**SnakeGame.java** - The main class to run the game

import javax.swing.JFrame;

public class SnakeGame {

public static void main(String[] args) {

JFrame frame = new JFrame();

GamePanel gamePanel = new GamePanel();

frame.add(gamePanel);

frame.setTitle("Snake Game");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setResizable(false);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

}

1. **GamePanel.java** - The panel where the game is rendered:

import javax.swing.JPanel;

import javax.swing.Timer;

import java.awt.Dimension;

import java.awt.Graphics;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

public class GamePanel extends JPanel implements ActionListener {

private static final int TILE\_SIZE = 25;

private static final int WIDTH = 600;

private static final int HEIGHT = 600;

private static final int ALL\_TILES = (WIDTH \* HEIGHT) / (TILE\_SIZE \* TILE\_SIZE);

private final int[] x = new int[ALL\_TILES];

private final int[] y = new int[ALL\_TILES];

private int bodyParts = 6;

private int foodX;

private int foodY;

private char direction = 'R';

private boolean running = false;

private Timer timer;

private final int DELAY = 150;

public GamePanel() {

setPreferredSize(new Dimension(WIDTH, HEIGHT));

setBackground(Color.black);

setFocusable(true);

addKeyListener(new MyKeyAdapter());

startGame();

}

public void startGame() {

newFood();

running = true;

timer = new Timer(DELAY, this);

timer.start();

}

@Override

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, TILE\_SIZE, TILE\_SIZE);

for (int i = 0; i < bodyParts; i++) {

if (i == 0) {

g.setColor(Color.green);

g.fillRect(x[i], y[i], TILE\_SIZE, TILE\_SIZE);

} else {

g.setColor(new Color(45, 180, 0));

g.fillRect(x[i], y[i], TILE\_SIZE, TILE\_SIZE);

}

}

} else {

gameOver(g);

}

}

public void newFood() {

foodX = (int) (Math.random() \* (WIDTH / TILE\_SIZE)) \* TILE\_SIZE;

foodY = (int) (Math.random() \* (HEIGHT / TILE\_SIZE)) \* TILE\_SIZE;

}

public void move() {

for (int i = bodyParts; i > 0; i--) {

x[i] = x[i - 1];

y[i] = y[i - 1];

}

switch (direction) {

case 'U':

y[0] = y[0] - TILE\_SIZE;

break;

case 'D':

y[0] = y[0] + TILE\_SIZE;

break;

case 'L':

x[0] = x[0] - TILE\_SIZE;

break;

case 'R':

x[0] = x[0] + TILE\_SIZE;

break;

}

}

public void checkFood() {

if ((x[0] == foodX) && (y[0] == foodY)) {

bodyParts++;

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) {

running = false;

}

if (x[0] >= WIDTH) {

running = false;

}

if (y[0] < 0) {

running = false;

}

if (y[0] >= HEIGHT) {

running = false;

}

if (!running) {

timer.stop();

}

}

public void gameOver(Graphics g) {

String msg = "Game Over";

g.setColor(Color.red);

g.drawString(msg, (WIDTH / 2) - 30, HEIGHT / 2);

}

@Override

public void actionPerformed(ActionEvent e) {

if (running) {

move();

checkFood();

checkCollisions();

}

repaint();

}

private class MyKeyAdapter extends KeyAdapter {

@Override

public void keyPressed(KeyEvent e) {

switch (e.getKeyCode()) {

case KeyEvent.VK\_LEFT:

if (direction != 'R') {

direction = 'L';

}

break;

case KeyEvent.VK\_RIGHT:

if (direction != 'L') {

direction = 'R';

}

break;

case KeyEvent.VK\_UP:

if (direction != 'D') {

direction = 'U';

}

break;

case KeyEvent.VK\_DOWN:

if (direction != 'U') {

direction = 'D';

}

break;

}

}

}

}