```java
import javax.swing.*;

import java.awt.*;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.util.Random;


public class SnakeGame extends JPanel implements KeyListener {

    private static final int SCREEN_WIDTH = 600;

    private static final int SCREEN_HEIGHT = 600;

    private static final int UNIT_SIZE = 25;

    private static final int GAME_UNITS = (SCREEN_WIDTH * SCREEN_HEIGHT) / UNIT_SIZE;

    private static final int DELAY = 100;

    private final int[] x = new int[GAME_UNITS];

    private final int[] y = new int[GAME_UNITS];

    private int bodyParts = 6;

    private int applesEaten;

    private int appleX;

    private int appleY;

    private char direction = 'R';

    private boolean running = false;

    private Timer timer;

    private JLabel scoreLabel;


    public SnakeGame() {

        this.setPreferredSize(new Dimension(SCREEN_WIDTH, SCREEN_HEIGHT));

        this.setBackground(Color.BLACK);

        this.setFocusable(true);

        this.addKeyListener(this);

        scoreLabel = new JLabel("Score: 0");

        scoreLabel.setForeground(Color.WHITE);

        scoreLabel.setFont(new Font("Arial", Font.BOLD, 20));
    add(scoreLabel);
    startGame();
}

public void startGame() {
    newApple();
    running = true;
    timer = new Timer(DELAY, e -> {
        if (running) {
            move();
            checkApple();
            checkCollisions();
            repaint();
        }
    });
    timer.start();
}

public void paintComponent(Graphics g) {
    super.paintComponent(g);
    draw(g);
}

public void draw(Graphics g) {
    if (running) {
        g.setColor(Color.RED);
        g.fillOval(appleX, appleY, UNIT_SIZE, UNIT_SIZE);

        for (int i = 0; i < bodyParts; i++) {
            if (i == 0) {
                g.setColor(Color.GREEN);
```

```java
                } else {
                    g.setColor(new Color(45, 180, 0));
                }
                g.fillRect(x[i], y[i], UNIT_SIZE, UNIT_SIZE);
            }
            g.setColor(Color.WHITE);
            g.setFont(new Font("Arial", Font.BOLD, 20));
            g.drawString("Score: " + applesEaten, 5, 20);
        } else {
            gameOver(g);
        }
    }

    public void newApple() {
        appleX = new Random().nextInt((int) (SCREEN_WIDTH / UNIT_SIZE)) * UNIT_SIZE;
        appleY = new Random().nextInt((int) (SCREEN_HEIGHT / UNIT_SIZE)) * UNIT_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] -= UNIT_SIZE;
                break;
            case 'D':
                y[0] += UNIT_SIZE;
                break;
            case 'L':
```

```
            x[0] -= UNIT_SIZE;
            break;
        case 'R':
            x[0] += UNIT_SIZE;
            break;
    }
}

public void checkApple() {
    if (x[0] == appleX && y[0] == appleY) {
        bodyParts++;
        applesEaten++;
        newApple();
    }
}

public void checkCollisions() {
    // Check if head collides with body
    for (int i = bodyParts; i > 0; i--) {
        if (x[0] == x[i] && y[0] == y[i]) {
            running = false;
        }
    }
    // Check if head touches left border
    if (x[0] < 0) {
        running = false;
    }
    // Check if head touches right border
    if (x[0] >= SCREEN_WIDTH) {
        running = false;
    }
```

```
        // Check if head touches top border
        if (y[0] < 0) {
            running = false;
        }
        // Check if head touches bottom border
        if (y[0] >= SCREEN_HEIGHT) {
            running = false;
        }
        if (!running) {
            timer.stop();
        }
    }

    public void gameOver(Graphics g) {
        g.setColor(Color.RED);
        g.setFont(new Font("Ink Free", Font.BOLD, 75));
        FontMetrics metrics = getFontMetrics(g.getFont());
        g.drawString("Game Over", (SCREEN_WIDTH - metrics.stringWidth("Game Over")) / 2,
SCREEN_HEIGHT / 2);
        g.setColor(Color.WHITE);
        g.setFont(new Font("Arial", Font.BOLD, 20));
        g.drawString("Score: " + applesEaten, 5, 20);
    }

    @Override
    public void keyTyped(KeyEvent e) {
    }

    @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;
        }
    }

    @Override
    public void keyReleased(KeyEvent e) {
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Snake Game");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setContentPane(new SnakeGame());
        frame.pack();

        frame.setLocationRelativeTo(null);

        frame.setVisible(true);

    }

}
```