Java Snake Game: Methods Documentation

August 5, 2025

Contents

1 Introduction

This document provides a comprehensive list of all methods used in the Java Snake Game project. The game is built using Java Swing and consists of four main classes: `SnakeGame`, `GamePanel`, `Snake`, and `Apple`. Each section below details the methods defined in these classes, including their signatures and functionality. This report serves as a reference for understanding the implementation details of the game.

2 SnakeGame Class

The `SnakeGame` class is the entry point of the application, responsible for initializing the game window.

2.1 main(String[] args)

Signature: `public static void main(String[] args)`

Description: Initializes the game by creating a `JFrame`, adding a `GamePanel`, and setting up the window properties (non-resizable, centered, visible).

3 GamePanel Class

The `GamePanel` class extends `JPanel` and implements `ActionListener`. It handles the game loop, rendering, and user input.

3.1 GamePanel()

Signature: `public GamePanel()`

Description: Constructor that sets up the panel's size, background, focus, and key listener. Initializes the "Play Again" button and starts the game.

3.2 startGame()

Signature: `public void startGame()`

Description: Initializes a new game by creating a `Snake` and `Apple`, starting the game timer, and setting the game state to running.

3.3 restartGame()

Signature: `public void restartGame()`

Description: Restarts the game by stopping the timer, reinitializing the game state, repainting the panel, and requesting focus.

3.4 paintComponent(Graphics g)

Signature: `public void paintComponent(Graphics g)`

Description: Overrides `JPanel`'s method to call the `draw` method for rendering the game.

3.5 draw(Graphics g)

Signature: `public void draw(Graphics g)`

Description: Renders the game elements (apple, snake, score) if running, or displays the game-over screen otherwise.

3.6 gameOver(Graphics g)

Signature: `public void gameOver(Graphics g)`

Description: Displays the game-over screen with the final score and shows the "Play Again" button.

3.7 checkApple()

Signature: `public void checkApple()`

Description: Checks if the snake's head collides with the apple. If so, grows the snake and spawns a new apple.

3.8 checkCollisions()

Signature: `public void checkCollisions()`

Description: Checks for collisions between the snake's head and its body or the game boundaries, stopping the game if a collision occurs.

3.9 actionPerformed(ActionEvent e)

Signature: `public void actionPerformed(ActionEvent e)`

Description: Implements `ActionListener`. Updates the game state (moves snake, checks apple and collisions) and triggers a repaint if the game is running.

3.10 MyKeyAdapter Inner Class

3.10.1 keyPressed(KeyEvent e)

Signature: `public void keyPressed(KeyEvent e)`

Description: Handles key presses (arrow keys or W/A/S/D) to set the snake's direction.

4 Snake Class

The `Snake` class manages the snake's position, movement, and growth.

4.1 move()

Signature: `public void move()`

Description: Updates the snake's position by shifting body segments and moving the head based on the current direction.

4.2 grow()

Signature: `public void grow()`

Description: Increases the snake's length by incrementing the body parts count.

4.3 setDirection(char newDir)

Signature: `public void setDirection(char newDir)`

Description: Sets the snake's direction, preventing 180-degree turns (e.g., from left to right).

4.4 getHeadX()

Signature: `public int getHeadX()`

Description: Returns the x-coordinate of the snake's head.

4.5 getHeadY()

Signature: `public int getHeadY()`

Description: Returns the y-coordinate of the snake's head.

5 Apple Class

The `Apple` class handles the apple's position and spawning logic.

5.1 spawnApple(int gridWidth, int gridHeight, Snake snake)

Signature: `public void spawnApple(int gridWidth, int gridHeight, Snake snake)`

Description: Spawns the apple at a random position on the grid, ensuring it does not overlap with the snake's body.