NED UNIVERSITY OF ENGINEERING AND TECHNOLOGY
B.E / BS PROGRAMME OF USMAN INSTITUTE OF TECHNOLOGY
SOFTWARE ENGINEERING DEPARTMENT

Spring 2024

SE-312 Software Construction and Development

Project Summary Report

Semester: 6th

PROJECT TITLE:

Snake Game in Java

TEAM MEMBERS:

Name	Roll Number
KHADEEJAH ASHRAF	21B-037-SE
ALI AHMED	21B-085-SE
AQSA	21B-198-SE
ALISHBA AHMED	21B-208-SE

Project Description:

The Snake Game project is a classic implementation of the popular snake game using Java and the Swing library. The objective of this project is to create an engaging, visually appealing, and functional game where the player controls a snake to collect cherries and grow in length while avoiding collisions with the boundaries and its own tail. The project aims to provide a fun and interactive experience, showcasing effective use of object-oriented programming principles and GUI design.

Problem Statement:

To develop a simple, yet engaging snake game that provides smooth gameplay and incorporates basic game mechanics such as collision detection, score tracking, and difficulty progression.

Objectives:

- Implement a snake game with responsive controls.
- Ensure the game has clear visual indicators for score and game status.
- Provide a smooth and visually appealing user interface.
- Incorporate basic game mechanics and logic, including collision detection and random cherry spawning.

Scope:

The project focuses on developing a single-player snake game with essential features and a graphical user interface. It does not include advanced features like multiple levels, different snake skins, or multiplayer functionality.

Constraints:

- The game is designed to run on desktop platforms with Java installed.
- Limited to basic game mechanics without advanced AI or complex animations.

Frontend Languages/Framework/Libraries:

Java Swing:

Reason for Choice:
Java Swing is a robust and versatile GUI toolkit that provides a wide range of components and features for building sophisticated user interfaces. It is part of the Java Foundation Classes (JFC) and is widely used for developing desktop applications.

Contribution:
Swing is used to create the main game window and user interface elements, such as the score display, game messages, and boundaries. Its flexibility allows for custom rendering of the game components and smooth user interactions.

Java AWT (Abstract Window Toolkit):

Reason for Choice:
Java AWT provides essential tools for handling low-level events and creating graphical components. It complements Swing by offering foundational graphics capabilities.

Contribution:
AWT is utilized for handling input events (like key presses) and managing the game window's properties. It also assists in basic drawing operations needed for the game graphics.

Java 2D API:

Reason for Choice:
The Java 2D API extends the capabilities of AWT and Swing by providing advanced 2D graphics features, including text, shapes, and image rendering with enhanced control over appearance and performance.

Contribution:
The Java 2D API is used to render the snake, cherries, and other game elements. It allows for smooth animations, custom shapes, and detailed control over colors and strokes, enhancing the visual quality of the game.

Backend Languages/Framework/Libraries:

Java:

Reason for Choice:
Java is versatile, portable, and well-suited for developing complex applications like games.

Contribution:
It implements game logic, including snake movement, collision detection, score tracking, and game state management, leveraging its robust object-oriented features.

Java.util.Timer:

Reason for Choice:
Provides a simple way to schedule tasks at fixed intervals, essential for game loops.

Contribution:
Manages the game loop, ensuring regular updates and rendering, which creates a smooth gameplay experience.

Java.io:

Reason for Choice:
Offers comprehensive classes for handling system input/output and file operations.

Contribution:
Loads external resources, like the cherry image, with proper exception handling to manage I/O errors.

The Best/Unique Feature of your Project:

The standout feature of this Snake game project is its **dynamic difficulty adjustment based on player performance**. Unlike traditional Snake games that maintain a constant speed and difficulty level, this project adjusts the game's speed and complexity in real-time. As the player scores more points, the snake's speed increases, adding an extra layer of challenge and excitement.

Significance and Value:

- ***Enhanced Engagement:*** The dynamic difficulty keeps the game challenging and engaging for players of all skill levels, preventing it from becoming monotonous.

- ***Skill Development:*** Players are encouraged to improve their reflexes and strategic planning, as the game progressively becomes more difficult.

- ***Replayability:*** The increasing difficulty ensures that the game remains interesting over multiple play sessions, promoting replayability.

This feature adds significant value by providing a tailored gaming experience that adapts to the player's skill, making the game more enjoyable and competitive.

Project and Lab Topics Synchronization:

The Snake game project aligns with several topics covered in the lab sessions. Below, We will detail the specific lab topics and how they are applied in the project:

Basic Java Programming

Lab Topic:
Introduction to Java syntax, variables, control structures, and basic input/output.

Project Alignment:
The project uses fundamental Java constructs such as variables, loops, conditionals, and methods to manage game logic and player interactions.

Object-Oriented Programming (OOP)

Lab Topic:
Concepts of classes, objects, inheritance, polymorphism, and encapsulation.

Project Alignment:

- Classes and Objects: The game is structured using multiple classes such as Game, Point, Snake, and Main.
- Encapsulation: Private fields and public methods ensure encapsulation. For example, the Point class encapsulates coordinates with getter and setter methods.
- Inheritance and Polymorphism: Although inheritance and polymorphism are not applied because the class structure doesn't require hierarchical relationships, and the design prioritizes simplicity and single responsibility principles.

Event-Driven Programming

Lab Topic:
Handling events in Java, especially key and mouse events.

Project Alignment:
The game uses key event listeners to handle user inputs (e.g., moving the snake and pausing the game). The KeyListener class processes these events to update the game state.

Graphics and GUI Development

Lab Topic:
Introduction to Java Swing for creating graphical user interfaces.

Project Alignment:
The game uses JPanel for rendering graphics and JFrame for the game window. The paintComponent method is overridden to draw the game elements, including the snake and cherry.

Collections and Data Structures

Lab Topic:
Understanding and using Java collections such as lists, sets, and maps.

Project Alignment:

- The Snake class uses an ArrayList<Point> to manage the segments of the snake's body.
- Data structures are crucial for efficiently updating and rendering the game state.
-

Concurrency and Timers

Lab Topic:
Basics of concurrency in Java and using timers for scheduling tasks.

Project Alignment:

- The game uses a Timer to control the game loop, which updates the game state and refreshes the display at regular intervals.
- The dynamic difficulty adjustment feature modifies the timer's delay to change the game speed.

Exception Handling

Lab Topic:
Exception handling mechanisms in Java.

Project Alignment:
The project includes exception handling for loading resources (e.g., the cherry image), ensuring the game can run even if some resources are missing.

File I/O

Lab Topic:
Reading from and writing to files in Java.

Project Alignment:
Although not extensively used, the game demonstrates the basic file I/O in loading the cherry image using ImageIO.read.

Test-Driven Development:

Lab Topic:
Unit Testing through Code or Document

Project Alignment:
JUnit testing was employed to ensure the reliability and functionality of the game logic.