Snake

Drew Jordan

Project Demo and Project Link

Github

Key Components

Board.Java

Apple.java

Snake.java

Main.java

Board.java

1) Game Initialization and Control: The pseudocode outlines a snake game where a Board class is defined, extending JPanel and implementing ActionListener. It initializes the game by setting up the game board, spawning the initial apple, and handling key inputs for game controls. If the game is over, the player can restart by pressing 'R'.

2) Game Mechanics and Rendering: It includes methods for game mechanics like spawning apples, checking for collisions, and moving the snake. The draw method renders the game elements (snake and apple) on the board, and a drawGameOver method to display a game-over message. The game is updated and redrawn at regular intervals using a timer.

3) Collision and Game Over Handling: The game logic includes checking for collisions (either with the snake itself or the board boundaries) and growing the snake when it eats an apple. The game ends (setting the gameOver flag) upon collision, and a method restartGame is available to reset and start a new game.

Apple.java

1) Apple Position Management: The class manages the position of an apple in the game, using a Point object to store its x and y coordinates. This allows the apple to be placed at specific locations within the game board.

2) Constructor and Position Initialization: It includes a constructor that takes x and y coordinates as parameters. These values are used to initialize the apple's position, creating a new Point object representing its location on the game board.

3) Position Handling Methods: The class provides methods to get and set the apple's position. getPosition returns the current position of the apple, and setPosition allows updating the apple's location by accepting new x and y coordinates, enhancing the flexibility in moving the apple within the game.

Snake.java

1) Initialization and Structure of the Snake: The class initializes a snake composed of segments (each represented by a Point) with a specified initial length and scale. It calculates the starting position of the snake, typically at the center of the game board, and sets the initial direction.

2) Movement and Growth Mechanics: The move method updates the snake's position based on its current direction, adding a new head segment in the direction of movement and removing the last segment, simulating the snake's forward motion. The grow method allows the snake to increase in size by adding a new segment without removing the last one, typically used when the snake eats an apple.

3) Utility Methods for Game Mechanics: It includes methods to set and get the snake's direction, retrieve the snake's parts (segments), and obtain the head of the snake (the first segment). Additional utility methods like getSCALE for returning the size of each segment, and setSnakeParts for updating the snake's segments, facilitate game management and interaction with other game components.

Main.java

1) Game Initialization: It initializes the snake game using Swing, a Java graphical user interface toolkit. The main method schedules a job for the event-dispatching thread, which is responsible for managing the application's GUI.

2) GUI Creation: The createAndShowGUI method sets up the game's graphical user interface. It creates a new window (JFrame) titled "Snake Game", makes it non-resizable, and adds an instance of the Board class (which likely contains the game's logic and visual components).

3) Window Configuration and Display: The method configures the window to close the application when the user exits it (EXIT_ON_CLOSE). It also ensures that the window is sized appropriately to fit its components (pack method) and centers the window on the screen. Finally, it makes the window visible, displaying the game to the user.

Project Analysis

1) At the beginning of the project I planned on making a react app with a java backend to visualize and show financial metrics such as money in and money out.
2) Switched to building the classic snake game as I already knew react and wanted to challenge myself to learn something new. In the example of my project, I learned how to make a GUI in java as well as take in keyboard input to use a game controls.
3) Testing: Throughout the project I was constantly running unit test to test certain components of the game
4) Problems: At first, I was running into issues with maven regarding the pom.xml file which is needed for compiling the program, however, after research into maven I was able to figure it out