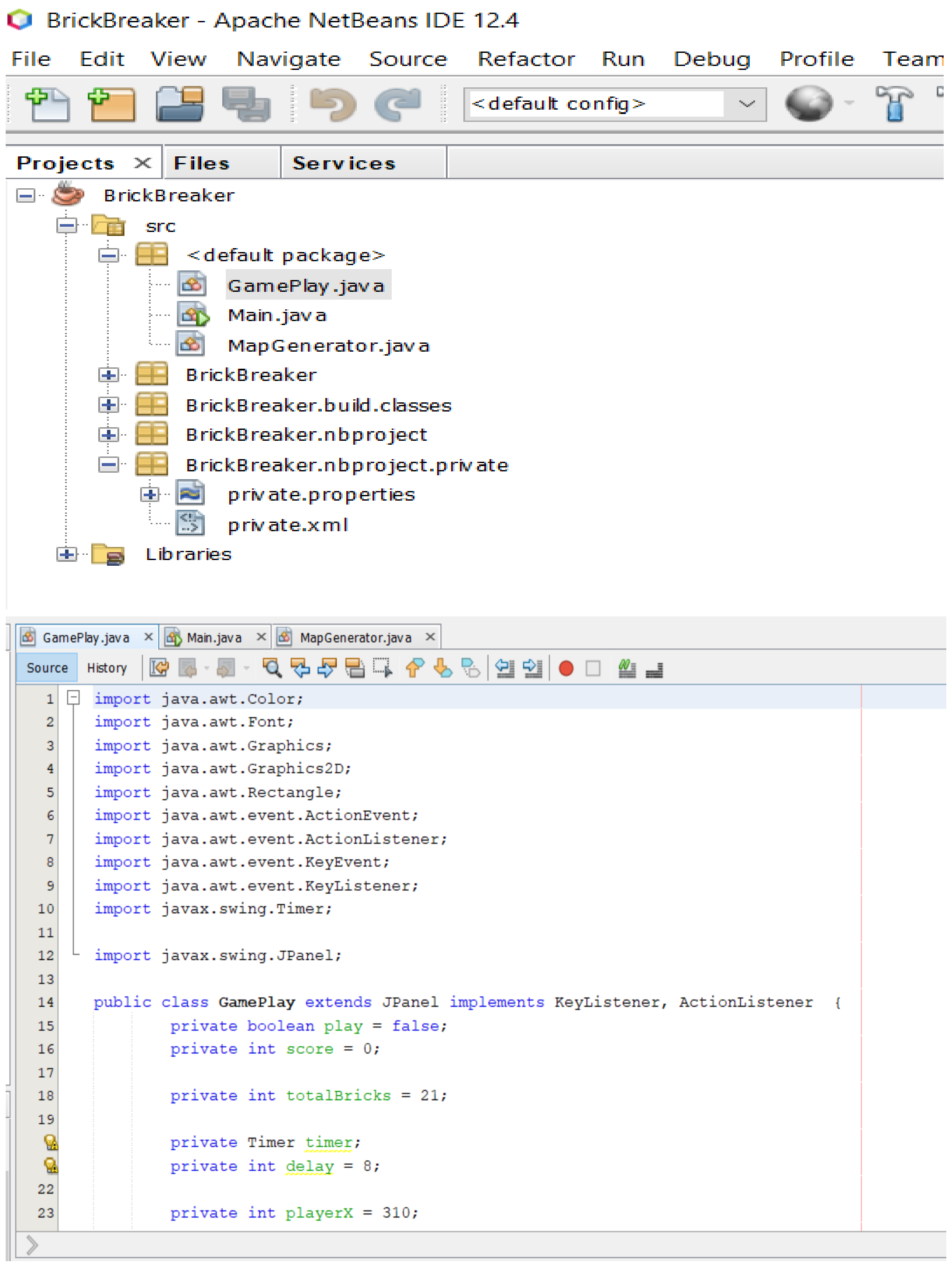
Software will count for the scores and calculate the bonus points gotten by how many bricks broken in a row and other effects. The score will be displayed on the up right corner of the screen.

GAME Features: -

- Easy to play.
- User-friendly interface.
- Customizable.
- Very smooth touch control

Steps to make a brick breaker game: -

- Create a new project and name it Brick Breaker. In source create 3 new class and name it **“Main”, “Gameplay” and “MapGenerator”.**



```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.Timer;

import javax.swing.JPanel;

public class GamePlay extends JPanel implements KeyListener, ActionListener  {
        private boolean play = false;
        private int score = 0;

        private int totalBricks = 21;

        private Timer timer;
        private int delay = 8;

        private int playerX = 310;
```

Main Class Code Explanation:

In the main class, we have the main method like public static void main (String[] args) are available. We'll define an object that will have the properties of JFrame. And later, we have set the properties of this frame using methods such as setBounds(), setTitle(), setResizable() etc. we've added the methods and functions of the Gameplay class in the main class.

Main class (code):

```
import javax.swing.JFrame;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        JFrame obj = new JFrame();
        GamePlay gamePlay = new GamePlay();
        obj.setBounds(10, 10, 700, 600);
        obj.setTitle("Brick Breaker");
        obj.setResizable(false);
        obj.setVisible(true);
        obj.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        obj.add(gamePlay);

    }
}
```

Gameplay class Explanation:

Following libraries are used in gameplay class:

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Rectangle;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import javax.swing.Timer;

import javax.swing.JPanel;

- In the gameclass since,we need a container that can store and organize different components, we have used JPanel which is found in the Java Swing Package. We've ensured that this class inherits the attributes of the JPanel container by using the keyword ‘extends’.
- We've also ensured to implement the KeyListener and ActionListener interfaces by using the ‘implements’ keyword. KeyListener in Java takes care of all the operations, or any actions related to the keyboard such as pressing the left button to move and so on. The ActionListener interface in Java is automatically notified when you click or interact with a defined object within the bounds of the frame.
- After all, this we have initialized and defined all the parameters which will be required to play and create this game.
- We've used the setFocusable() method from the component class in Swing package to set certain components on focus.

- The setFocusTraversalKeysEnabled() helps you to decide whether or not to focus the traversal keys, and it only accepts Boolean parameters. Since we require a timer, we've also designed a code to create the timer.

- The Timer class can be obtained from the javax.swing.Timer package. The public void paint(Graphics g)is a class that contains the paint interface from the Graphics class. it contains and describes the graphic environment in which the game is running.

- In this method, we have set the graphics of objects such as the background, ball, paddle, etc.

- Here, we've set the color of the background as black and we've made the frame rectangular. The fillRect() method requires 4 arguments (one x, one y coordinate, height, and width as arguments).

- We have enabled the draw() method to invoke the Graphics object. Next, we've set the color of the borders as well as the bounds of the rectangle that will be reachable to the ball to intercept when we actually play the game.

- We've also set the font style, font size as well as color of the font. And we've used the drawString() method to display the score in the window. We've created the paddle as an object and we set the color of it. The paddle is a rectangular bar that intersects the ball when the player moves the keys accordingly.

- That is why we are sing the fillRect() method. Lastly, we've designated the ball in the shape of the oval using the fillOval method. And we've set the color of the ball.

- In the game, we need to ensure that if all the bricks are broken and none are left, the game should terminate. So, for that we're using the if-else construct to find whether the player has lost or won the game. We've also included the option to press enter to restart the game.

- Next, let’s use the override method which allows the child class to facilitate a unique implementation of a method that is available in the parent classes. We are using the override the actionPerformed() method specifically invoke the method when you click on a specific object.
- To detect the intersection of 2 objects within the window of the game, we've created a structure using if -else and for loop to find the course of action according to the position and direction of the ball.
- We've used the labeling method with the break statement to exit the loop once a condition has been met. After breaking out of the loop, we used the repaint() method to redraw all the components again.
- We can keep the override methods as it is since we don’t have any functions that will require this implementation. However, removing will cause an error due to disruption of the method structure.
- After that, let’s construct the events and methods in the keypressed method using override. here, we have created an if-else loop to detect the use of the key that is used to enable motion. In the structure, it's ensured that if the keyPressed method detects that right key is being pressed, then the object moves to the right. Same for the left key. if enter is being pressed, the game will restart itself. we'll generate a new map and hence the use of repaint method.

Gameplay Class

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.Timer;
import javax.swing.JPanel;

public class GamePlay extends JPanel implements KeyListener,
ActionListener  {
        private boolean play = false;
        private int score = 0;

        private int totalBricks = 21;

        private Timer timer;
        private int delay = 8;

        private int playerX = 310;

        private int ballposX = 120;
        private int ballposY = 350;
private int ballXdir = -1;
private int ballYdir = -2;

private MapGenerator map;

public GamePlay() {
	map = new MapGenerator(3, 7);
	addKeyListener(this);
	setFocusable(true);
	setFocusTraversalKeysEnabled(false);
	timer = new Timer(delay, this);
	timer.start();



}
public void paint(Graphics g) {
	g.setColor(Color.white);
	g.fillRect(1, 1, 692, 592);
	map.draw((Graphics2D)g);




	g.setColor(Color.yellow);
	g.fillRect(0, 0, 3, 592);
```

```
g.fillRect(0, 0, 692, 3);
g.fillRect(691, 0, 3, 592);
g.setColor(Color.blue);
g.fillRect(playerX, 550, 100, 8);
g.setColor(Color.green);
g.fillOval(ballposX, ballposY, 20, 20);
g.setColor(Color.black);
g.setFont(new Font("serif", Font.BOLD, 25));
g.drawString("" + score, 590, 30);
if (totalBricks <= 0) {
        play = false;
        ballXdir = 0;
        ballYdir = 0;
        g.setColor(Color.green);
        g.setFont(new Font("serif", Font.BOLD, 30));
        g.drawString("You Won, Score: " + score, 190, 300);

        g.setFont(new Font("serif", Font.BOLD, 20));
        g.drawString("Press Enter to Restart.", 230, 350);


}




if(ballposY > 570) {
        play = false;
```

```
                        ballXdir = 0;
                        ballYdir = 0;
                        g.setColor(Color.red);
                        g.setFont(new Font("serif", Font.BOLD, 30));
                        g.drawString("Game Over, Score: " + score, 190, 300);

                        g.setFont(new Font("serif", Font.BOLD, 20));
                        g.drawString("Press Enter to Restart.", 230, 350);
                }



                g.dispose();

        }

        @Override
        public void actionPerformed(ActionEvent arg0) {
                // TODO Auto-generated method stub
                timer.start();
                if(play) {
                        // Ball - Pedal  interaction
                        if(new Rectangle(ballposX, ballposY, 20, 20).intersects(new 
Rectangle(playerX, 550, 100, 8))) {
                                ballYdir = - ballYdir;
                        }
```

```
            for( int i = 0; i<map.map.length; i++) {
                for(int j = 0; j<map.map[0].length; j++) {
                    if(map.map[i][j] > 0) {
                        int brickX = j*map.brickWidth + 80;
                        int brickY = i*map.brickHeight + 50;
                        int brickWidth= map.brickWidth;
                        int brickHeight = map.brickHeight;

                        Rectangle rect = new Rectangle(brickX,
brickY, brickWidth, brickHeight);
                        Rectangle ballRect = new
Rectangle(ballposX, ballposY, 20,20);
                        Rectangle brickRect = rect;

                        if(ballRect.intersects(brickRect) ) {
                            map.setBrickValue(0, i, j);
                            totalBricks--;
                            score+=5;

                            if(ballposX + 19 <= brickRect.x ||
ballposX +1 >= brickRect.x + brickRect.width)
                                ballXdir = -ballXdir;
                             else {
                                ballYdir = -ballYdir;
                            }
                        }
                    }
```

```
				}
			}



			ballposX += ballXdir;
			ballposY += ballYdir;
			if(ballposX < 0) {
				ballXdir = -ballXdir;
			}
			if(ballposY < 0) {
				ballYdir = -ballYdir;
			}
			if(ballposX > 670) {
				ballXdir = -ballXdir;
			}
		}
		repaint();
	}



	@Override
	public void keyTyped(KeyEvent arg0) {
		// TODO Auto-generated method stub

	}
```

```
@Override
public void keyPressed(KeyEvent arg0) {
        // TODO Auto-generated method stub
        if(arg0.getKeyCode() == KeyEvent.VK_RIGHT) {
                if(playerX >= 600) {
                        playerX = 600;
                } else {
                        moveRight();


                }
        }
        if(arg0.getKeyCode() == KeyEvent.VK_LEFT) {
                if(playerX < 10) {
                        playerX = 10;
                } else {
                        moveLeft();
                }
        }
        if(arg0.getKeyCode() == KeyEvent.VK_ENTER) {
                if(!play) {
                        play = true;
                        ballposX = 120;
                        ballposY = 350;
                        ballXdir = -1;
                        ballYdir = -2;
                        score = 0;
                totalBricks = 21;
                map = new MapGenerator(3,7);
                repaint();
            }
        }
    }
        public void moveRight() {
            play = true;
            playerX += 20;
        }
        public void moveLeft() {
            play = true;
            playerX -= 20;
        }
    @Override
    public void keyReleased(KeyEvent arg0) {
        // TODO Auto-generated method stub
    }


}
```

<u>MapGenerator Class Explanation:</u>

Following libraries are used:

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Graphics2D;

- Here, we’re also using the BasicStroke class in the awt package that is used to produce certain attributes along with the Graphics2D class such as dash, join cap, etc. let’s create a class named ‘mapgenerator’. We'll be developing this class to contain objects and attributes such as brick width, brick height etc. we've also declared an array named ‘map’, which will contain all the bricks.

- So, inside the constructor named ‘mapgenerator’, we will receive the number of rows and columns that notify and help us understand about how many rows and columns should be generated in order to accommodate a given number of bricks.

- We have instantiated a 2D array with values of ‘row’ and ‘col’ and after this, we have created a for loop to iterate through the number of rows and columns. Also, we have added 1 inside each element of the 2D array so that it will notify us when the ball has not intersected with brick and it will show up in the panel.

- Once the constructor is called, we need to have a function for drawing those particular bricks. We will do that by using the draw() method. Also, note that the draw function should contain the Graphics2D object. When the draw function with Graphics objects is called the bricks will be drawn on the particular positions where there’s a value of 1 identified on the array.

- To accomplish this, we will use the for loop to iterate once again. Within the for loop, we will need to check if the particular value is greater than not. if it is greater than zero, it creates the bricks. We have also set the color of the bricks and used the fillRect method to fill the rectangle at the particular positions.

- Finally, we need to create a function that will give us a value when the ball intersects the brick. We know that the total number of bricks is 21. When the function value attains zero, the game will terminate and display a ‘ you won!’ message. The following code ensures that the value is accordingly interpreted by the 2D array too.

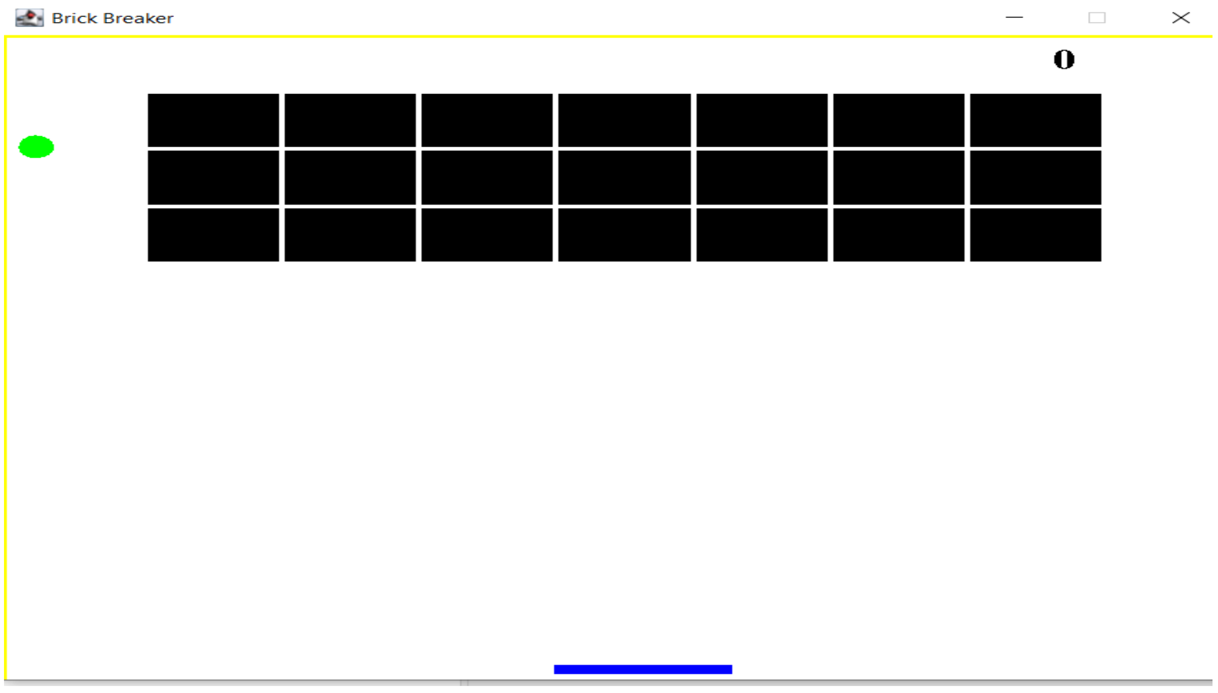
MapGenerator Class

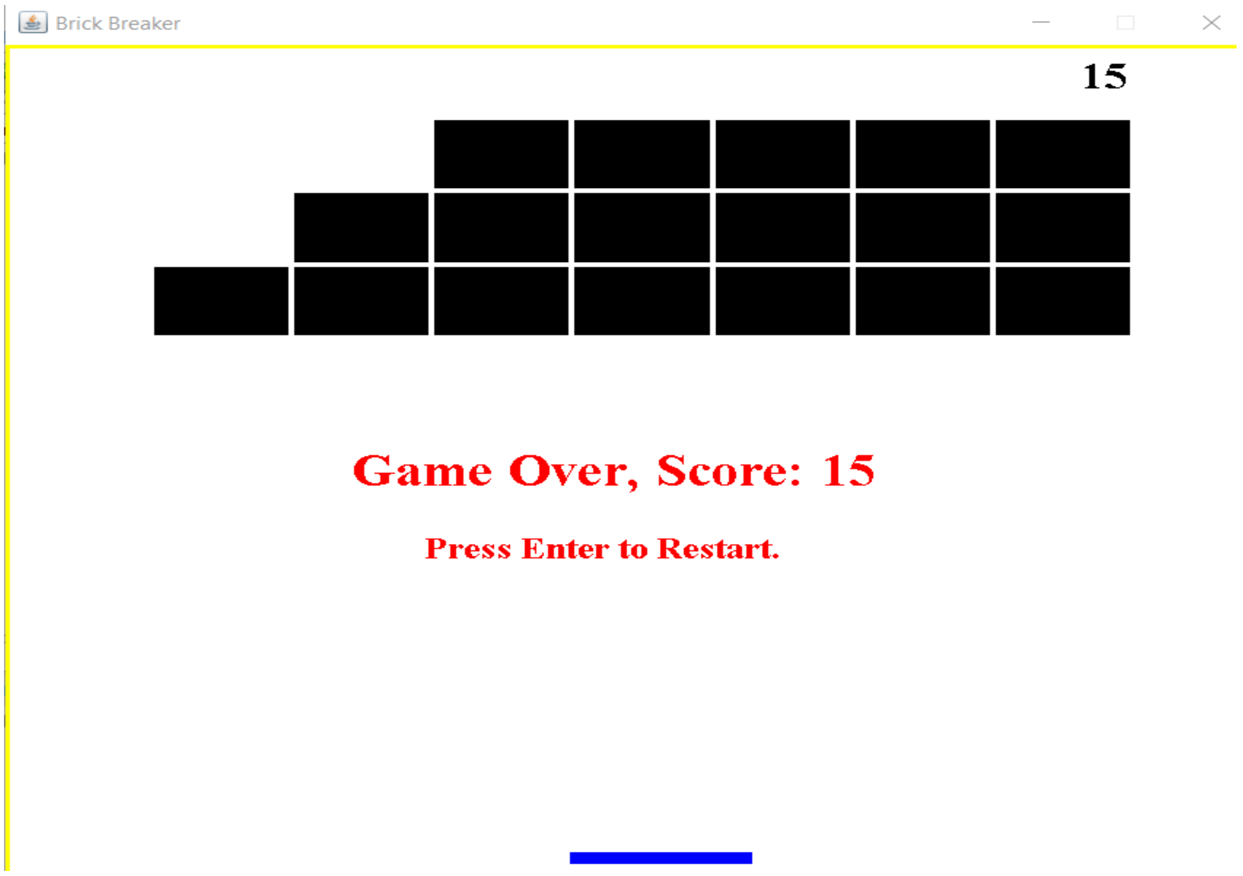
```
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;

public class MapGenerator {

        public int map [][];
        public int brickWidth;
        public int brickHeight;
        public MapGenerator(int row, int col) {
                map = new int [row][col];
                for (int i = 0; i < map.length; i++) {
                        for (int j=0; j< map[0].length;j++) {
                                map[i][j] = 1;
                        }
                }
                brickWidth = 540/col;
        brickHeight = 150/row;
    }

    public void draw(Graphics2D g) {
        for (int i = 0; i < map.length; i++) {
            for (int j=0; j< map[0].length;j++) {
                if(map[i][j] > 0) {
                    g.setColor(Color.black);
                    g.fillRect(j*brickWidth + 80, i*brickHeight +
50, brickWidth, brickHeight);

                    g.setStroke(new BasicStroke(3));
                    g.setColor(Color.white);
                    g.drawRect(j*brickWidth + 80, i*brickHeight +
50, brickWidth, brickHeight);
                }
            }
        }
        }
    public void setBrickValue(int value, int row, int col) {
        map[row][col] = value;
    }
}
```

- **Final step is to run all the codes and the game is ready to play**

PRIMARY GOALS

- The player should be able to start the game in the terminal by pressing the Enter key.
- The player should be able to move all the paddle sideways using the leftwards and the rightwards arrow on keyboard.
- Once the player loses the ball and touches the bottom of the screen, the game ends.
- Breaking each brick should provide the player a certain no. of points. Lets say each brick contains 5 points, if a player breaks 10 such bricks, he gets 50 Points.
- Once the game ends, the terminal will present the final score of the player and give him the option to restart the game again.
- This game helped us to test our java skills and implement it in the form of a game.





Game Over, Score: 15

Press Enter to Restart.

REFRENCES:-

- https://www.slideshare.net/waqashashmi09/brick-breaker-70465909
- https://zetcode.com/javagames/breakout/
- https://www.youtube.com/watch?v=K9qMm3JbOH0
- https://www.crio.do/projects/java-breakout-game/