BRICK BREAKER GAME

A PROJECT REPORT

Submitted by

NAVEENA R (2303811710422104)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AnAutonomousInstitution,affiliatedtoAnnaUniversityChennaiandApprovedbyAICTE,New Delhi)

SAMAYAPURAM–621112

NOVEMBER- 2024

K.RAMAKRISHNANCOLLEGEOFTECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM–621112

BONAFIDECERTIFICATE

Certified that this project report on **" BRICK BREAKER GAME"** is the bonafide work of **NAVEENA R (2303811710422104)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.,),
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 03.12.2024

INTERNAL EXAMINER EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **"BRICK BREAKER GAME"** is the result of original work done by us and best of our knowledge, similar workhas not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201-JAVA PROGRAMMING.**

.

Signature

NAVEENA R

Place: Samayapuram

Date:03.12.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequateduration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.,),** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patiencewhich motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The Brick Breaker game, developed using Java, is a dynamic, arcade-style game in which the player controls a paddle to bounce a ball and break all the bricks arranged on the screen. The objective is to clear all the bricks by striking them with the ball while preventing the ball from falling off the screen. The game features multiple levels, each presenting increasingly complex brick patterns, faster ball speeds, and additional obstacles, creating a progressively challenging experience.

The core gameplay mechanics are driven by real-time physics and collision detection, which are implemented using Java's built-in libraries and custom algorithms. The ball, paddle, and bricks are all objects within the game, and Java’s object-oriented programming (OOP) features are utilized to manage interactions, such as ball movement, paddle control, and brick destruction. Additionally, power-ups such as enlarged paddles, multi-ball modes, and speed boosters add variety and strategic elements to the game.

The user interface is simple yet intuitive, with arrow keys or mouse controls for paddle movement and a score tracking system to measure player progress. The game’s flow is controlled by a game loop, ensuring smooth animations and real-time interactions. Java’s event-handling mechanisms enable responsive input, while the game's physics engine simulates realistic ball movements and collisions with walls, the paddle, and bricks.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Brick Breaker Game is a classic arcade game where the player controls a paddle to bounce a ball and break bricks arranged on the screen. The game challenges the player with progressively harder levels, power-ups, and obstacles. The ball's speed increases as the game advances, requiring the player to use strategy and quick reflexes.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1Objective

The objective of the Brick breaker game is for the player to control a paddle and bounce a ball to break all the bricks on the screen. The player must avoid letting the ball fall off the bottom of the screen, which would result in a loss of a life. Each brick requires one or more hits to break, and the player earns points for every brick destroyed. As the player progresses through the levels, the difficulty increases, with more complex brick layouts, faster ball speeds, or additional obstacles. The ultimate goal is to clear all the bricks while keeping the ball in play and achieving the highest score possible.

1.2Overview

Brick Breaker is a classic arcade game where players control a paddle to bounce a ball and break bricks arranged at the top of the screen. The goal is to clear all the bricks by using the ball, which bounces off the paddle and destroys bricks on contact. As players progress through levels, the game increases in difficulty, featuring faster ball speeds, more complex brick arrangements, and various types of bricks, some of which require multiple hits or are indestructible. Power-ups that modify gameplay such as larger paddles, multiple balls, or enhanced ball speed—add an extra layer of challenge and excitement. With its simple mechanics and progressively harder levels, Brick Breaker offers endless replayability, making it a timeless favorite across different platforms. It’s easy to learn yet challenging enough to keep players engaged, improving hand-eye coordination and reflexes while offering quick and satisfying gameplay sessions.

1.3 Java Programming Concepts

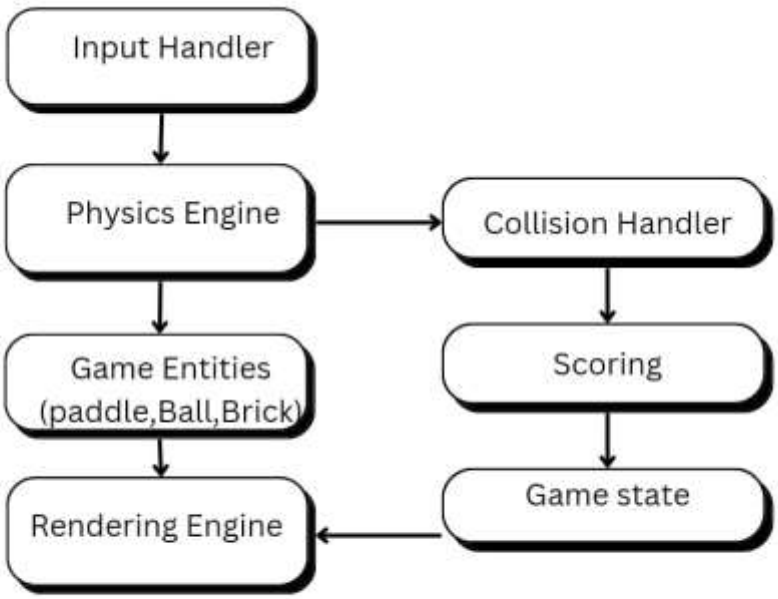
- **Classes** and **Objects** are the building blocks of any Java program. In a Brick Breaker game, you will create classes that define the various entities in the game (e.g., Ball, Paddle, Brick, and Game.
- **Inheritance** : Used to create different types of bricks that share common properties (like position and size), but each brick can have its own unique behavior (e.g., indestructible or power-up).
- **Control flow**(**loops and conditionals**):Loops are used to continuously check for game events such as moving the ball, checking for collisions, and updating the game state. Conditionals (if, else) are used to make decisions, such as when the ball hits the paddle, bricks, or walls.
- **Event handling:** In Java, event handling like ActionListener is used for handle time and Keylistener is used for handle user input
- **Swing and AWT** : Component like JFrame ,JLabel, JButton and Jpanel are used to build the user interface.
- **Array list** :ArrayLists and other collections are useful for managing dynamic collections of objects, such as the collection of bricks in the game.

CHAPTER 2
PROJECT METHODOLOGY

2.1Proposed Work

The proposed work for the Brick Breaker game includes designing core game mechanics such as paddle movement, ball interaction with bricks, and defining the goal of clearing all bricks. Multiple levels will be created with increasing difficulty, and a scoring system will track the player's progress. Key game components include the ball, paddle, and different types of bricks (normal, indestructible, power-up). The user interface will feature a start screen, HUD (score, lives, level), and game-over screen. Input handling will allow paddle control via keyboard or mouse, and game events like ball collisions will be managed throughout the game play.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 Game Initialization Module

Sets up the game window, initializes objects (ball, paddle, bricks), and defines initial conditions like score and lives. It also sets up input handling for user controls.

Game Initialization Module is a Main Menu Screen. It allows players to select options like Start Game, View Instructions, Adjust Settings, or Exit, providing a structured and organized entry point to the game. This enhances user experience and adds a professional touch.

3.2 Ball Module

Ball Module makes the ball move and interact with the paddle and bricks. It ensures the ball bounces off walls and the paddle, and it detects brick collisions to update the game state.

The Ball Module could be ball speed variation, where the ball gradually speeds up over time or after breaking certain bricks. This increases the challenge and keeps the gameplay dynamic.

3.3 Paddle Module

Paddle Module gives the player control over the paddle and interacts with the ball to keep it in play. It also resizes the paddle during power-ups and ensures it remains on-screen. An extra point for the Paddle Module could be the addition of paddle power-ups. For example, when certain bricks are destroyed, they could drop power-ups that temporarily increase the paddle size, allow the paddle to shoot lasers, or make it move faster. This adds variety and excitement to the gameplay by giving the player new abilities.

3.4 Brick Module

Brick Module handles the creation and destruction of bricks, which are the obstacles the player must break to win. It also manages power-ups and special brick effects.

3.5 Game Logic and Control Module

The Game Logic and Control Module is responsible for managing the overall flow of the game, including game progression, scoring, level transitions, and win/loss conditions. It tracks the player's score, which increases as bricks are broken, and manages the player's lives, decreasing each time the ball is missed. When a level is completed, it transitions to the next, often increasing the difficulty by speeding up the ball or changing the brick layout. The module ensures the game properly ends when the player runs out of lives or completes all levels, displaying appropriate messages for win or loss conditions. This module is crucial for maintaining the structure of the game, ensuring smooth gameplay and progression.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

In conclusion, Brick Breaker is a simple yet highly addictive arcade game that combines easy-to-learn mechanics with progressively challenging gameplay. By using a paddle to bounce a ball and break bricks, players experience satisfying, skill-based action that keeps them engaged through increasing difficulty and power-ups. Its timeless appeal lies in its balance of simplicity, challenge, and quick play sessions, making it a favorite among casual gamers across various platforms. The game fosters hand-eye coordination and reflexes, offering not only fun but also light mental exercise. The endless replay value, driven by random brick patterns and power-up combinations, ensures that Brick Breaker remains enjoyable with each playthrough.

4.2 FUTURE SCOPE

The future scope of brick breaker games holds significant potential for innovation as technology and gaming trends evolve. We can expect advancements in gameplay mechanics, with more sophisticated physics engines that allow for realistic ball behavior, like spin, velocity, and trajectory. This could lead to dynamic and unpredictable interactions between the ball and bricks, making each level feel unique. In addition, multiplayer modes could emerge, enabling players to compete or cooperate in breaking bricks together. Game mechanics might also evolve with procedural level generation, offering endless variety and challenges, while power-ups and abilities would become more diverse, allowing players to tailor gameplay to their preferences. The rise of cloud gaming and cross-platform play could bring brick breaker games to a broader audience, making them more accessible and interconnected across devices. The future of brick breaker games will likely involve a blend of cutting-edge technology, deeper interactivity, and more engaging gameplay, ensuring the genre remains fresh and appealing to players of all ages.

APPENDIX A
(SOURCE CODE)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

 class BrickBreaker {
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        Gameplay gameplay = new Gameplay();

        frame.setBounds(10, 10, 700, 600);
        frame.setTitle("Brick Breaker");
        frame.setResizable(false);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(gameplay);
        frame.setVisible(true);
    }
}

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) {
    // 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);

    // scores
    g.setColor(Color.white);
```

```
g.setFont(new Font("serif", Font.BOLD, 25));
g.drawString("Score: " + score, 550, 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);

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!", 260, 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, Scores: " + 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 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) {}

@Override
public void keyPressed(KeyEvent e) {
if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
    if (playerX >= 600) {
```

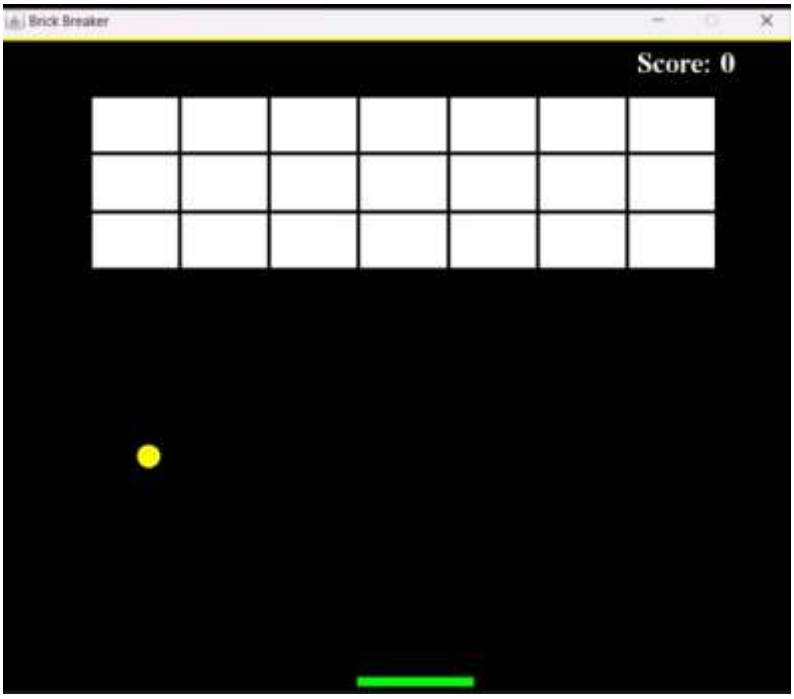
```
            playerX = 600;
        } else {
            moveRight();
        }
    }

    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 = 21;
            map = new MapGenerator(3, 7);

            repaint();
        }
    }
}

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

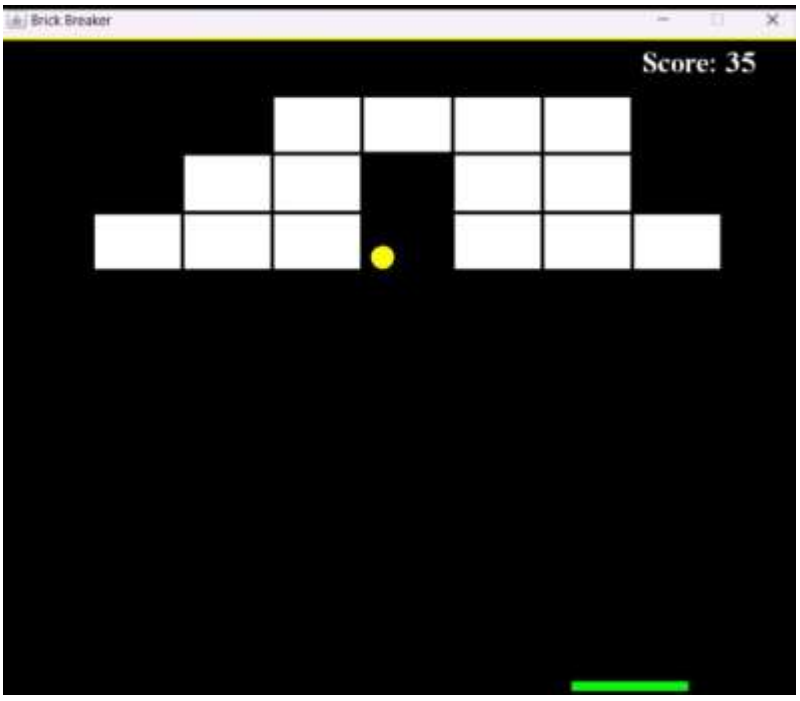
```
    }

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

    @Override
    public void keyReleased(KeyEvent e) {}
}

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);
```

```
                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;
}
}
```

APPENDIX B

(SCREENSHOTS)






Brick Breaker
Score: 50
Game Over, Scores: 50
Press Enter to Restart

REFERENCES

Book

- **Core Java Volume I-Fundamentals –Cay S.Horstmann.**
- **Java :The complete References-Herbert Schildt**

websites

- **GeeksforGeeks**
- **Java tutorial**