DECLARATION

I declare that the project report on **“BRICK BREAKER GAME”** is the result of original work done by me and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of BACHELOR OF ENGINEERING. This project reportis submitted on the partial fulfillment of the requirement of the award of the cours **CGB1201-JAVA PROGRAMMING**

Signature

AGARA MUTHALVAN P S

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in debtedness to our institution, “**K.RAMAKRISHNAN COLLEGE OF ENGINEERING (Autonomous)**”,for providing me with the opportunity to do this project. I extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided 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 our project and offering an adequate duration to complete it.I would like to thank **Dr. D. SRINIVASAN, M.E., Ph.D., FIE., MIIW.,MISTE., MISAE., C. Engg.,** Principal, who gave the opportunity to frame the project to full satisfaction.

Dr. B. KIRAN BALA, B.Tech., M.E., M.B.A., Ph.D., M.I.S.T.E., U.A.C.E.E., IAENG, Head of the Department of Artificial Intelligence and Machine Learning, for providing his encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide**Mrs. P.GEETHA, M.E.,** Department of Artificial Intelligence and Data Science, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the 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 .

INSTITUTE VISION AND MISSION

<u>VISION OF THE INSTITUTE:</u>

To achieve a prominent position among the top technical institutions.

<u>MISSION OF THE INSTIITUTE:</u>

M1: To be show standard technical education excellence through state of the art infrastructure, competent faculty and high ethical standards.

M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.

M3: To provide education for developing high-quality professionals to transform the society.

DEPARTMENT VISION AND MISSION

<u>DEPARTMENT OF CSE(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)</u>

<u>Vision of the Department</u>

To become a renowned hub for Artificial Intelligence and Machine Learning

Technologies to produce highly talented globally recognizable technocrats to meet

Industrial needs and societal expectations.

<u>Mission of the Department</u>

M1: To impart advanced education in Artificial Intelligence and Machine Learning,

Built upon a foundation in Computer Science and Engineering.

M2: To foster Experiential learning equips students with engineering skills to

Tackle real-world problems.

M3: To promote collaborative innovation in Artificial Intelligence, machine

Learning, and related research and development with industries.

M4: To provide an enjoyable environment for pursuing excellence while upholding

Strong personal and professional values and ethics.

Programme Educational Objectives (PEOs):

Graduates will be able to:

PEO1: Excel in technical abilities to build intelligent systems in the fields of

Artificial Intelligence and Machine Learning in order to find new opportunities.

PEO2: Embrace new technology to solve real-world problems, whether alone or

As a team, while prioritizing ethics and societal benefits.

PEO3: Accept lifelong learning to expand future opportunities in research and

Product development.

Programme Specific Outcomes (PSOs):

PSO1: Ability to create and use Artificial Intelligence and Machine Learning

Algorithms, including supervised and unsupervised learning, reinforcement

Learning, and deep learning models.

PSO2: Ability to collect, pre-process, and analyze large datasets, including data

Cleaning, feature engineering, and data visualization..

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,reviewresearchliterature,andan

alyze 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 effectivelyon 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 is a classic arcade-style project designed to enhance problem-solving, logical thinking, and programming skills. This interactive game involves a player-controlled paddle, a bouncing ball, and a grid of bricks to break, offering engaging gameplay mechanics. Players score points by successfully hitting and breaking bricks using the ball, with the challenge escalating as the game progresses. The game logic incorporates collision detection to handle interactions between the ball, paddle, and bricks. Multiple levels with increasing difficulty are implemented, encouraging replayability and testing the player's reflexes and strategies.This project is ideal for showcasing skills in programming, graphical interface design, and game mechanics development. It also serves as an enjoyable way to explore core concepts of software development while creating an engaging user experience.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
The Brick Breaker game is a classic arcade-style project designed to enhance problem-solving, logical thinking, and programming skills. This interactive game involves a player-controlled paddle, a bouncing ball, and a grid of bricks to break, offering engaging gameplay mechanics. Players score points by successfully hitting and breaking bricks using the ball, with the challenge escalating as the game progresses. The game logic incorporates collision detection to handle interactions between the ball, paddle, and bricks. Multiple levels with increasing difficulty are implemented, encouraging replayability and testing the player's reflexes and strategies.This project is ideal for showcasing skills in programming, graphical interface design, and game mechanics development. It also serves as an enjoyable way to explore core concepts of software development while creating an engaging user experience.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3**	**PSO1 -3** **PSO2 -3**

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

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective:

The primary objective of the Brick Breaker game project is to design and develop an interactive arcade game that challenges players' reflexes and strategic thinking. This project seeks to provide a platform for learning and implementing core Java programming concepts in a practical, hands-on manner. It also aims to enhance the understanding of event-driven programming through real-time user interactions and dynamic game logic. Another goal is to introduce players to progressively challenging levels, maintaining their interest and encouraging continuous improvement in their skills. Overall, the objective is not only to create an engaging game but also to showcase a comprehensive understanding of Java programming and its application in game development.

1.2 Overview:

The Brick Breaker game is a classic arcade game where players control a paddle to bounce a ball and break bricks on the screen. The game begins with a predefined number of bricks, and the player’s goal is to clear all the bricks without losing the ball. Key features include intuitive controls, dynamic gameplay, and progressively challenging levels.To enhance the gaming experience, features like power-ups, special bricks with unique behaviors, and sound effects are integrated. This project serves as a foundational step for aspiring game developers, showcasing the practical application of Java programming concepts and the process of creating an interactive application.

1.3 Java Programming Concepts:

The Brick Breaker game leverages several core and advanced Java programming concepts to deliver a functional, efficient, and user-friendly experience. These concepts ensure seamless gameplay while providing a comprehensive understanding of Java programming. Below is an elaboration on the concepts applied:

Object-Oriented Programming (OOP):

- **Classes and Objects:** The game is structured around key objects like the paddle, ball, bricks, and game logic, each encapsulated in their respective classes. This modular design promotes reusability and better organization.
- **Inheritance:** The inheritance feature enables the creation of specialized brick types, such as power-up bricks, by reusing the base brick class logic, reducing code duplication.
- **Polymorphism:** Polymorphism ensures that different types of bricks (e.g., regular bricks and special bricks) exhibit unique behaviors when hit by the ball, such as adding extra points or triggering special effects.

Event Handling:

The game incorporates event-driven programming by capturing and responding to user inputs, such as moving the paddle left or right through keyboard events. Java's KeyListener interface is used to detect these actions.

Graphics and GUI:

Java AWT (Abstract Window Toolkit) and Swing: These libraries form the foundation of the game's graphical interface. They enable rendering the game components, such as the paddle, ball, and bricks, as well as animating their movements dynamically during gameplay.

Collision Detection:

Logical conditions are implemented to detect and manage interactions between the ball and other objects, such as the paddle, bricks, and game boundaries. This ensures realistic and consistent gameplay behavior, such as reversing the ball's direction upon collision.

Threads:

Threads are utilized to handle the game’s logic and animations in parallel, ensuring smooth gameplay. For instance, a Timer object is used to periodically update the ball’s position and refresh the screen without freezing other operations.

Collections Framework:

Data structures such as arrays or ArrayLists are used to dynamically manage and track multiple bricks or power-ups during gameplay. This allows efficient updates when bricks are hit or removed.

Exception Handling:

Proper exception handling mechanisms are employed to manage runtime errors, such as invalid inputs or unexpected behavior during gameplay, ensuring the game continues to run smoothly without crashing.

Loops and Conditional Statements:

Core game logic relies heavily on loops and conditionals to control actions like ball movement, collision detection, score updates, and level progression. These constructs form the backbone of the game’s flow.

Encapsulation:

Encapsulation is used to protect the game’s internal state, such as the position of the paddle, ball, and score. Access to these variables is restricted through getter and setter methods, ensuring data integrity.

File Handling (Optional):

File handling can be used to store high scores, player names, or other game-related data, enabling persistence across gaming sessions. This feature enhances user engagement and adds a professional touch to the game.

Significance:

This combination of concepts not only ensures a fully functional and interactive game but also demonstrates the power and versatility of Java programming. By integrating these features, the project provides a hands-on experience with real-world Java applications, making it an excellent learning opportunity for aspiring developers.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work:

Proposed Work for the Brick Breaker Game

The proposed development of the Brick Breaker game adopts a structured and systematic approach, ensuring the creation of a fully functional and engaging arcade-style game. The project progresses through clearly defined phases, each focused on implementing critical aspects to ensure the game is efficient, user-friendly, and scalable. Below is an elaboration of the proposed work:

1. Game Design and Planning

- **Define Game Layout:** A clear definition of the game layout involves deciding the arrangement of elements such as the paddle, ball, bricks, and boundaries. The layout should balance aesthetic appeal with functional gameplay.
- **Rules and Objectives:** Rules govern gameplay, such as losing a life when the ball touches the bottom boundary or scoring points for breaking bricks. Objectives include clearing all bricks or achieving a high score.
- **Scoring System:** Establish a scoring mechanism where points are awarded based on brick durability or special actions like activating power-ups. This incentivizes players to perform well.
- **Modular Architecture:** Plan a modular structure to ensure scalability and ease of maintenance. Each component (e.g., paddle, bricks, collision logic) is developed as an independent module, allowing for easier updates or modifications.

2. Game Components

- **Paddle:**

Develop a responsive paddle controlled by the user through keyboard inputs.Implement smooth and precise movements to enhance the player’s control and gameplay experience.

- **Ball Physics:**

Design the ball’s movement using realistic physics principles, including trajectory changes based on paddle hits and wall rebounds.Introduce velocity adjustments for dynamic gameplay.

- **Bricks:**

Create bricks with varying properties, such as different colors indicating durability levels.Add power-up bricks that grant special bonuses, like additional lives or paddle size increases.

3. Collision Detection

- **Interactions:**

Implement algorithms to handle interactions between the ball and game components, such as bricks and the paddle.

- **Behavior Handling:**

Define specific behaviors for various collision scenarios, such as reversing the ball’s direction when it hits the paddle or destroying a brick on impact.

4. Level Design

- **Unique Layouts:**

Design multiple levels with distinct brick arrangements to maintain player engagement and increase challenge variety.

- **Level Progression:**

Incorporate progressive difficulty by increasing ball speed, reducing paddle size, or introducing complex brick configurations as players advance through levels.

5. GUI Development

- **Menus and Displays:**

Develop intuitive menus for starting the game, pausing, or accessing options.

- **Score Display:**

Implement a dynamic score display and include a life counter for visual feedback.

- **Java Swing or AWT:**

Use these frameworks to create visually appealing and functional graphical user interfaces (GUIs).

6. Power-Ups and Features

- **Bonuses:**

Introduce power-ups like paddle extensions, ball speed reduction, or extra lives to enhance gameplay.

- **Special Bricks:**

Develop bricks with unique effects, such as explosions that destroy surrounding bricks or bricks that trigger multi-ball scenarios.

7. Sound and Animation

- **Sound Effects:**

Add sound effects to provide audio feedback for actions like ball collisions or scoring points.

- **Simple Animations:**

Incorporate animations, such as bricks disappearing or the paddle moving smoothly, to enhance visual appeal.

8. Testing and Debugging

- **Comprehensive Testing:**

Test the game on various systems to identify and fix bugs or performance issues.

- **Optimization:**

Ensure smooth gameplay by optimizing performance for responsiveness and stability.

9. Final Deployment

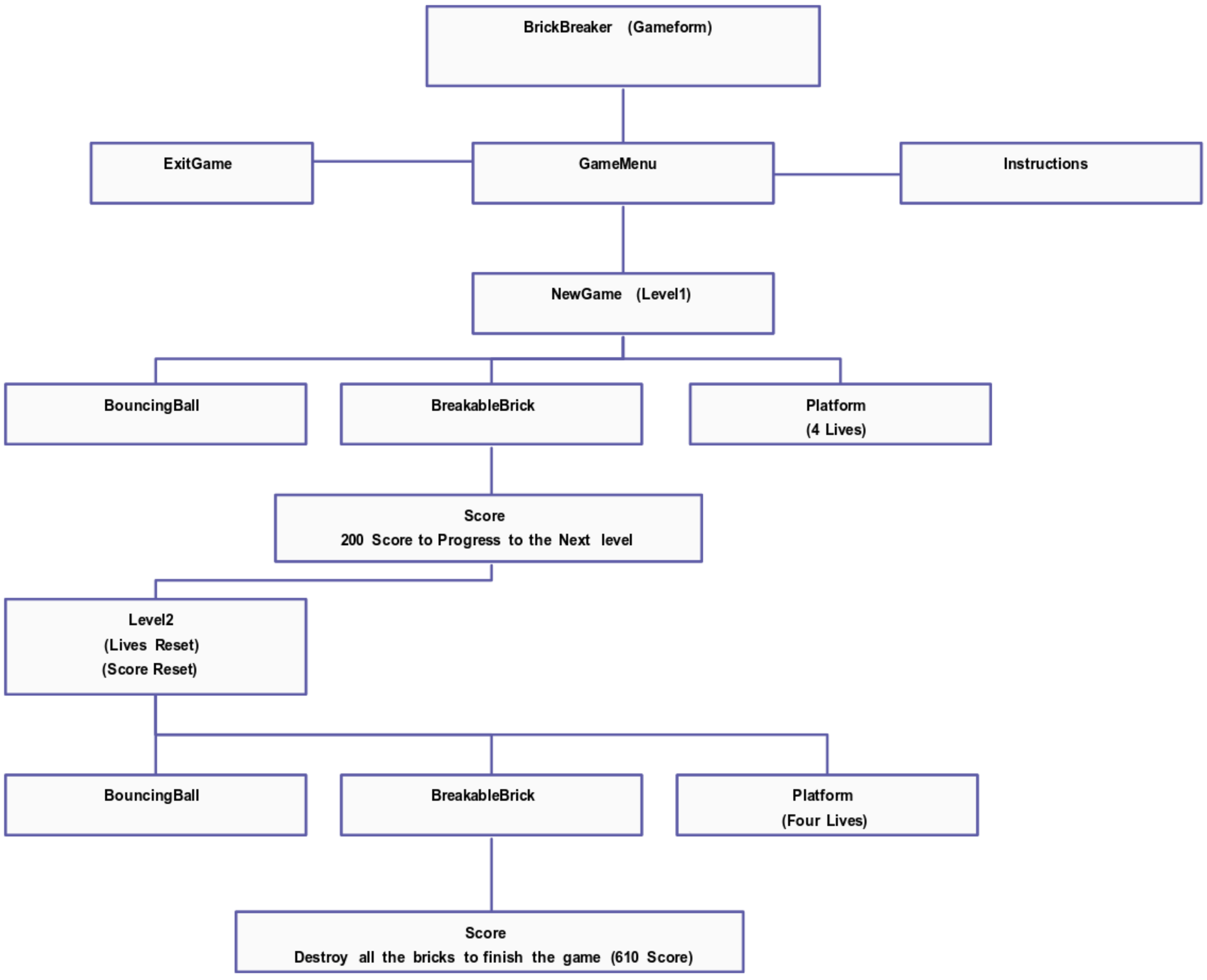
- **Packaging:**

Package the game into a standalone application, complete with clear installation instructions for users.

- **User-Friendly Experience:**

Focus on delivering a polished product with minimal technical barriers for the end user.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Main Module

The main module serves as the entry point for the application. It initializes the game by creating an instance of the **BrickBreaker** JFrame, which in turn initializes the **GamePanel**. This module sets up the main window's properties, such as its size, title, visibility, and behavior when closed.

Key Responsibilities:

- Launch the game.
- Create and configure the JFrame container for the game.

3.2 Game Logic Module

This module handles the game's core mechanics and interactions, such as paddle movement, ball trajectory, collision detection, and score tracking. It ensures the game progresses smoothly by reacting to user inputs and updating the ball's position, detecting collisions with bricks, and managing game over scenarios.

Key Responsibilities:

- Detect ball collisions with the paddle, walls, and bricks.
- Update the ball's position and direction based on game logic.
- Track the score and the number of remaining bricks.
- Handle game-over conditions (win or lose).

3.3 Graphics Rendering Module

This module focuses on rendering the visual components of the game, including the background, paddle, ball, bricks, and borders. It also displays the current score and handles the "Game Over" or "You Won" messages when the game ends.

Key Responsibilities:

- Draw the paddle, ball, and bricks on the screen.
- Render the game background, borders, and score.
- Display appropriate end-of-game messages.

3.4 Input Handling Module

This module manages user input through keyboard events. It listens for specific keys pressed (e.g., left, right, and space) and triggers corresponding actions, such as moving the paddle or starting the game.

Key Responsibilities:

- Respond to user input for paddle movement.
- Enable starting or restarting the game using the space key.
- Ensure smooth control over the paddle's position.

3.5 Brick Management Module

This module is dedicated to managing the bricks in the game. It defines the brick layout, dimensions, and visibility. It also contains the logic for updating bricks when they are hit by the ball.

Key Responsibilities:

- Define the arrangement and dimensions of bricks.
- Handle brick visibility when hit by the ball.
- Provide methods for modifying the brick map.

CHAPTER 4

CONCLUSION & FUTUER SCOPE

4.1 CONCLUSION

The Brick Breaker game project successfully demonstrates the application of fundamental programming concepts in developing an interactive and engaging game. Through this project, key aspects such as collision detection, event handling, and graphical interface design were explored and implemented effectively. The integration of power-ups and multiple levels added complexity and replayability, enhancing the overall gaming experience.This project highlights the importance of modular code structure, which ensures maintainability and scalability for future enhancements. By incorporating intuitive controls and dynamic feedback, the game delivers a seamless user experience. The focus on both functionality and aesthetics emphasizes the balance between technical development and user-centric design.Additionally, this project provided valuable learning opportunities in problem-solving, logical thinking, and debugging techniques. It also laid the foundation for understanding game development principles that can be expanded upon in more complex projects. Overall, the Brick Breaker game stands as a testament to the successful application of programming skills and creative thinking in building an enjoyable and functional software product.

4.2 FUTURE SCOPE

Future Scope of the Brick Breaker Game

The Brick Breaker game, while simple and classic, has immense potential for enhancements and expansion. Its future scope involves incorporating modern technologies and additional features to increase its appeal, interactivity, and adaptability. Below are key areas where the game can evolve

:

- **Multiplayer Mode**: Introduce competitive and co-op gameplay modes for real-time and collaborative experiences.
- **Enhanced Graphics**: Upgrade to 3D graphics, dynamic animations, and customizable themes.
- **AI Integration**: Add smart opponents, dynamic difficulty adjustment, and advanced gameplay responses.
- **Advanced Features**: Include new power-ups, story mode, and unique gameplay challenges.
- **Cross-Platform Compatibility**: Optimize for mobile, web, and gaming consoles for broader access.
- **Leaderboards and Social Features**: Add global rankings and options for social media sharing.
- **AR and VR Integration**: Enable augmented and virtual reality for immersive experiences.
- **Procedural Levels**: Create endless dynamic levels for enhanced replayability.
- **Educational Applications**: Adapt the game for teaching physics, programming, or problem-solving.
- **Monetization and Community**: Allow custom level creation, in-app purchases for cosmetic upgrades, and player-driven content sharing.

APPENDIX
SOURCE CODE

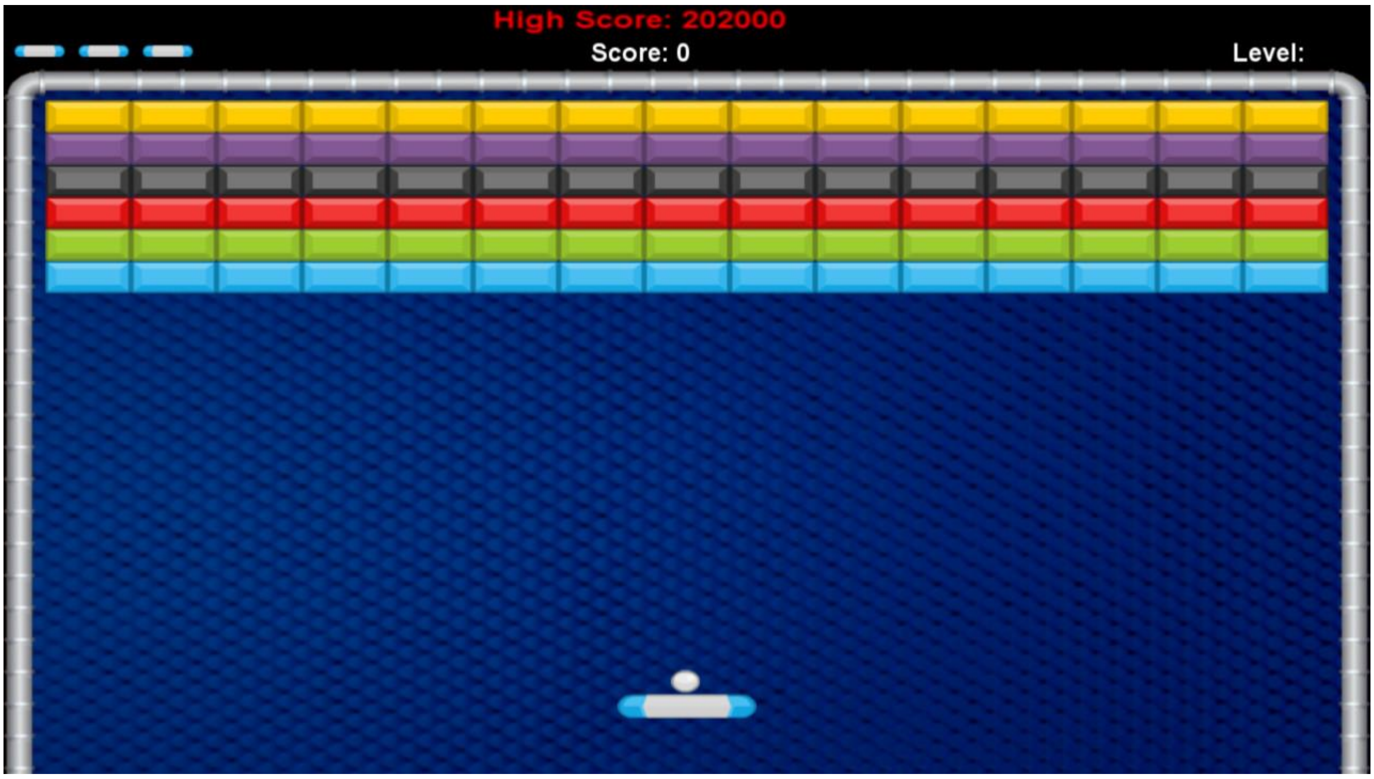
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

public class BrickBreaker extends JFrame {
   public BrickBreaker() {
      add(new GamePanel());
      setTitle("Brick Breaker");
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setResizable(false);
      setSize(700, 600);
      setLocationRelativeTo(null);
      setVisible(true);
   }

   public static void main(String[] args) {
      SwingUtilities.invokeLater(BrickBreaker::new);
   }
}

class GamePanel 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 paddleX = 310;
  private int ballPosX = 120;
  private int ballPosY = 350;
  private int ballDirX = -1;
  private int ballDirY = -2;

  private BrickGenerator bricks;

  public GamePanel() {
    bricks = new BrickGenerator(3, 7);
    setFocusable(true);
    setFocusTraversalKeysEnabled(false);
    addKeyListener(this);
    timer = new Timer(delay, this);
    timer.start();
  }

  @Override
  public void paint(Graphics g) {
    // Background
    g.setColor(Color.BLACK);
    g.fillRect(0, 0, 700, 600);
    // Borders
    g.setColor(Color.YELLOW);
    g.fillRect(0, 0, 3, 600);
    g.fillRect(0, 0, 700, 3);
    g.fillRect(697, 0, 3, 600);

    // Score
    g.setColor(Color.WHITE);
    g.setFont(new Font("Arial", Font.BOLD, 20));


g.drawString("Score: " + score, 550, 30);

    // Paddle
    g.setColor(Color.GREEN);
    g.fillRect(paddleX, 550, 100, 8);

    // Ball
    g.setColor(Color.RED);
    g.fillOval(ballPosX, ballPosY, 20, 20);

    // Bricks
    bricks.draw((Graphics2D) g);

    // Game over
    if (totalBricks <= 0) {
        play = false;
        ballDirX = 0;
        ballDirY = 0;
        g.setColor(Color.RED);
```

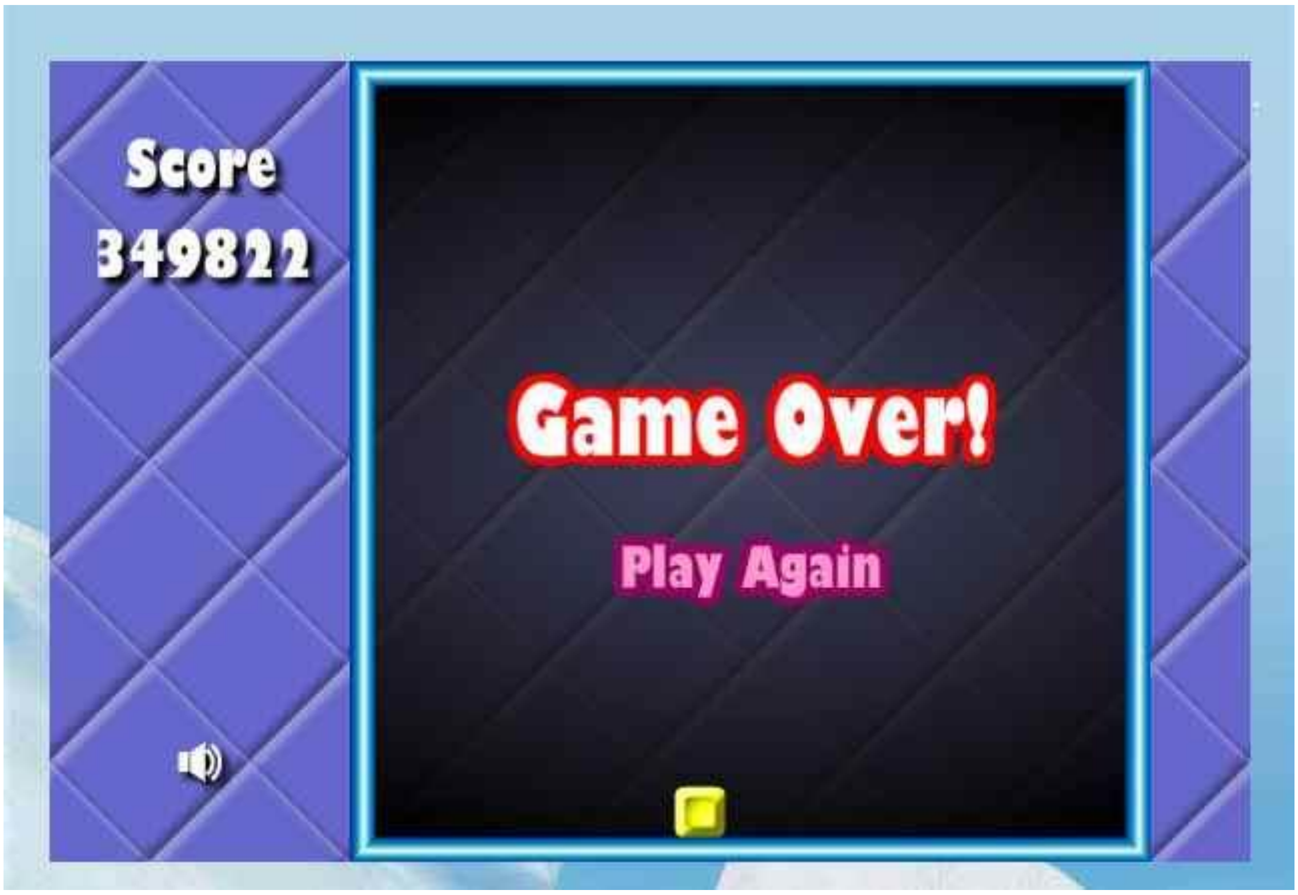
```
            g.setFont(new Font("Arial", Font.BOLD, 30));
            g.drawString("You Won!", 260, 300);
            g.drawString("Score: " + score, 270, 340);
        }

        if (ballPosY > 570) {
            play = false;
            ballDirX = 0;
            ballDirY = 0;
            g.setColor(Color.RED);
            g.setFont(new Font("Arial", Font.BOLD, 30));
            g.drawString("Game Over", 250, 300);
            g.drawString("Score: " + score, 270, 340);
        }

        g.dispose();
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (play) {
            if (new Rectangle(ballPosX, ballPosY, 20, 20).intersects(new
Rectangle(paddleX, 550, 100, 8))) {
                ballDirY = -ballDirY;
            }

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

                        Rectangle brickRect = new Rectangle(brickX, brickY,
brickWidth, brickHeight);
                        Rectangle ballRect = new Rectangle(ballPosX, ballPosY, 20, 20);

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

                            if (ballPosX + 19 <= brickRect.x || ballPosX + 1 >= brickRect.x
+ brickWidth) {
                        ballDirX = -ballDirX;
                            } else {
                        ballDirY = -ballDirY;
                            }

                            break;
                        }
                    }
                }
            }

            ballPosX += ballDirX;
            ballPosY += ballDirY;

            if (ballPosX < 0) {
                ballDirX = -ballDirX;
            }
            if (ballPosY < 0) {
                ballDirY = -ballDirY;
            }
            if (ballPosX > 670) {
                ballDirX = -ballDirX;
            }
        }

        repaint();
    }

    @Override
    public void keyTyped(KeyEvent e) {}

    @Override
    public void keyReleased(KeyEvent e) {}

    @Override
    public void keyPressed(KeyEvent e) {

       if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
            if (paddleX >= 600) {
                paddleX = 600;
            } else {
                moveRight();
            }
        }
        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            if (paddleX <= 10) {
                paddleX = 10;
            } else {
                moveLeft();
            }
        }
        if (e.getKeyCode() == KeyEvent.VK_SPACE) {
            play = true;
        }
    }

    private void moveRight() {
        play = true;
        paddleX += 20;
    }

    private void moveLeft() {
        play = true;
        paddleX -= 20;
    }
}

class BrickGenerator {
    public int[][] map;
    public int brickWidth;
    public int brickHeight;

    public BrickGenerator(int row, int col) {
        map = new int[row][col];
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                map[i][j] = 1; // 1 means the brick is visible
            }
        }
        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

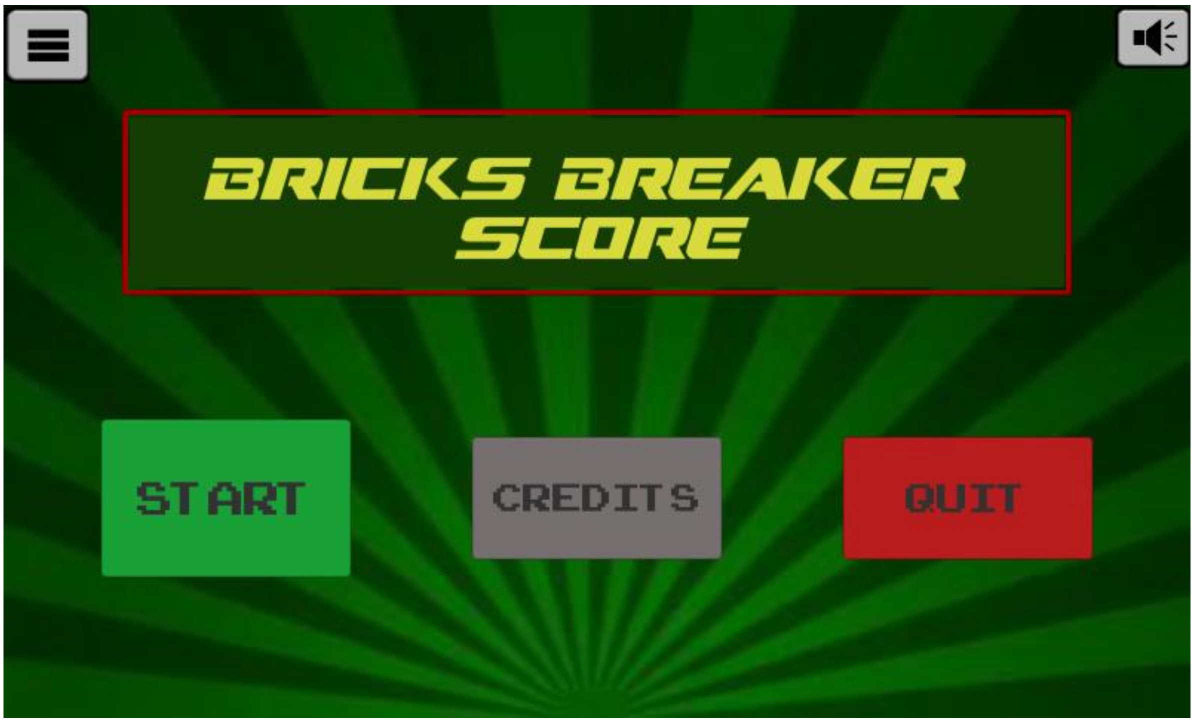
SCREENSHOT



Main Game Window: Represents the primary gameplay interface, including the paddle, ball, and bricks.



Game Over Screen: Indicates the end of the game and showcases the player's final score.



Start Menu Screen: The Opening Display With Options Like "Start" , "Quit" Etc., To Begin Or Exit The Game.

REFERENCES:

BOOKS:

1. **"Head First Java" by Kathy Sierra and Bert Bates** – Covers object-oriented programming and event handling.
2. **"Java: The Complete Reference" by Herbert Schildt** – Provides insights into Swing and core Java APIs like Timer and Graphics.
3. **"Core Java Volume I – Fundamentals" by Cay S. Horstmann** – Explains Swing, AWT components, and user input handling.
4. **"Programming with Java: A Primer" by E. Balagurusamy** – Introduces Java basics and GUI programming.
5. **"Java Swing" by Marc Loy et al.** – A detailed guide for creating GUI applications with Swing.

WEBSITES:

1. **Oracle Java Documentation** – Official tutorials on Java and Swing.
2. **GeeksforGeeks** – Guides on Java, Swing, and game development.
3. **W3Schools** – Java basics and GUI programming.
4. **TutorialsPoint** – Tutorials on Java and graphics programming.
5. **JavaTPoint** – Comprehensive Java and Swing concepts.

YOUTUBE LINKS:

1. **ProgrammingKnowledge** – Java tutorials for beginners and GUI programming.
2. **CodeWithHarry** – Java game development and Swing basics.
3. **Telusko** – Java concepts, including event handling and GUI.
4. **Bro Code** – Java tutorials on building games and using Swing.
5. **freeCodeCamp.org** – Java tutorials and graphical programming projects.