Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2023), B.Sc. in CSE (eve)

Break The BriX Game

Course Title: Object Oriented Programming lab
Course Code: CSE 202
Section: 221 D17

Students Details

Name	ID
Humayra Afia Hany	221002338

Submission Date: 22-06-2023
Course Teacher's Name: Mr. Montaser Abdul Quade

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

The Break The BriX game is a classic arcade-style game that involves breaking bricks with a ball and a paddle. The game is challenging and addictive, with increasing difficulty levels as the player progresses through the game.

1.2 Motivation

The motivation behind choosing this project is to learn and apply Object-Oriented Programming concepts using Java. Moreover, building a game like Break The BriX provides an opportunity to explore the graphical user interface (GUI) and event-driven programming in Java.

1.3 Problem Definition

1.3.1 Problem Statement

The problem is to design and develop a Break The BriX game using Java that is challenging, engaging, and enjoyable to play.

1.3.2 Complex Engineering Problem

The development of a Break The BriX game involves various engineering challenges, such as designing the game logic, implementing the graphical user interface, handling user input, managing the game state, and optimizing game performance.

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	Moderate knowledge of Java programming language, object-oriented programming concepts, game development concepts, and the Java game development framework is required to develop the game.
P2: Range of conflicting requirements	—-
P3: Depth of analysis required	A basic understanding of game design principles and an analysis of game mechanics and physics will be required.
P4: Familiarity with issues	Basic familiarity with common game development issues such as game logic, game physics, and game UI design is needed.
P5: Extent of applicable codes	The project will primarily involve coding in Java language and using existing libraries and frameworks.
P6: Extent of stakeholder involvement and conflicting requirements	—-
P7: Interdependence	

1.4 Design Goals

The objectives of the project are to design and implement a Break The BriX game using Java and Object-Oriented Programming principles.

- Developing a user-friendly interface that is easy to use and understand.
- The primary design goals for the Break The BriX game project are to create a game that is engaging, challenging, and fun for players of all ages.
- The game will have an intuitive user interface, smooth gameplay, and cross-platform compatibility.
- The game will be optimized for performance and provide a challenging and enjoyable experience for the player.

1.5 Application

The Break The BriX game has wide-ranging applications in the gaming industry, including mobile games, desktop games, and browser games. The game's simple mechanics and addictive gameplay make it a popular choice for casual gamers. Additionally, the project's focus on Java-based game development libraries makes it accessible to a wider audience.

Chapter 2

Implementation of the Project

2.1 Introduction

The Break the Bricks game is a classic arcade game where the player controls a paddle to bounce a ball and break bricks. The objective is to clear all the bricks on the screen by hitting them with the ball. The game includes multiple levels with increasing difficulty and tracks the player's score and lives.

This project report provides an overview of the implementation details of the Break the Bricks game using Java. It covers the project's objectives, implementation approach, and key features

2.2 Project Details

In this section, I will elaborate on all my project details.

2.2.1 Objective:

The objective of this project is to develop a Break the Bricks game using Java that offers an enjoyable gaming experience for players. The game will feature multiple levels, scoring system, and lives for the player. The implementation will utilize Java's Swing library for GUI components and event handling.

2.3 Implementation

2.3.1 Game Components

The Break the Bricks game consists of the following key components:

- Game Board: Represents the playing area where bricks, paddle, and ball are displayed.

- Bricks: Rectangular elements arranged in a pattern, which the player needs to break by hitting them with the ball.
- Paddle: A movable platform controlled by the player to bounce the ball.
- Ball: A moving object that interacts with the bricks and paddle to break the bricks.
- Scoring System: Tracks the player's score based on the number of bricks destroyed.
- Lives: Represents the number of chances the player has before the game ends.

2.3.2 The workflow

The game follows the following flow:

1. Display the game board and initialize the bricks, paddle, and ball.
2. Prompt the player to enter their name and select the desired level.
3. Start the game with the selected level and initialize the score and lives.
4. Render the game components on the screen, including bricks, paddle, and ball.
5. Allow the player to control the paddle using arrow keys to bounce the ball.
6. Update the ball's position, check for collisions with the paddle and bricks.
7. If the ball hits a brick, remove the brick, update the score, and check for win conditions.
8. If the ball hits the paddle, change the ball's direction.
9. If the ball goes below the paddle, reduce the player's life count.
10. If the player runs out of lives, end the game and display the game over message with the final score.
11. If the player clears all the bricks, display the victory message with the final score.
12. Allow the player to restart the game by pressing the Enter key.
13. Provide an option to select a different level by pressing the corresponding number key.
14. Repeat steps 4-13 until the player chooses to exit the game.

2.3.3 Tools and Technologies Used:

- Java programming language
- Swing library for GUI
- Visual Studio Code

2.4 Algorithms

- Step 1: Import the necessary packages and classes required for the game.
- Step 2: Create a 'Map' class that represents the game board and initializes the bricks.
- Step 3: Implement the 'draw' method in the 'Map' class to render the bricks on the game board.
- Step 4: Create a 'GamePlay' class that extends 'JPanel' and implements 'KeyListener' and 'ActionListener'.
- Step 5: Initialize variables for game state, score, lives, level, player position, and ball properties.
- Step 6: Create a timer and set it to trigger the 'actionPerformed' method at a fixed interval.
- Step 7: Override the 'paintComponent' method to draw the game elements on the panel.
- Step 8: Handle collisions between the ball and bricks, update the score, and check for win conditions.
- Step 9: Implement key events to move the player paddle and restart the game.
- Step 10: Create a 'showGameOverMessage' method to display the game over a message with the final score.
- Step 11: Implement the 'actionPerformed' method to update the game state, handle ball movement and collisions.
- Step 12: Override the 'keyPressed' method to handle user input for paddle movement and game restart.
- Step 13: Implement the 'restartGame' method to reset the game state based on the selected level.
- Step 14: Create a 'Main' class to prompt the user for their name and level selection.
- Step 15: Validate the inputs and create an instance of the 'GamePlay' class.
- Step 16: Create and show the game frame.

2.4.1 Implementation details (with screenshots and programming codes)

```java
package myapp;
/*
 * @author Humayra
 */
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class Map {
    public int mp[][];
    public int brickWidth;
    public int brickHeight;


    public Map(int row, int col) {
        mp = new int[row][col];
        for (int r = 0; r < row; r++) {
            for (int c = 0; c < col; c++) {
                mp[r][c] = 1;
            }
        }

        brickWidth = 540 / col;
        brickHeight = 200 / row;
    }
```

Figure 2.1: import classes

```java
class Main {
    Run | Debug
    public static void main(String[] args) {
        String name = JOptionPane.showInputDialog(parentComponent:null, message:"Enter your name

        if (name != null && !name.isEmpty()) {
            int selectedLevel = JOptionPane.showOptionDialog(parentComponent:null, message:"Sele
                    JOptionPane.DEFAULT_OPTION, JOptionPane.PLAIN_MESSAGE, icon:null, new String

            if (selectedLevel != JOptionPane.CLOSED_OPTION) {
                SwingUtilities.invokeLater(() -> createAndShowGameFrame(name, selectedLevel + 1)
            } else {
                JOptionPane.showMessageDialog(parentComponent:null, message:"Invalid level selec
                        title:"Error", JOptionPane.ERROR_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(parentComponent:null, message:"Invalid player name. Ga
                    title:"Error", JOptionPane.ERROR_MESSAGE);
        }
    }
```

Figure 2.2: Main class

```java
public GamePlay(String name, int selectedLevel) {
    playerName = name;
    level = selectedLevel;

    switch (level) { // Create map based on selected level
        case 1: // Easy level
            map = new Map(row:4, col:7);
            totalBricks = 28;
            break;
        case 2: // Medium level
            map = new Map(row:6, col:7);
            totalBricks = 42;
            break;
        case 3: // Hard level
            map = new Map(row:8, col:8);
            totalBricks = 64;
            break;
    }
```

Figure 2.3: Multiple levels

```java
// This draws the bricks
public void draw(Graphics2D g) {

    int r= mp.length;
    for (int i = 0; i < mp.length; i++) {
        for (int j = 0; j < mp[0].length; j++) {
            if(i<r/2){
            if (mp[i][j] > 0) {
                g.setColor(new Color(rgb:0X7f00ff)); // Brick color
                g.fillRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight

                g.setStroke(new BasicStroke(width:4));
                g.setColor(Color.BLACK);
                g.drawRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight
            }
        }
        else{
            if (mp[i][j] > 0) {
                g.setColor(new Color(rgb:0Xff33ff)); // Brick color
                g.fillRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight

                g.setStroke(new BasicStroke(width:4));
                g.setColor(Color.BLACK);
                g.drawRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight
            }
        }
        }
```

Figure 2.4: Drawing the bricks

Chapter 3

Performance Evaluation

3.1 Result

First player have to enter their name.

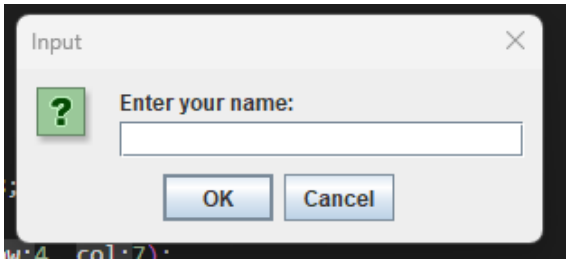

Figure 3.1: Enter name

Then select level

Figure 3.2: Select level

3.1.1 Results Overall Discussion

The Break the Bricks game implemented in this project provides an enjoyable gaming experience for users. The game follows the classic gameplay mechanics of the original

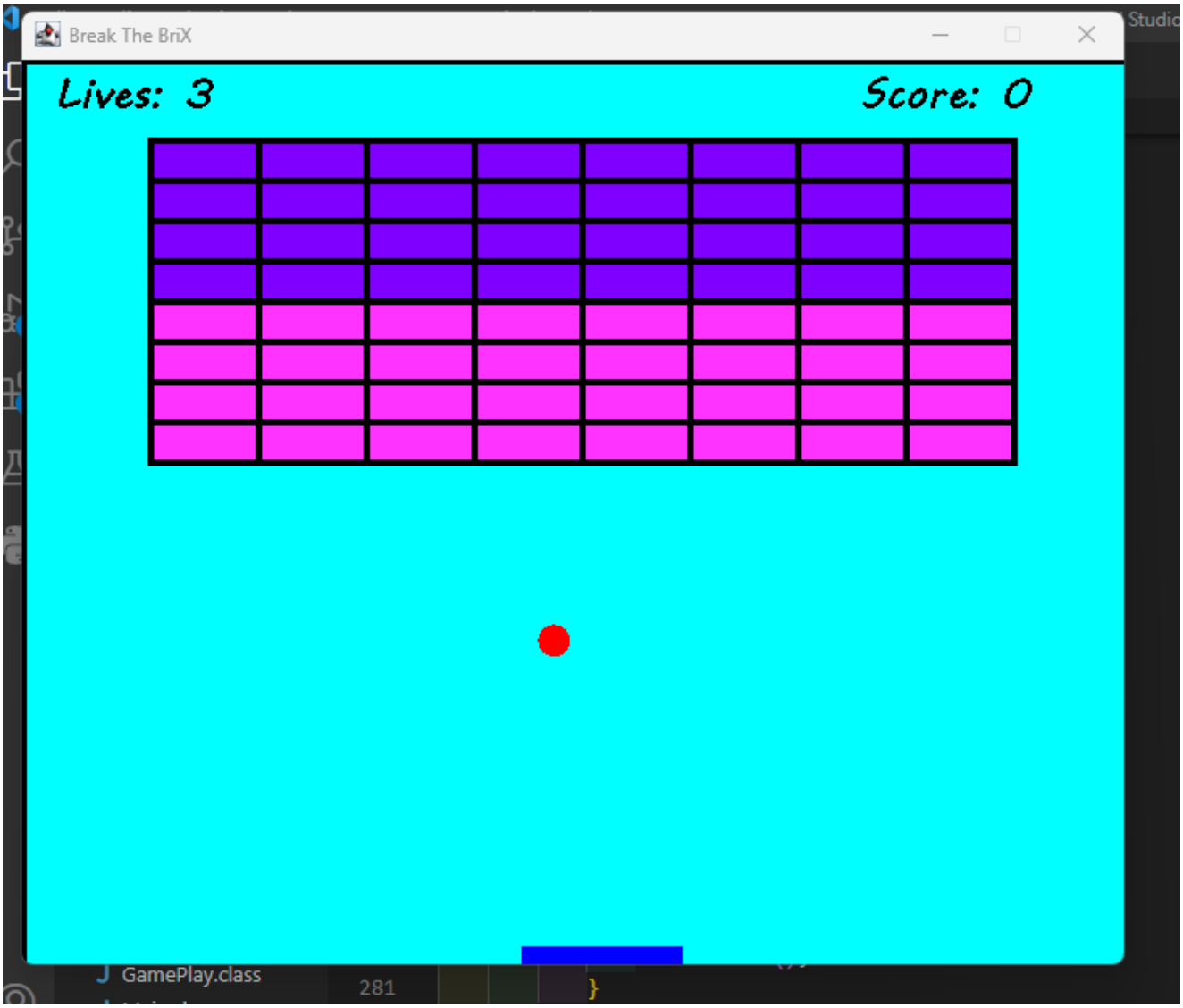

Figure 3.3: Game start

Breakout game while introducing multiple levels of difficulty. The project successfully utilizes Java's Swing library for creating a graphical user interface and handling user input. The code structure is modular and organized, making it easy to understand and maintain.

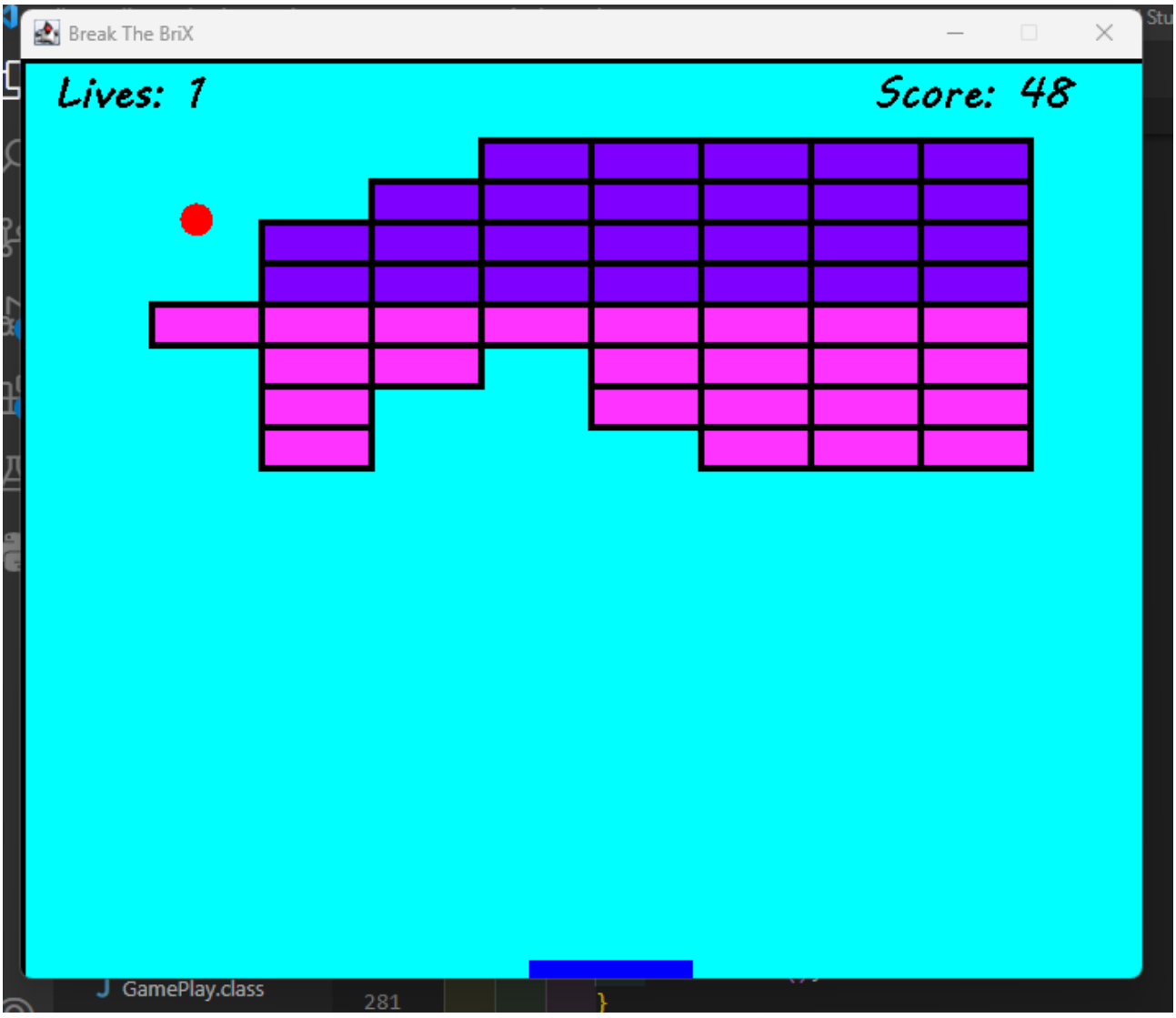

Figure 3.4: Playing

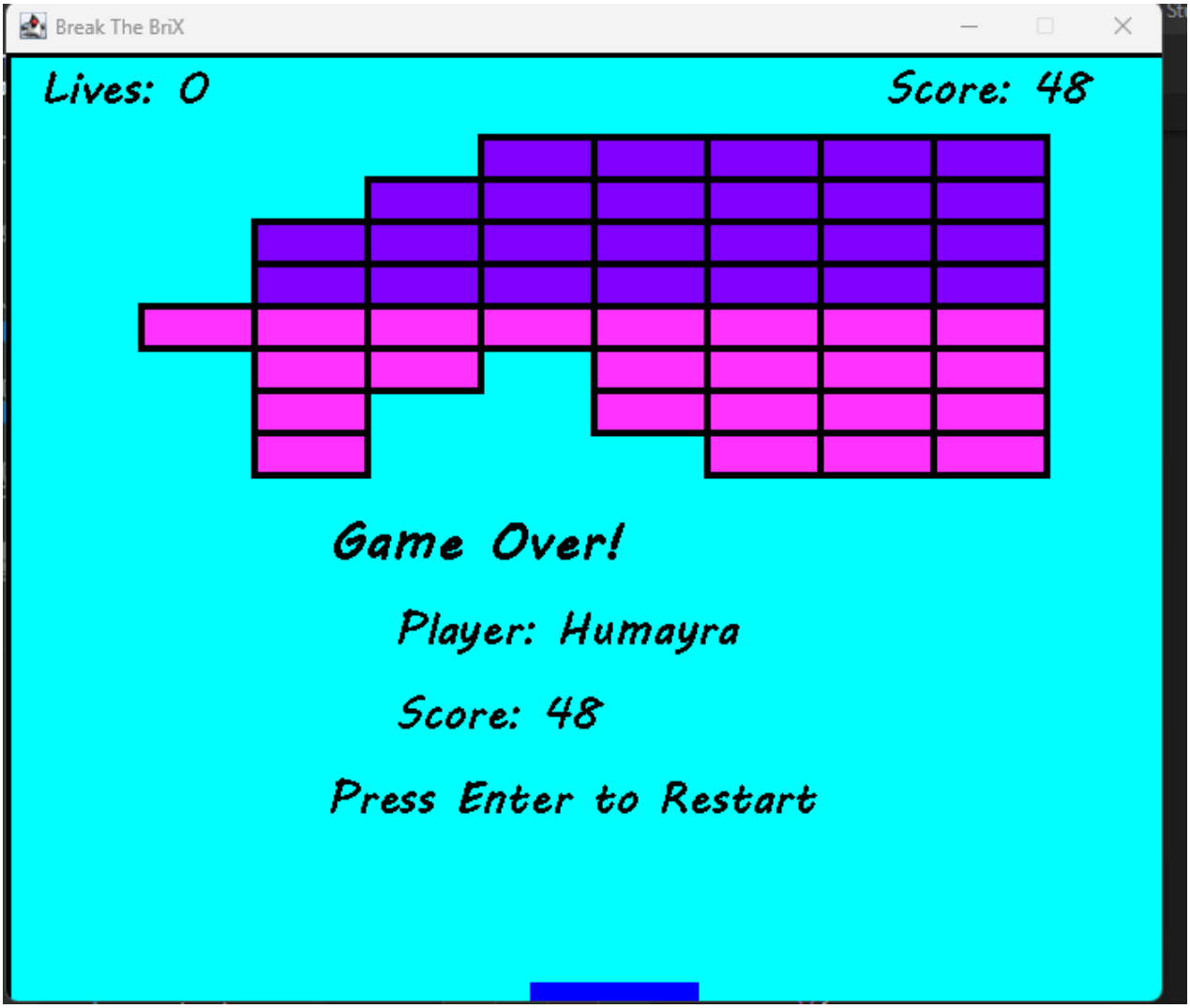

Figure 3.5: Lives 0

Figure 3.6: level complete

Chapter 4

Conclusion

The Break the Bricks game project implemented using Java provides an engaging gaming experience for players. It offers multiple levels of increasing difficulty, tracks the player's score and lives, and allows for game restarts and level selections. The use of Java's Swing library enables the creation of a graphical user interface for the game. This project helps improve programming skills, understanding of game development concepts, and event-driven programming using Java.

Overall, the Break the Bricks game project serves as an enjoyable way to apply Java programming knowledge and explore game development concepts.

4.1 Discussion

The Break the Bricks game implemented in this project provides an enjoyable gaming experience for users. The game follows the classic gameplay mechanics of the original Breakout game while introducing multiple levels of difficulty. The project successfully utilizes Java's Swing library for creating a graphical user interface and handling user input. The code structure is modular and organized, making it easy to understand and maintain.

4.2 Limitations

- The game currently has a fixed layout and predefined levels. It does not provide the option for custom levels or dynamic level generation.
- The game lacks sound effects and background music, which could enhance the gaming experience.
- The game does not have a high score tracking or leaderboard feature.

4.3 Scope of Future Work

- Add sound effects and background music to enhance the gaming experience.

- Implement a high score tracking system and leaderboard.
- Introduce power-ups or special bricks to add more variety and challenges to the game.
- Implement level editor functionality to allow users to create custom levels.
- Add additional visual effects and animations to make the game more visually appealing.
- Enhance the game's responsiveness and smoothness by optimizing the code and utilizing multi-threading techniques.