SNAKE GAME USING JAVA

SEMINAR REPORT

Submitted by

SREEDEV SREENIVASAN

To

The APJ Abdul Kalam Technological University

In partial fulfilment of the requirements for the award of the Degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science and Engineering

HEERA COLLEGE OF ENGINEERING AND TECHNOLOGY

PANAVOOR P.O,TRIVANDRUM,KERALA-696 568

20-November-2025

HEERA COLLEGE OF ENGINEERING AND TECHNOLOGY

Nedumangad, Thiruvananthapuram

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the microproject work report entitled 'SNAKE GAME USING JAVA' submitted by **SREEDEV SREENIVASAN** of Third Semester, B.Tech in Computer Science & Engineering has been successfully carried out under our supervision and guidance in partial fulfillment of the requirements of the microproject for the subject PBCST304 – Object Oriented Programming, prescribed by APJ Abdul Kalam Technological University, Kerala. It is further certified that this work is a bonafide record of the microproject carried out by the above students during the academic year 2025–2026.

Miss Sinija	Miss Shaeba CS
(Project Guide)	Professor and Head
Designation, CSE	Dept.of CSE
HCET, Nedumangad	HCET, Nedumangad

DECLARATION

I am hereby declare that the microproject report SNAKE GAME USING JAVA submitted for the partial fulfilment of the requirements of the Third Semester Microproject for the subject Object Oriented Programming, prescribed by APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under the guidance of Asst. Prof. Miss Sinija. The submission represents my ideas in my own words and where ideas or words of others have not been included. I have adequately and accurately cited and referred the original sources. I also declare that I have adhered to the ethics and integrity and have nit misinterpreted or fabricated any data or idea or fact or source in our submission .I understand that any violation of the above will be a case for disciplinary action by the institution and/or the University and can also evoke penal actions from the sources which have thus not been properly cited or from whom proper permission has not been obtained .This report has not previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Nedumangad

Student Name: SREEDEV SREENIVASAN

Date: 22-10-2025

INSTITUTION VISION

To be an Institute of repute recognised for excellence in education, innovation and social contribution.

INSTITUTION MISSION

M1: Infrastructural Relevance: Develop, maintain and manage our campus for our stakeholders.

M2: Life Long Learning: Encourage our stakeholders to participate in lifelong learning through industry and academic interactions.

M3: Social Connect: Organize socially relevant outreach programs for the benefit of humanity.

DEPARTMENT VISION

To create industry ready and socially skilled Computer Science Engineers.

DEPARTMENT MISSION

M1: Provide a learning platform that encourages thinking and analytical ability in the area of computer software and hardware.

M2: Inculcate and lifelong and professional skills through the interaction of academicians and industrialists. M3: Engage with society through social programs in and out of campus.

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

Graduates of Computer Science & Engineering will be able to:

PEO1: Professional Practice Apply Engineering practices required for Software development, Hardware development and Embedded systems.
PEO2: Entrepreneurial Skills Exhibit innovation, Self – confidence and team work skills in the organization and society.

PEO3: Technological Competence and Adaptability Graduates shall be competent in the evolving field of computer science by adapting to new tools, technologies, and industry practices, ensuring continued professional and career growth

PROGRAM SPECIFIC OUTCOMES (PSOs)

Student of the Computer Science and Engineering program will:

PSO1: Professional Skills: Ability to understand the architecture and working of computer hardware and software system.

PSO2: Design and Development Skills: Ability to design and develop software for technology application to fulfill industrial and social needs.

PROGRAM OUTCOMES (POs)

Engineering Graduates 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, research literature, and analyze complex engi- neering 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

INTRODUCTION

The Snake Game is one of the most classic arcade games and has been recreated using various technologies over the years. In this project, I have developed a simple yet functional version of the Snake Game using Java, one of the most widely used programming languages in the world. The game is built using Java Swing, a lightweight GUI toolkit for building graphical applications, and it implements basic features such as movement control, collision detection, and score tracking.

The gameplay is straightforward: the player controls a snake using the keyboard arrow keys. A piece of food appears randomly on the screen, and the snake must eat the food to grow longer. Each time the snake eats, the player's score increases. The challenge arises as the snake grows — if it collides with the walls or its own body, the game ends. The game window updates continuously using a timer, and event listeners are used to detect key presses.

This project not only serves as a fun and nostalgic game but also helps to understand and apply fundamental programming concepts such as object-oriented programming, data structures (like arrays and lists), GUI design, animation using timers, and eventdriven programming in Java.

The project was developed in the Eclipse IDE, and it runs completely offline without the need for external libraries or internet access. It is a great example of how a simple game can demonstrate powerful programming concepts in an interactive and enjoyable way.

OBJECTIVES

The primary objective of this project is to develop a fully functional and interactive Snake Game using the Java programming language. This project is designed to provide students with hands-on experience in applying programming concepts in a practical scenario, allowing them to transform theoretical knowledge into a working software application. By implementing the Snake Game, students gain the opportunity to enhance their logical thinking, problem-solving abilities, and understanding of fundamental programming constructs.

The Snake Game project aims to create an environment where the player controls a snake that moves continuously within a defined boundary, consuming food items to grow longer while avoiding collisions with walls or its own body. The project emphasizes the importance of structured programming, where each component of the game, such as the snake, food, score system, and collision detection, is carefully designed and implemented. This approach enables students to understand how large programs can be broken down into smaller, manageable modules, improving both readability and maintainability of the code.

One of the key objectives is to familiarize students with graphical user interface (GUI) development using Java Swing, which allows the creation of a visually appealing and interactive game interface. The project provides insight into designing windows, panels, and graphical elements such as the snake and food items, as well as integrating dynamic components like scores and game over messages. Through this, students learn how to combine graphics with programming logic to create engaging user experiences.

Another important goal of this project is to implement event-driven programming by capturing keyboard input through KeyListener. This allows real-time control of the snake's movement, enhancing the interactive nature of the game. Students also learn how to use timers to regulate the speed of the snake, control the refresh rate of the game display, and create smooth animation, all of which are critical concepts in game development.

The project further focuses on the application of arrays to store the positions of the snake's body segments and the development of algorithms to update positions, detect collisions, and manage game states such as running, eating, and game over. By designing these algorithms, students gain experience in logical reasoning, step-by-step problem solving, and efficient program design. These skills are transferable to a wide range of computer science applications beyond game development.

Additionally, this project provides a platform for students to explore software design principles, including modularity, reusability, and maintainability. Each aspect of the game, from movement logic to scoring and game termination, demonstrates how complex systems can be structured in an organized and efficient manner. The project encourages creativity in terms of visual design, snake behavior, and gameplay mechanics, allowing students to personalize the game while adhering to sound programming practices.

Ultimately, the Snake Game project aims to bridge the gap between theoretical learning and practical application, providing students with a comprehensive understanding of programming concepts, GUI development, event handling, and algorithmic problem-solving. It helps cultivate critical thinking, attention to detail, and the ability to develop functional software that is both engaging and reliable. The project also fosters a sense of accomplishment and motivation, as students can see the tangible results of their work in a complete and playable game.

By successfully completing this project, students will not only gain technical knowledge and experience in Java programming but also develop essential skills such as logical reasoning, software design, and project management. The Snake Game project serves as an excellent demonstration of how fundamental computer science concepts can be applied to create interactive and enjoyable applications, making it a valuable learning experience for all participants.

Software And Tools Used

To successfully develop the Snake Game project, several software tools and development environments are utilized. Each of these tools plays a critical role in ensuring the project is implemented efficiently, runs smoothly, and meets the required objectives. The following is a detailed description of the software and tools used for this project:

1. **Java Programming Language**

Java is a widely-used, object-oriented programming language that provides a robust platform for software development. For this project, Java is used due to its simplicity, portability, and ability to handle both graphical and logical components effectively. Java's object-oriented nature allows the program to be structured in a modular way, making it easier to manage the snake, food, scoring, and game logic as separate components. Additionally, Java provides built-in libraries and features such as arrays, loops, conditional statements, event handling, and timers, which are essential for creating the Snake Game. Java's platform independence ensures that the program can run on multiple operating systems without modification, making it an ideal choice for academic projects.

2. Eclipse IDE

Eclipse Integrated Development Environment (IDE) is used as the primary software tool for writing, compiling, and running the Java program. Eclipse provides a user-friendly interface with features like syntax highlighting, autocompletion, debugging tools, and project management, which greatly enhance development efficiency. Its integrated compiler allows for instant detection of errors, while the graphical debugger helps in identifying logical mistakes in the game. Eclipse also supports the inclusion of Java libraries, making it easier to implement GUI components and timers required for the game. Overall, Eclipse simplifies the development process and ensures that the project is organized and maintainable.

3. Java Swing Library

Java Swing is a part of the Java Foundation Classes (JFC) used to create graphical user interfaces. In the Snake Game project, Swing is employed to design and manage the game's visual components, including the game panel, snake, food items, score display, and game over messages. Swing provides flexibility in customizing graphics, colors, fonts, and layouts, enabling a professional-looking interface. Its support for components such as JPanel, JFrame, and JLabel allows for effective rendering of the game screen, smooth animation, and dynamic interaction with the user. Swing's event-handling capabilities are crucial for capturing keyboard input to control the snake, making it an essential tool for interactive game development.

4. KeyListener Interface

The KeyListener interface is used to capture and process keyboard events, enabling the player to control the snake's movement in real-time. By detecting key presses for arrow keys or WASD controls, the program can update the direction of the snake instantly, providing a responsive gaming experience. This interface is a core component of event-driven programming in Java and ensures that the game reacts accurately to user input.

5. Timer Class

The Timer class in Java is used to schedule tasks at fixed intervals, which is essential for controlling the speed of the snake and updating the game state periodically. By setting a timer delay, the program ensures that the snake moves smoothly across the screen and that the game logic, such as collision detection and score updates, is processed consistently. The timer also allows for dynamic adjustments to the game speed as the player's score increases, adding a layer of challenge and making the gameplay more engaging.

6. Graphics Class

The Graphics class in Java is utilized to draw the snake, food items, and other visual elements on the game panel. This class provides methods for drawing shapes, filling colors, setting fonts, and rendering text. In the Snake Game project, it is used to create a visually appealing interface by rendering the snake's body as rectangles or blocks, drawing food items as circles, and displaying the player's score dynamically. The Graphics class, combined with Swing components, allows for smooth and interactive game animation.

7. JDK (Java Development Kit)

The Java Development Kit is required to compile and run Java programs. It includes the Java Runtime Environment (JRE), compiler (javac), and various development tools necessary for building the Snake Game. The JDK ensures that the Java code is translated into bytecode that can run on the Java Virtual Machine (JVM), providing cross-platform compatibility. Using the latest version of JDK ensures access to updated libraries and enhanced performance, which is beneficial for creating a stable and responsive game.

8. Operating System

The project can be developed and executed on various operating systems such as Windows, Linux, or macOS because Java is platform-independent. This

flexibility allows the developer to write and run the program on any compatible system without worrying about compatibility issues.

9. Text Editor / IDE Features

Within Eclipse or any other IDE, additional features such as syntax highlighting, auto-completion, and error detection play an important role in the development process. These tools reduce coding errors, improve productivity, and help maintain the program's readability. For instance, code indentation and highlighting make it easier to understand the flow of the program, which is essential when implementing game logic, loops, and conditionals.

Summary

The combination of Java, Eclipse IDE, Swing library, KeyListener interface, Timer class, and other development tools provides a complete environment for creating a professional and interactive Snake Game. Each tool contributes to different aspects of the project—from writing the code and managing the game logic to rendering graphics and handling user input—ensuring that the final program is both functional and visually appealing. By using these tools, students gain practical experience in software development, GUI programming, event-driven programming, and algorithm design, making this project an effective and educational exercise in computer science.

CODE

```
import java.awt.*; import java.awt.event.*; import javax.swing.*; import java.util.Random;
public class SnakeGame extends JFrame {
public SnakeGame() {        setTitle("Snake Game");
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setResizable(false);        add(new GamePanel());
pack();        setLocationRelativeTo(null);        setVisible(true);
}

public static void main(String[] args) {        new SnakeGame();
}
}

class GamePanel extends JPanel implements ActionListener, KeyListener {

static final int SCREEN_WIDTH = 600;     static final int SCREEN_HEIGHT = 600;     static final int
UNIT_SIZE = 20;
static final int GAME_UNITS =
(SCREEN_WIDTH*SCREEN_HEIGHT)/UNIT_SIZE;
static final int DELAY = 100;     final int x[] = new int[GAME_UNITS];     final int y[] = new
int[GAME_UNITS];     int bodyParts = 5;
int foodX;     int foodY;     int score = 0;     char direction = 'R';     boolean running = false;
Timer timer;
Random random;

GamePanel() {        random = new Random();
setPreferredSize(new Dimension(SCREEN_WIDTH, SCREEN_HEIGHT));
setBackground(Color.black);        setFocusable(true);        addKeyListener(this);        startGame();
}

public void startGame() {        newFood();        running = true;        timer = new Timer(DELAY,this);
timer.start();
}

public void paintComponent(Graphics g) {        super.paintComponent(g);        draw(g);
}

public void draw(Graphics g) {        if(running) {
g.setColor(Color.red);
```

```
g.fillOval(foodX, foodY, UNIT_SIZE, UNIT_SIZE);            for(int i = 0; i < bodyParts; i++) {                if(i ==
0) {
g.setColor(Color.green);
} else {
g.setColor(Color.yellow);
}
g.fillRect(x[i], y[i], UNIT_SIZE, UNIT_SIZE);
}

g.setColor(Color.white);
g.setFont(new Font("Arial", Font.BOLD, 20));
g.drawString("Score: " + score, 10, 30);

} else {            gameOver(g);
}
}

public void newFood() {
foodX =
random.nextInt((int)(SCREEN_WIDTH/UNIT_SIZE)) *
UNIT_SIZE;
foodY =
random.nextInt((int)(SCREEN_HEIGHT/UNIT_SIZE)) * UNIT_SIZE;
}

public void move() {        for(int i = bodyParts; i > 0; i--) {            x[i] = x[i-1];            y[i] = y[i-1];
}

if(direction == 'U') y[0] -= UNIT_SIZE;        if(direction == 'D') y[0] += UNIT_SIZE;        if(direction == 'L')
x[0] -= UNIT_SIZE;        if(direction == 'R') x[0] += UNIT_SIZE;     }

public void checkFood() {        if((x[0] == foodX) && (y[0] == foodY)) {            bodyParts++;
score++;            newFood();
}
}

public void checkCollisions() {        for(int i = bodyParts; i > 0; i--) {            if((x[0] == x[i]) && (y[0] ==
y[i])) {                running = false;
}
}

if(x[0] < 0 || x[0] > SCREEN_WIDTH || y[0] < 0 || y[0]
> SCREEN_HEIGHT) {
running = false;
}
```

```
if(!running) timer.stop();
}

public void gameOver(Graphics g) {
g.setColor(Color.red);
g.setFont(new Font("Arial", Font.BOLD, 50));
g.drawString("GAME OVER", 150, 250);

g.setColor(Color.white);
g.setFont(new Font("Arial", Font.PLAIN, 30));
g.drawString("Score: " + score, 220, 300);
}

@Override    public void actionPerformed(ActionEvent e) {
if(running) {          move();          checkFood();          checkCollisions();
}       repaint();
}

@Override    public void keyPressed(KeyEvent e) {       switch(e.getKeyCode()) {          case
KeyEvent.VK_UP:
if(direction != 'D') direction = 'U';              break;
case KeyEvent.VK_DOWN:
if(direction != 'U') direction = 'D';              break;
case KeyEvent.VK_LEFT:
if(direction != 'R') direction = 'L';              break;
case KeyEvent.VK_RIGHT:
if(direction != 'L') direction = 'R';              break;
}
}

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

Working of the Snake Game

The Snake Game operates as an interactive program where the player controls a snake on the screen, guiding it to eat food items, grow in length, and avoid collisions with walls or its own body. The game demonstrates how a combination of logical programming, graphical rendering, and event driven input can create a functional and engaging application. The following explains the working of the game in detail:

1. Game Initialization

When the program is executed in the Eclipse IDE, the game window opens with a black or designated background and a visible playing area. The snake is initialized with a default length, typically consisting of several body segments represented using arrays to store their coordinates. A food item is randomly placed on the screen within the defined boundaries. A timer is also started, which controls the movement of the snake and refreshes the game display at regular intervals.

2. Snake Movement

The snake moves continuously in one of four directions: up, down, left, or right. The movement is implemented using arrays that store the positions of the snake's head and body segments. At each tick of the timer, the position of each segment is updated to follow the previous segment, creating the illusion of smooth movement. The direction of the snake is controlled by the player using the arrow keys or WASD keys, and the program ensures that the snake cannot reverse directly into itself to prevent instant collisions.

3. Food Consumption and Growth

When the head of the snake reaches the coordinates of the food item, the game detects this as a collision with food. Upon eating, the snake grows by adding a new segment at its tail, and the player's score is incremented by

one point. A new food item is then randomly generated at a location on the screen that does not overlap with the snake's body. This process continues, making the snake longer and increasing the challenge for the player.

4. Collision Detection

Collision detection is a critical aspect of the game that ensures proper game mechanics. The program continuously checks for two types of collisions:

- Self-Collision: The snake collides with one of its own body segments if the head's coordinates match any of the body segments.
- Wall-Collision: The snake collides with the boundary of the playing area if its head crosses the screen limits.

When either collision occurs, the game is terminated, the timer is stopped, and the game over state is triggered.

5. Score Display

The current score is displayed prominently on the game window using the Graphics class, allowing the player to track progress. Each time the snake eats a food item, the score is incremented and updated on the screen in real time. This feature provides immediate feedback to the player and encourages continued gameplay.

6. Game Over State

Upon collision, the game enters the game over state. A message such as "GAME OVER" is displayed, along with the final score achieved by the player. In some implementations, the player may have the option to restart the game, either by pressing a key (e.g., Enter) or clicking a button, which reinitializes the snake and food positions and resets the score.

7. Game Loop and Timer Functionality
8. The game is driven by a loop controlled by the Timer class. At each interval (for example, 100 milliseconds), the timer triggers the following sequence:

- Update the positions of all snake segments
- Check for collisions with food, walls, or itself
- Update the score if food is eaten
- Repaint the screen to reflect the latest positions and changes

This loop continues until the snake collides or the game is exited, creating continuous, real-time gameplay.

9. User Interaction

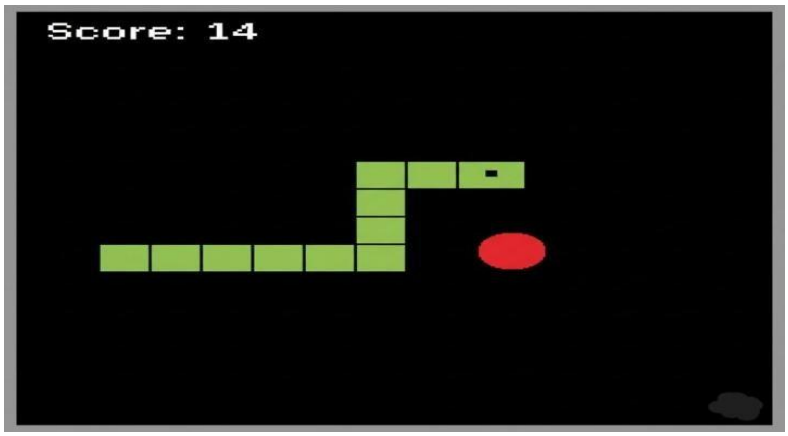
The game relies on event-driven programming to respond to user input. The KeyListener interface listens for key presses and changes the snake's direction accordingly. This makes the game interactive, as the player must react quickly to guide the snake, consume food, and avoid collisions.

CONCLUSION

In summary, the Snake Game works by continuously updating the snake's position, detecting collisions, managing growth and score, and rendering graphics in real-time. It combines algorithmic logic with interactive graphics, demonstrating fundamental programming concepts such as arrays, loops, conditionals, event handling, and GUI design. Through this implementation, students gain practical experience in developing a dynamic, responsive, and visually appealing program, bridging the gap between theory and hands-on software development.

SYSTEM OUTPUT

The output will be like this after game starts:



The Green block is the Snake and the Red circle is it's food. When the Snake eats the food then it's size will increase and the score will also increase with res

pect to the size of Snake.

When the snake go and hit to the screen wall the Game will over.

References

1. **GeeksforGeeks – Snake Game in Java**

https://www.geeksforgeeks.org/snake-game-in-java/

Step-by-step tutorial for implementing Snake Game using Java Swing and event handling.

2. **YouTube – Java Snake Game Tutorials**

Example,channel:CodeWithHarry,Telusko,

ProgrammingKnowledge

Search: "Java Snake Game tutorial"

Video tutorials for GUI setup, snake movement, collision detection, and scoring.

3. **Oracle Java Tutorials (Swing GUI)**

https://docs.oracle.com/javase/tutorial/uiswing/

Official guide for JFrame, JPanel, event handling, and timers.

4. **W3Schools – Java Basics**

https://www.w3schools.com/java/