Roll No and Name -20BCE526_Jainil Solanki, 20BCE505_Dev Delvadiya

Batch-D2D

Subject-OOP

Project Title-Brick Breaker Game

Code:

We have created three files for this project each file is defining the separate functionalities of game.

Gameplay.java

☐ This file defines the control of games the positioning of the map and scoreboard etc.

```java
import java.util.*;
import java.awt.event.*;

import javax.swing.*;

import java.awt.*;

import javax.swing.*;
import javax.swing.Timer;

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

        private int totalBricks = 48;

        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;
```

```java
    public Gameplay()
    {
            map = new MapGenerator(4, 12);
            addKeyListener(this);
            setFocusable(true);
            setFocusTraversalKeysEnabled(false);
timer=new Timer(delay,this);
            timer.start();
    }

    public void paint(Graphics g)
    {
            // background
            g.setColor(Color.black);
            g.fillRect(1, 1, 692, 592);

            // drawing map
            map.draw((Graphics2D) g);

            // borders
            g.setColor(Color.yellow);
            g.fillRect(0, 0, 3, 592);
            g.fillRect(0, 0, 692, 3);
            g.fillRect(691, 0, 3, 592);

            // the scores
            g.setColor(Color.white);
            g.setFont(new Font("serif",Font.BOLD, 25));
            g.drawString(""+score, 590,30);

            // the paddle
            g.setColor(Color.green);
            g.fillRect(playerX, 550, 100, 8);

            // the ball
            g.setColor(Color.yellow);
            g.fillOval(ballposX, ballposY, 20, 20);

            // when you won the game
            if(totalBricks <= 0)
            {
                    play = false;
        ballXdir = 0;
            ballYdir = 0;
        g.setColor(Color.RED);
        g.setFont(new Font("serif",Font.BOLD, 30));
```

```java
        g.drawString("You Won", 260,300);

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

                // when you lose the game
                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, Scores: "+score, 190,300);

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

                g.dispose();
    }

    public void keyPressed(KeyEvent e)
    {
                if (e.getKeyCode() == KeyEvent.VK_RIGHT)
                {
                        if(playerX >= 600)
                        {
                                playerX = 600;
                        }
                        else
                        {
                                moveRight();
                        }
}

                if (e.getKeyCode() == KeyEvent.VK_LEFT)
                {
                        if(playerX < 10)
                        {
                                playerX = 10;
                        }
                        else
```

```java
                               {
                                      moveLeft();
                               }
               }
                       if (e.getKeyCode() == KeyEvent.VK_ENTER)
                       {
                               if(!play)
                               {
                                      play = true;
                                      ballposX = 120;
                                      ballposY = 350;
                                      ballXdir = -1;
                                      ballYdir = -2;
                                      playerX = 310;
                                      score = 0;
                                      totalBricks = 21;
                                      map = new MapGenerator(3, 7);

                                      repaint();
                               }
               }
       }

       public void keyReleased(KeyEvent e) {}
       public void keyTyped(KeyEvent e) {}

       public void moveRight()
       {
               play = true;
               playerX+=20;
       }

       public void moveLeft()
       {
               play = true;
               playerX-=20;
       }

       public void actionPerformed(ActionEvent e)
       {
               timer.start();
               if(play)
               {
                       if(new    Rectangle(ballposX,    ballposY,    20,    20).intersects(new
       Rectangle(playerX, 550, 30, 8)))
                       {
```

```java
                                        ballYdir = -ballYdir;
                                        ballXdir = -2;
                    }
                    else    if(new    Rectangle(ballposX,    ballposY,    20,    20).intersects(new
Rectangle(playerX + 70, 550, 30, 8)))
                    {
                                        ballYdir = -ballYdir;
                                        ballXdir = ballXdir + 1;
                    }
                    else    if(new    Rectangle(ballposX,    ballposY,    20,    20).intersects(new
Rectangle(playerX + 30, 550, 40, 8)))
                    {
                                        ballYdir = -ballYdir;
                    }

                    // check map collision with the ball
                    A: 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)
                                    {
                                            //scores++;
                                            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);
                                                    score+=5;
                                                    totalBricks--;

                                                    // when ball hit right or left of brick
                            if(ballposX + 19 <= brickRect.x || ballposX + 1 >= brickRect.x +
brickRect.width)
                                    {
                                                    ballXdir = -ballXdir;
                                    }
```

```
                                // when ball hits top or bottom of brick
                                else
                                {
                                        ballYdir = -ballYdir;
                                }

                                break A;
                                }
                        }
                }

                        ballposX += ballXdir;
                        ballposY += ballYdir;

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

                        repaint();
                }
        }
}
```

Main.java

☐ This is the main file which is used for creating objects of the other files and calling their functions.

import java.awt.Color;

import javax.swing.JFrame;

```
public class Main {
        public static void main(String[] args) {
                JFrame obj=new JFrame();
                Gameplay gamePlay = new Gameplay();
```

```
                obj.setBounds(10, 10, 700, 600);
                obj.setTitle("Breakout Ball");
                obj.setResizable(false);
                obj.setVisible(true);
                obj.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                obj.add(gamePlay);
            obj.setVisible(true);


        }

}
```

MapGenerator.java

☐ This file is for creating map and generating yellow blocks.

```java
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.white);
                                        g.fillRect(j * brickWidth + 80, i * brickHeight + 50,
brickWidth, brickHeight);

                                        // this is just to show separate brick, game can still run
without it
                                        g.setStroke(new BasicStroke(3));
                                        g.setColor(Color.black);
                                        g.drawRect(j * brickWidth + 80, i * brickHeight + 50,
brickWidth, brickHeight);
                                }
                        }
                }
        }

        public void setBrickValue(int value, int row, int col)
        {
                map[row][col] = value;
        }
}
```
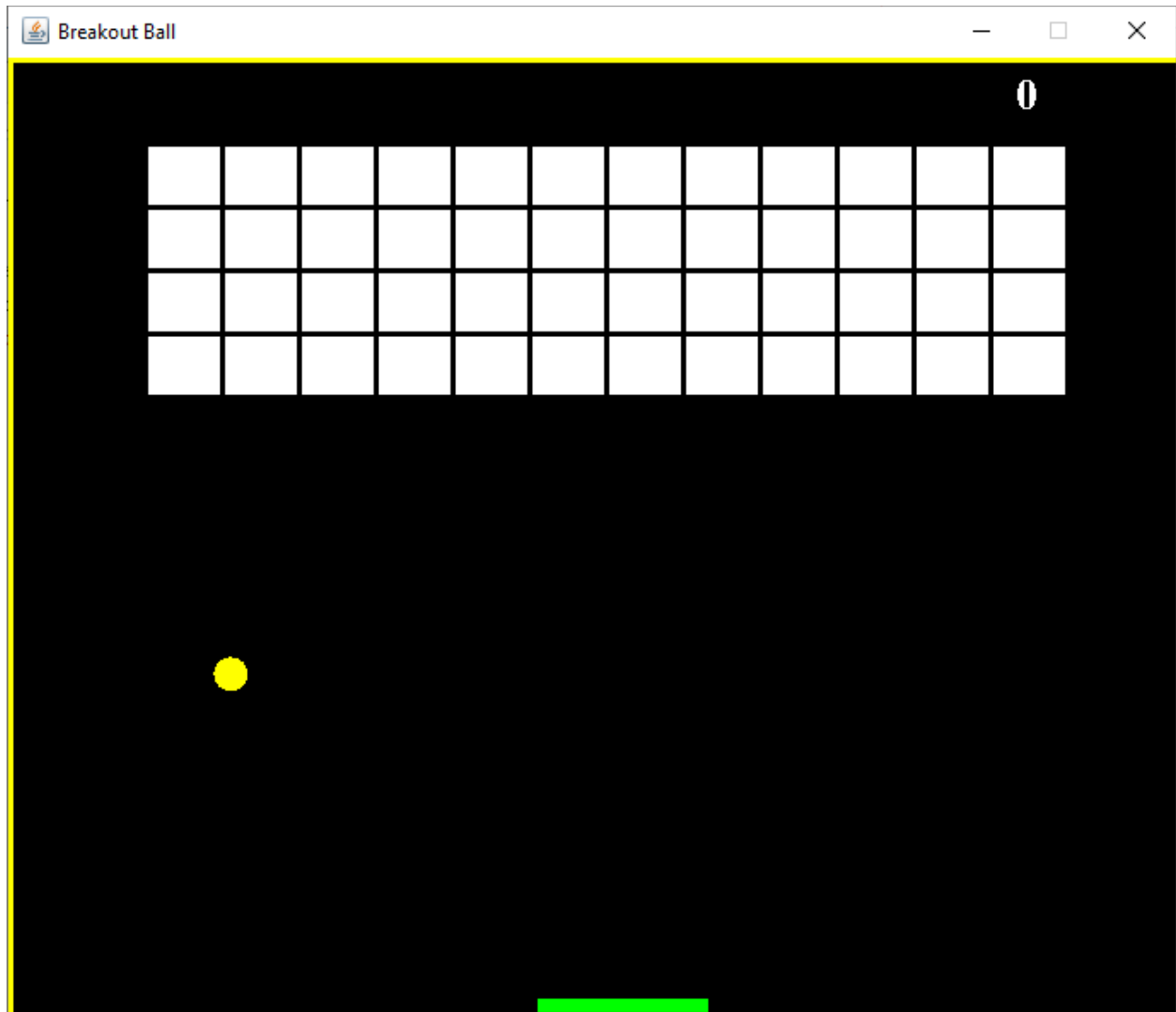
## Controls:

To start the game you first need to press Enter. You can use < (Left arrow) key to
move the bar to left and > (Right arrow) key to move the bar to right. The game will
end when you reach the score of 105 or if the ball falls down then it will show a
popup as press Enter to replay the game you can press Enter to replay the game.
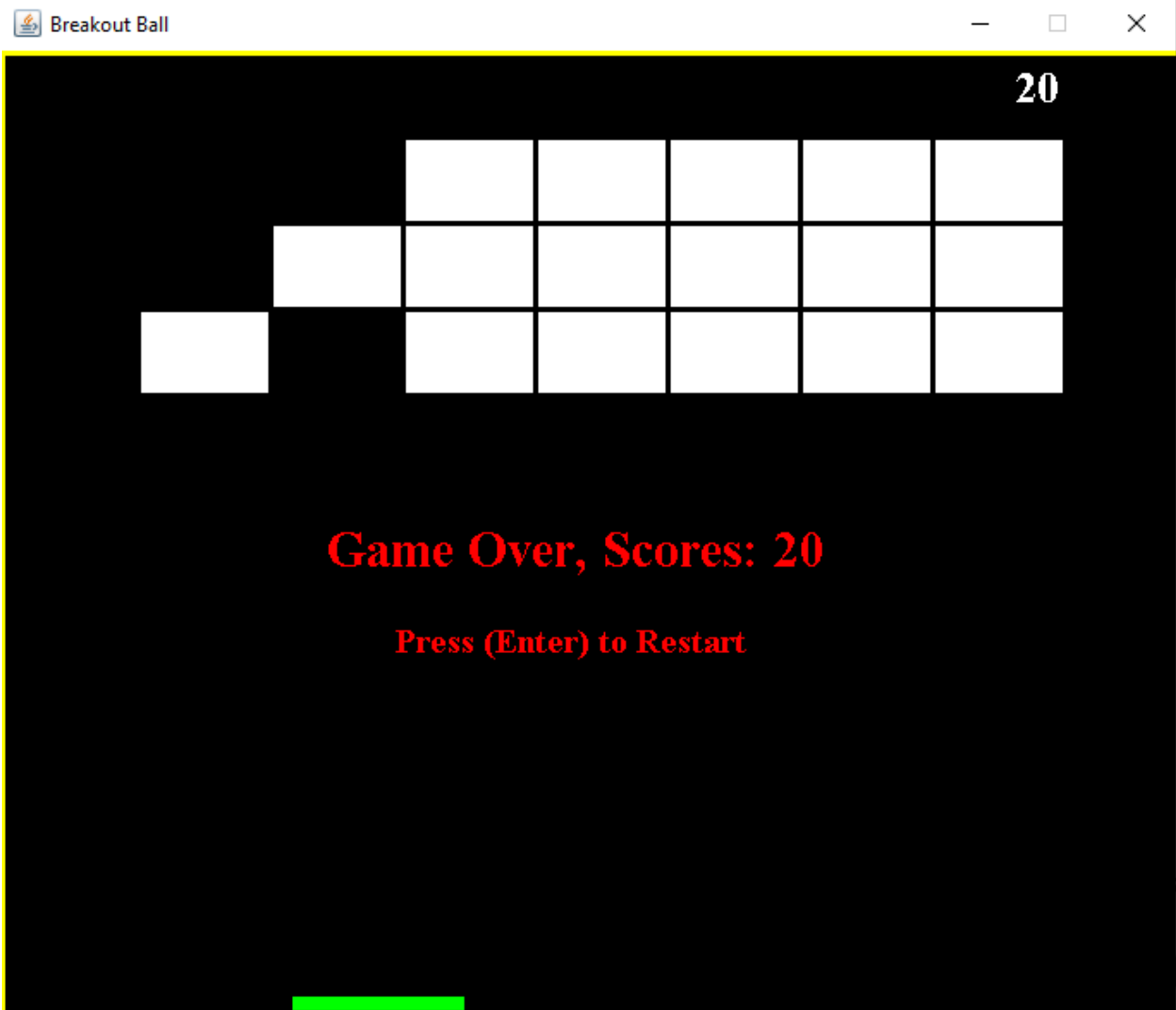
Output:

Before beginning the game.

After Pressing the Enter
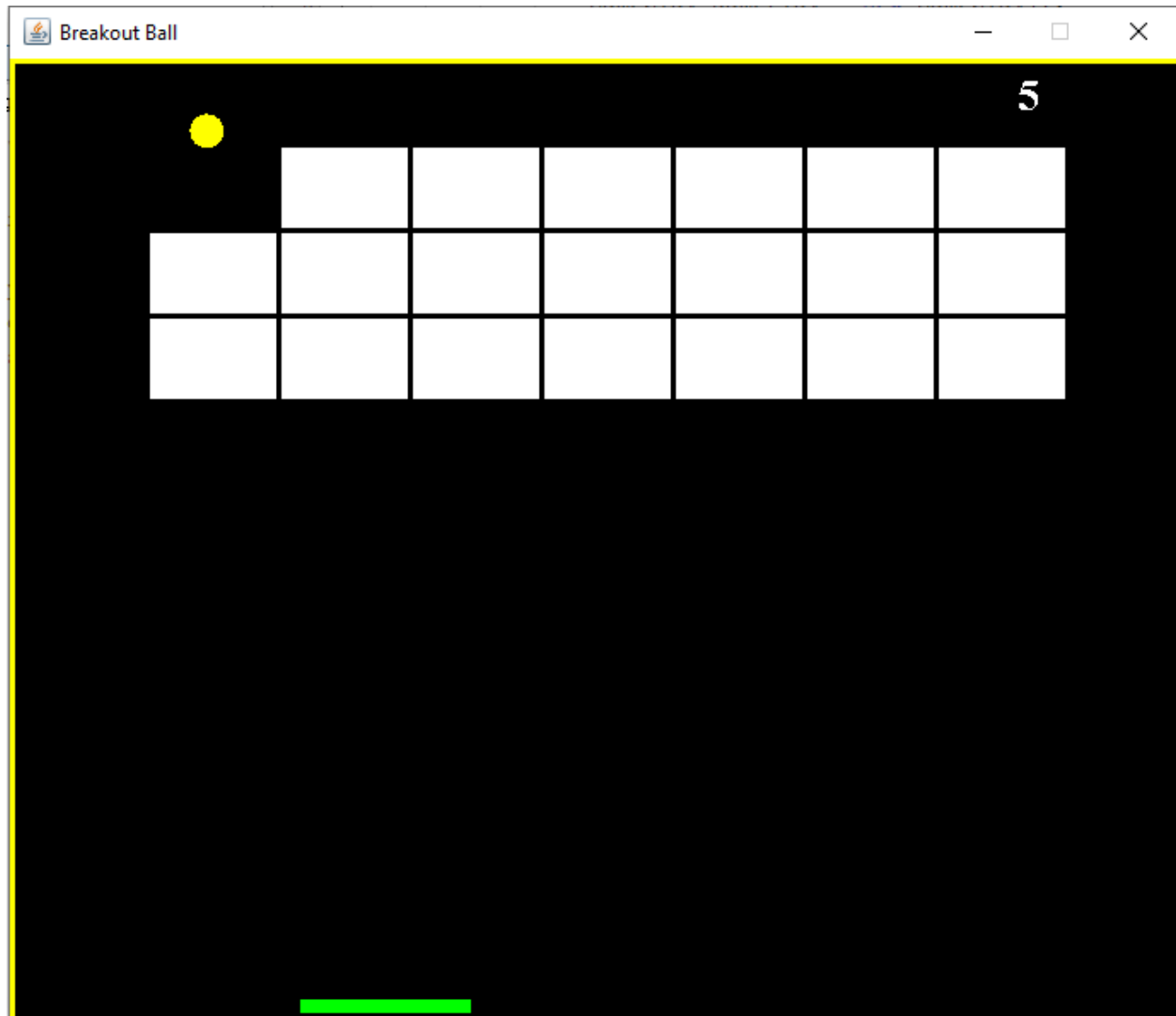
If the ball falls down then it will show:

If you win the game then: