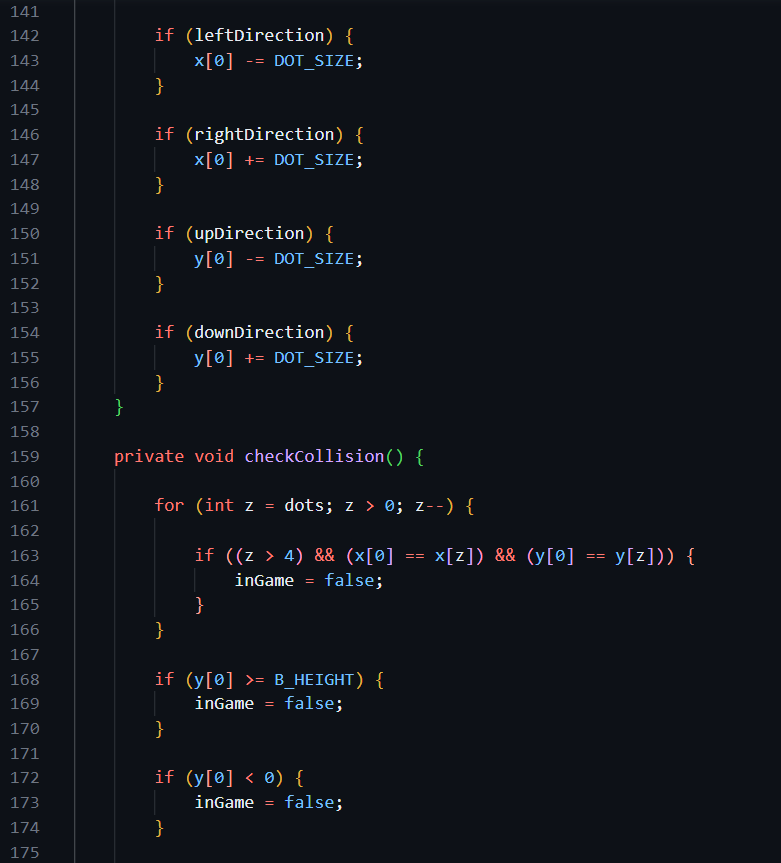
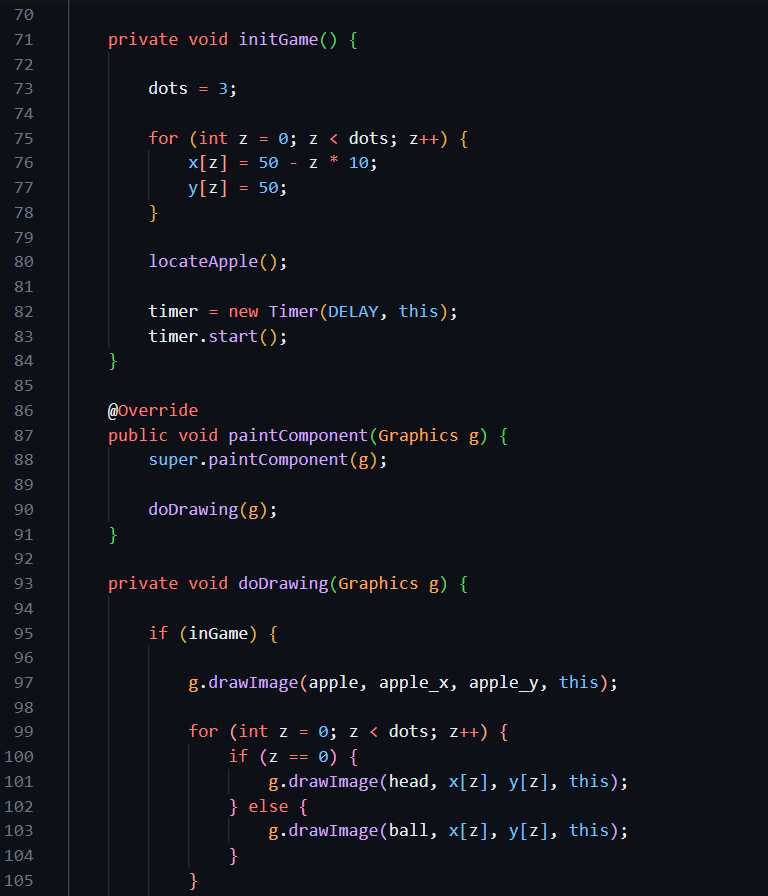
Documentation

Board.java



This Java code represents a simple implementation of the classic Snake game using the Swing library for creating a graphical user interface (GUI). Let's go through the code step by step:

**Class Overview:**

1. **Board Class:**
   * This class extends **JPanel** and implements the **ActionListener** interface.
   * It represents the game board where the Snake moves and interacts with the Apple.
2. **Instance Variables:**
   * **B\_WIDTH** and **B\_HEIGHT**: Width and height of the board.
   * **DOT\_SIZE**: Size of each Snake segment.
   * **ALL\_DOTS**: Maximum number of Snake segments.
   * **RAND\_POS**: Random position range for placing the Apple.
   * **DELAY**: Timer delay for controlling the speed of the game.
   * **x** and **y**: Arrays to store the coordinates of each Snake segment.
   * **dots**: Current number of Snake segments.
   * **apple\_x** and **apple\_y**: Coordinates of the Apple.
   * Directional boolean variables (**leftDirection**, **rightDirection**, **upDirection**, **downDirection**) to control the movement direction of the Snake.
   * **inGame**: Indicates whether the game is still active.
   * **timer**: A Swing Timer to trigger regular updates of the game state.
   * **ball**, **apple**, and **head**: Images representing the Snake segments, Apple, and Snake head, respectively.

**Methods:**

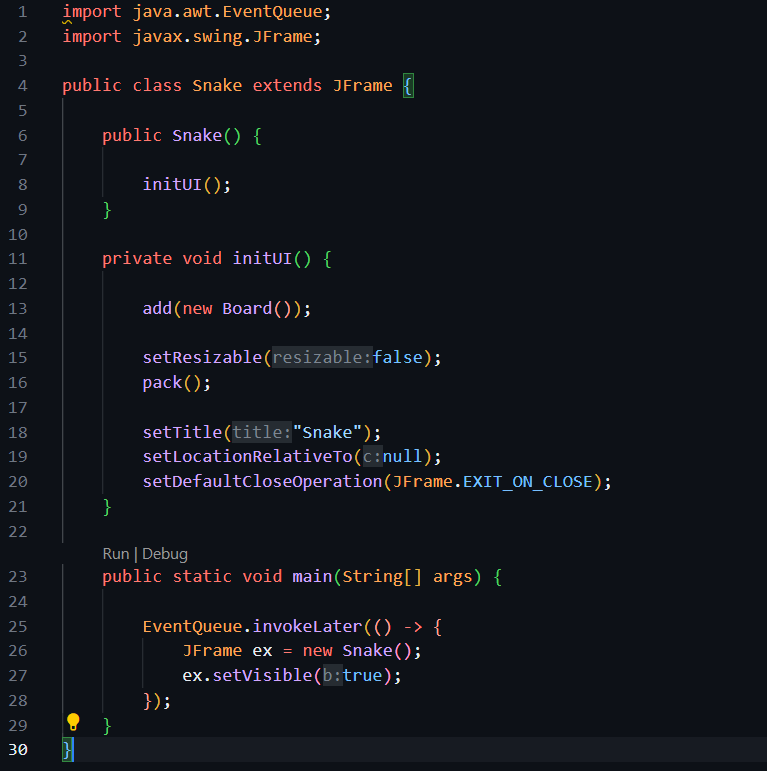
1. **Board() Constructor:**
   * Initializes the board by calling **initBoard()**.
2. **initBoard() Method:**
   * Sets up the initial configuration of the board.
   * Adds a **KeyAdapter** to handle keyboard input.
   * Sets the background color, makes the board focusable, and sets its preferred size.
   * Calls **loadImages()** and **initGame()**.
3. **loadImages() Method:**
   * Loads images for the Snake segments, Apple, and Snake head using **ImageIcon**.
4. **initGame() Method:**
   * Initializes the game state, including the initial Snake position, Apple placement, and starting the Timer.
5. **paintComponent(Graphics g) Method:**
   * Overrides the **paintComponent** method to draw the game graphics.
   * Calls **doDrawing(g)**.
6. **doDrawing(Graphics g) Method:**
   * Draws the Apple and Snake segments on the board.
   * If the game is over, displays a "Game Over" message.
7. **gameOver(Graphics g) Method:**
   * Displays the "Game Over" message on the board.
8. **checkApple() Method:**
   * Checks if the Snake head has collided with the Apple and increments the Snake length if true.
9. **move() Method:**
   * Moves the Snake segments based on the current direction.
10. **checkCollision() Method:**
    * Checks for collisions with the board boundaries and itself.
    * Stops the game if a collision is detected.
11. **locateApple() Method:**
    * Randomly places the Apple on the board.
12. **actionPerformed(ActionEvent e) Method:**
    * Handles the Timer's action events.
    * Checks for Apple collision, updates the game state, and triggers a repaint.
13. **TAdapter Inner Class:**
    * Extends **KeyAdapter** to handle keyboard input.
    * Adjusts the direction based on arrow key presses.

**Main Points:**

* The game uses a Timer to update the game state at regular intervals.
* Snake movement is controlled by arrow keys.
* Collision detection is performed to check for Apple collisions and collisions with the board or the Snake itself.
* The game stops when a collision occurs, and a "Game Over" message is displayed.

This code provides a basic structure for a Snake game and can be extended for additional features and improvements.

Snake.java



This Java code defines a simple Snake game using the Swing framework for creating graphical user interfaces (GUIs). Here's a breakdown of what the code does:

1. **Class Definition:** The code defines a class named `Snake` that extends the `JFrame` class. This means that the `Snake` class is a frame (window) for the Snake game.
2. **Constructor (`Snake()`):** The `Snake` class has a constructor that initializes the UI (User Interface) by calling the `initUI()` method.
3. **`initUI()` Method:** This method sets up the user interface for the game. It adds a new instance of the `Board` class (presumably representing the game board) to the frame. It also configures some properties of the frame, such as making it non-resizable, setting the title to "Snake," centring it on the screen, and specifying that the application should exit when the frame is closed.
4. **`main()` Method:** The main method is the entry point of the program. It uses the `EventQueue.invokeLater()` method to ensure that the GUI is constructed and modified on the event dispatching thread. Inside the `invokeLater` block, an instance of the `Snake` class is created and made visible.

Overall, this code is a basic structure for a Snake game using Java Swing. The game board is likely implemented in the `Board` class, which is added to the frame in the `initUI()` method. The `main()` method creates an instance of the `Snake` class and displays the game window. The actual game logic and behaviour are expected to be implemented in the `Board` class or other related classes.