PROJECT REPORT

on

BREAKOUT BALL GAME

Submitted by

GARIMA ARORA(1032191531**)**

ATUL KUMAR(1032191844**)**

JAY SRIVASTAVA(1032191969**)**

in

Object Oriented Programming

SY BTech

Under the Guidance of

Prof MRUNAL ANNADATE

School of Electronics & Communication Engineering

Dr. Vishwanath Karad

MIT WORLD PEACE UNIVERSITY, PUNE.

[2020-2021]

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Table of Contents

1. Introduction

Java might sound familiar if you're a fan of video games. Back in 2011, Minecraft was created using Java. Minecraft is one of the one of the best-selling video games of all time and one of the most popular games that run on Java. In 2014, Microsoft bought Minecraft and Mojang for $2.5 billion. This might reasonably lead you to believe that making games with Java is a good idea.

Breakout Ball is game in which a layer of bricks lines the top of the screen and the goal is to destroy them all. A ball moves straight around the screen, bouncing off the top and two sides of the screen. When a brick is hit, the ball bounces back and the brick is destroyed..

1.1 Aim and Objectives

To Create a Breakout Ball game in Java using concepts of classes , Oops, GUI. In an java game, the player will be able to break statically located bricks placed on the top of the screen using a ball. The ball keeps on moving from top to bottom . A paddle at the bottom can be used to change the direction of the ball touching the ground . If ball touches the ground , the player will lose the game but if player breaks all the bricks then the player is declared as winner.

- Game should start by pressing Enter Key.
- Movement of paddle through left and right arrow keys.
- Game should end once ball touches the ground.
- 5 points assigned after breaking each brick.
- Once game ends , option to restart should be available

2. Methodology

- **The JFrame development**

The building of this game starts by first building a JFrame that consist of Main function(i.e public static void main(String [] args)). Then in the JFrame function we will be setting the boundary constraints. Then after setting the boundaries , we will be making the Heading of our game using setTitle() function.

Note: for using standard functions of J frame, import the file

```
import javax.swing.JFrame;
```

- **The Gameplay Class development**

In this particular class we will be inheriting the concepts of JPanel Class and the interfaces ActionListener, KeyListener classes. Here in this particular class we are defining all the parameters that are required in our game development like the scores, the paddle position , the initial paddle position, it's color , shape , size , and the condition when it intersects with the ball and the condition when the ball intersects with the Bricks, etc.

Classes included:

import java.awt.Color
import java.awt.event.KeyListner
import java.awt.event.ActionListner
import javax.swing.Timer

```
import.java.awt.Graphics
```

- **The Map Generator Class**:
  In this particular class , we will define the brick size, shape, using the concepts of array. The logic that we have used in our project is that when the Brick gets intercepted by the ball then the position at where the brick was present toggles to 0 using a separate function in MapGenerator class.

2.3 System Requirements

Windows Requirement

- Eclipse IDE
- JDK
- MinGW

Eclipse IDE

The Eclipse IDE is famous for our Java Integrated Development Environment (IDE), but we have a number of pretty cool IDEs, including our C/C++ IDE, JavaScript/TypeScript IDE, PHP IDE, and more. You can easily combine multiple languages support and other features into any of our default packages, and the Eclipse Marketplace allows for virtually unlimited customization and extension.

The initial codebase originated from IBM VisualAge.[7] The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Since the introduction of the OSGi implementation (Equinox) in version 3 of Eclipse, plug-ins can be plugged-stopped dynamically and are termed (OSGI) bundles.[8]

Eclipse software development kit (SDK) is free and open-source software, released under the terms of the Eclipse Public License, although it is incompatible with the GNU General Public License.[9] It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

Figure 1

Java JDK

The Java Development Kit (JDK) is an implementation of either one of the Java Platform, Standard Edition, Java Platform, Enterprise Edition, or Java Platform, Micro Edition platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, macOS or Windows. The JDK includes a private JVM and a few other resources to finish the development of a Java application. Since the introduction of the Java platform, it has been by far the most widely used Software Development Kit The JDK is available for 64-bit x64 macOS (and that version also works with Rosetta 2), while an early access build (developer preview) from Microsoft is also available to support recent Apple M1 Macs. The JDK forms an extended subset of a software development kit (SDK). It includes "tools for developing, debugging, and monitoring Java

applications". Oracle strongly suggests to now use the term *JDK* to refer to the Java SE Development Kit. The Java SE SDK is available with or without the JDK, by which they specifically mean the Java SE 7 JDK.

Figure 2

MinGW

MinGW "Minimalist GNU for Windows", formerly mingw32, is a free and open source software development environment to create Microsoft Windows applications. The development of the MinGW project has been forked with the creation in 2005–2008 of an alternative project called Mingw-w64. MinGW includes a port of the GNU Compiler Collection (GCC), GNU Binutils for Windows (assembler, linker, archive manager), a set of freely distributable Windows specific header files and static import libraries which enable the use of the Windows API, a Windows native build of the GNU Project's GNU Debugger, and miscellaneous utilities.

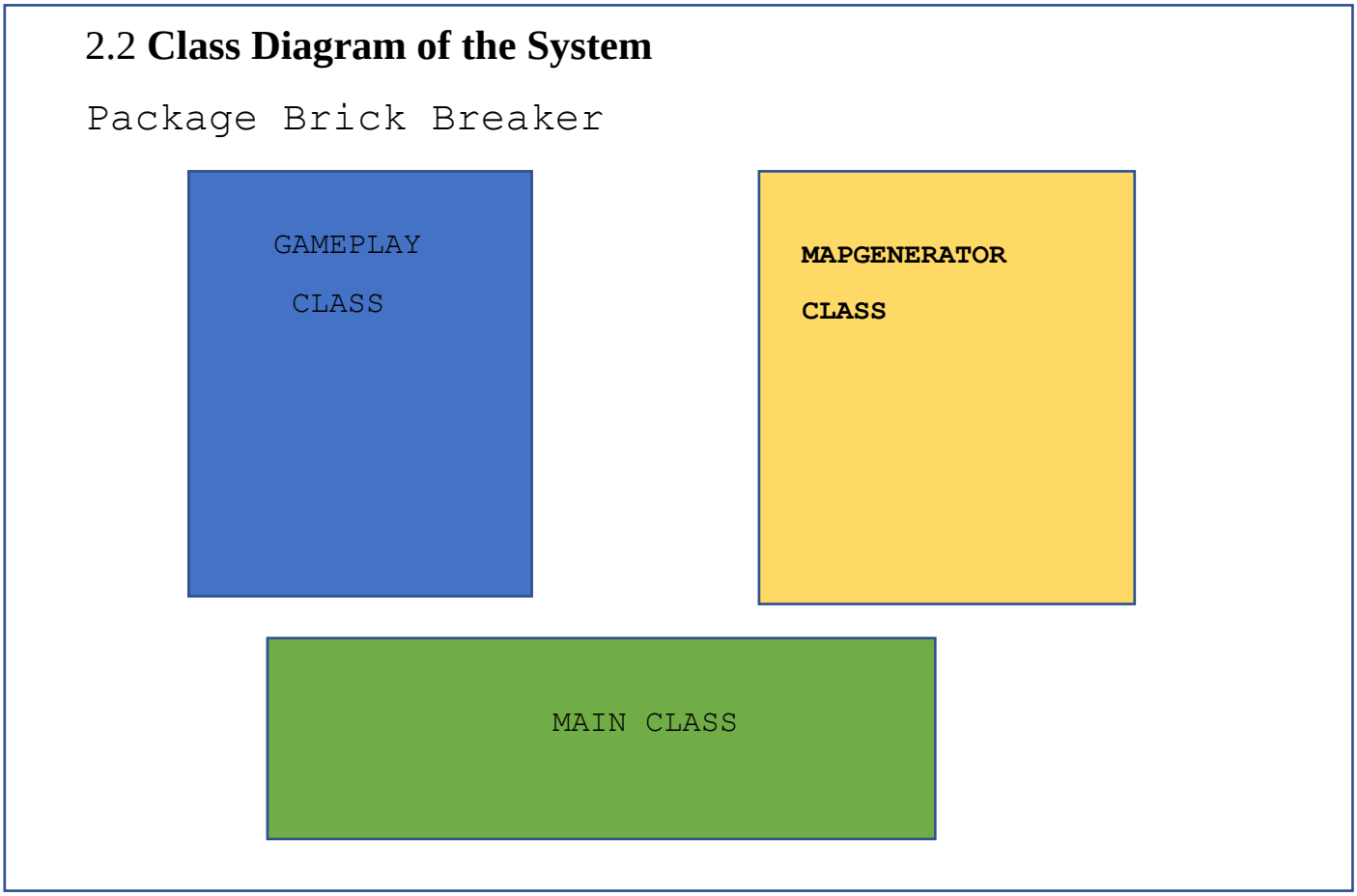
MinGW does not rely on third-party C runtime dynamic-link library (DLL) files, and because the runtime libraries are not distributed using the GNU General Public License (GPL), it is not necessary to distribute the source code with the programs produced, unless a GPL library is used elsewhere in the program. MinGW can be run either on the native Microsoft Windows platform, cross-hosted on Linux (or other Unix), or "cross-native" on Cygwin. Although programs produced under

MinGW are 32-bit executables, they can be used both in 32 and 64-bit versions of Windows.

Most languages supported by GCC are supported on the MinGW port as well. These include C, C++, Objective-C, Objective-C++, Fortran, and Ada. The GCC runtime libraries are used (libstdc++ for C++, libgfortran for Fortran) etc.

MinGW links by default to the Windows OS component library MSVCRT, which is the C library that Visual C++ version 6.0 linked to (the initial target was CRTDLL), which was released in 1998 and therefore does not include support for C99 features, or even all of C89. While targeting MSVCRT yields programs that require no additional runtime redistributables to be installed, the lack of support for C99 has caused porting problems, particularly where printf-style conversion specifiers are concerned. These issues have been partially mitigated by the implementation of a C99 compatibility library, libmingwex, but the extensive work required is far from complete and may never be fully realized.[10] Mingw-w64 has resolved these issues, and provides fully POSIX compliant printf functionality.

2.2 **Class Diagram of the System**



- Main() Class:

```
package brickBreaker;

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);


	}

}
```

- Gameplay Class()

```
package brickBreaker;

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.JPanel;
import javax.swing.Timer;

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


private int totalBricks = 35;

private Timer timer;
private int delay= 10;
```

```
private int playerX =120;  // position of slider
private int ballposX=150;
private int ballposY= 350;
private int ballXdir=-1;
private int ballYdir=-2;
private MapGenerator map;

public Gameplay() {
map=new MapGenerator(5,7);

addKeyListener(this);
setFocusable(true);
setFocusTraversalKeysEnabled(false);
timer =new Timer(delay,this);
timer.start();
}
public void paint(Graphics g) {

//background
g.setColor(Color.cyan);
g.fillRect(1, 1,692,592);


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

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

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


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


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

if(totalBricks<=0) {
play=false;
ballXdir=0;
ballYdir=0;
g.setColor(Color.RED);
g.setFont(new Font("serif",Font.BOLD,30));
g.drawString("YOU WON!: ",190,30);
```

```
g.setFont(new Font("serif",Font.BOLD,30));
g.drawString("Press Enter to Restart ",190,350);

}


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: ",190,30);

g.setFont(new Font("serif",Font.BOLD,30));
g.drawString("Press Enter to Restart ",190,350);

}
g.dispose();

}

@Override
public void actionPerformed(ActionEvent e) {
timer.start();

if(play) {

if(new               Rectangle(ballposX,ballposY,20,20).intersects(new
Rectangle(playerX,550,100,8))) {

ballYdir=-ballYdir;
}

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) {
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;
}
break A;
}
}
}
}

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 e) {
// TODO Auto-generated method stub

}
@Override
public void keyReleased(KeyEvent e) {
// TODO Auto-generated method stub

}
@Override
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 {

moveLeft();
}
```

```
        }
        if(e.getKeyCode()== KeyEvent.VK_ENTER) {
        if(!play) {
        play=true;
        ballposX=120;
        ballposY=350;
        ballXdir=-1;
        ballYdir=-2;
        playerX=310;
        score=0;
        totalBricks=35;
        map=new MapGenerator(5,7);


        repaint();
        }

        }


        }
        public void moveRight() {
        play=true;
        playerX+=10;

        }

        public void moveLeft() {
        play=true;
        playerX-=10;
        }



        }
```

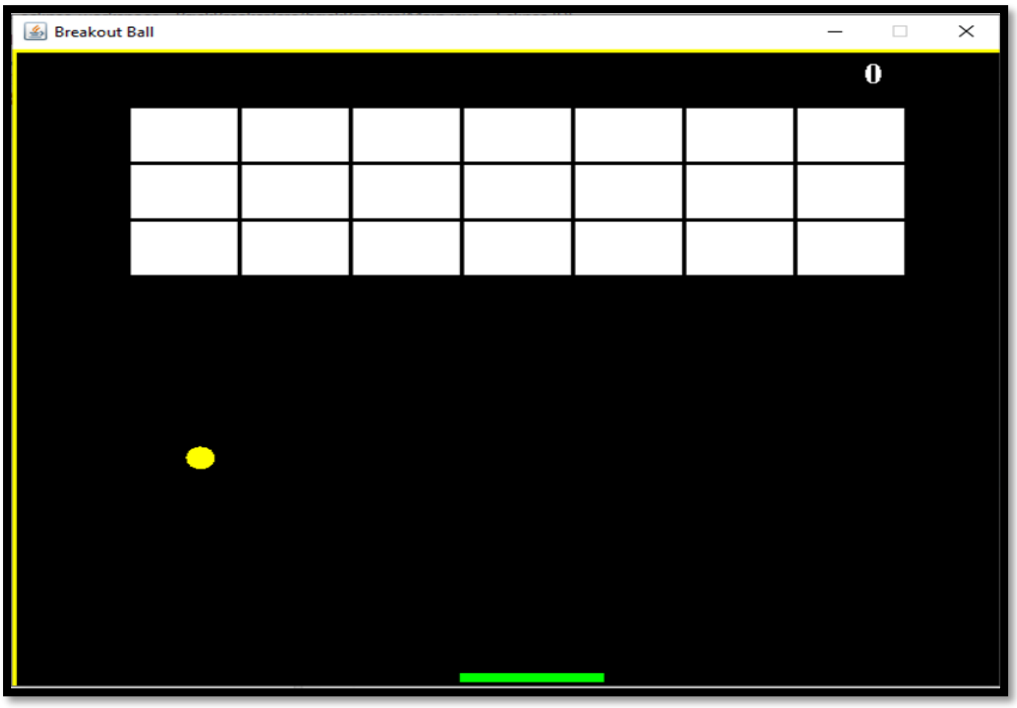
- **MapGenerator CLASS**

```
package brickBreaker;


import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Graphics2D;


public class MapGenerator {


        public int map[][];
```

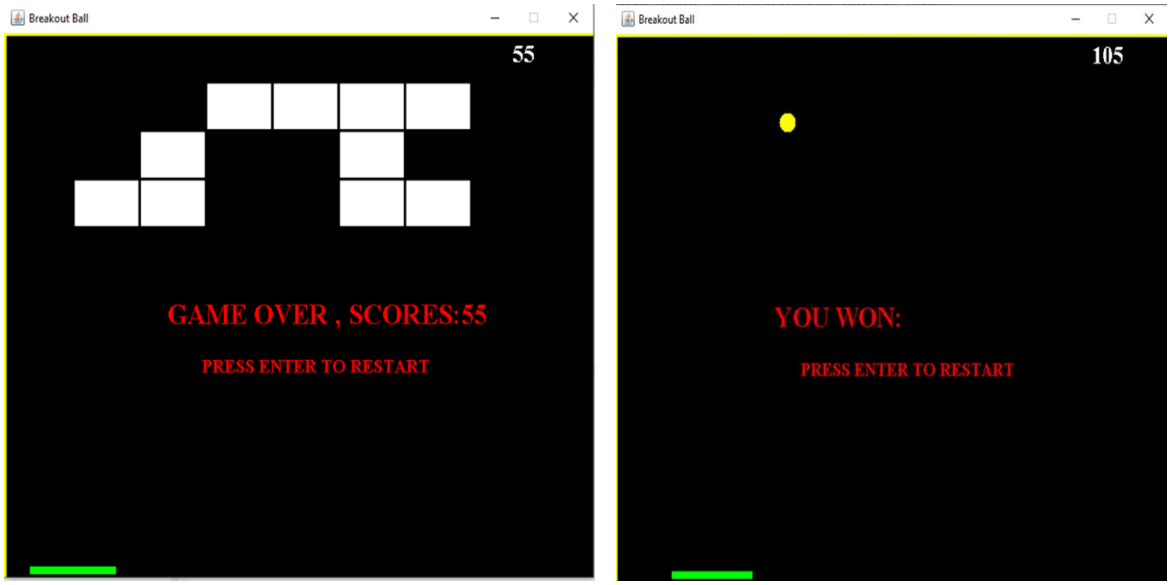
```
	public int brickWidth;
	public int brickHeight;

	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; // 1 will detect that this particular brick have not been intercepted by the ball
			}
		}
		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.RED);
				g.fillRect(j*brickWidth+80,i*brickHeight+50, brickWidth,brickHeight);
				g.setStroke(new BasicStroke(3));
				g.setColor(Color.BLUE);
				g.drawRect(j*brickWidth+80,i*brickHeight+50, brickWidth,brickHeight);

			}
```

```
                }


        }

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

3. RESULT



The first screen that is shown in the picture is of the game terminal from where our game starts. Thus as we press one of the arrow keys then the ball starts moving towards the brick.



The picture shown above is the case when the ball falls off the paddle and the second picture is when we complete the whole game.

4. CONCLUSION

Thus from the above project , we have created a game using Eclipse IDE in java programming language. We have used three main classes i.e the Main , The Gameplay and the Map Generator class and combing them all in a single package to generate a gameplay.

5. REFERENCES

- https://www.tutorialspoint.com/java/index.htm
- https://www.javatpoint.com/
- https://www.w3schools.com/java/
- https://www.youtube.com/watch?v=K9qMm3JbOH0&t=2670s