Visvesvaraya Technological University

Jnana Sangama, Belagavi – 590018

Object Oriented Concepts [17CS42] Activity Report

on

SNAKE GAME USING JAVA APPLET

Submitted in the partial fulfilment of the requirements for the award of the degree of

Bachelor of Engineering

in

Information Science & Engineering

for the academic year 2018 – 2019

Submitted by

Siddharth S 1JS17IS070

under the guidance of

Dr. Dayananda P

Associate Professor & Head, Department of ISE, JSSATE

2018 – 2019

Department of Information Science & Engineering

JSS Academy of Technical Education

JSS Campus, Dr. Vishnuvardhan Road, Bengaluru – 560060

INTRODUCTION

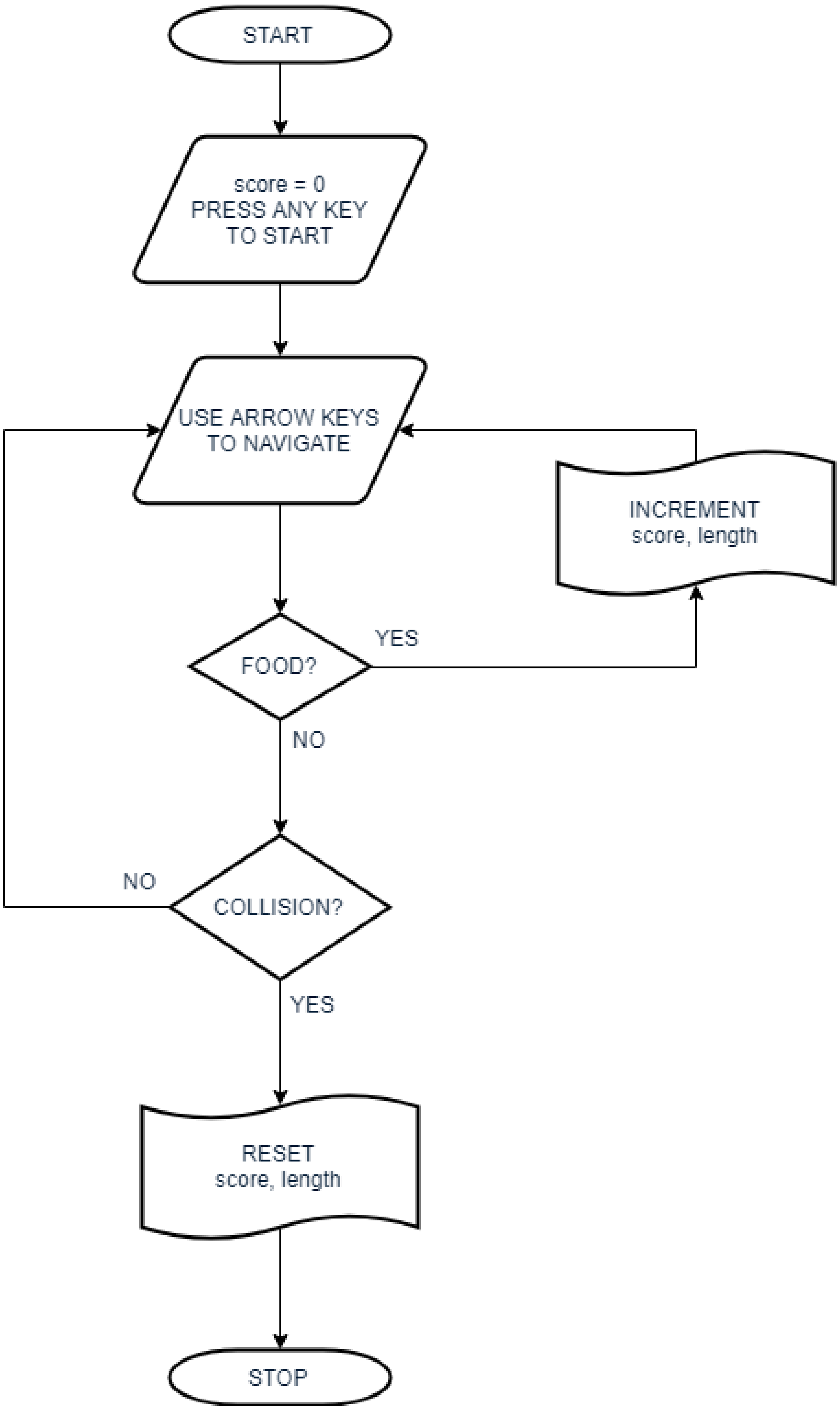
The **Applet** class is contained in the **java.applet** package. **Applet** contains several methods that give you detailed control over the execution of your applet.

There are 2 types of applets. The first are those based directly on the **Applet** class. These applets use the **Abstract Window Toolkit (AWT)** to provide the graphic user interface (or use no GUI at all). This style of applet has been available since Java was first created.

The second type of applets are those based on the Swing class **JApplet**. Swing applets use the Swing classes to provide the GUI. Swing offers a richer and often easier-to-use user interface than does the AWT. Thus, Swing-based applets are now the most popular. However, traditional AWT-based applets are still used, especially when only a very simple user interface is required. Thus, both AWT- and Swing-based applets are valid.

Because **JApplet** inherits **Applet**, all the features of **Applet** are also available in **JApplet**. All applets are subclasses (either directly or indirectly) of **Applet**. Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer.

IMPLEMENTATION



Flowchart of the program

SOURCE CODE

The **Main.java** file has the following code.

```
import java.awt.Color;

import javax.swing.JFrame;

public class Main {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		JFrame jf = new JFrame();
		Gameplay gameplay = new Gameplay();

		jf.setBounds(10, 10, 905, 700);
		jf.setBackground(Color.DARK_GRAY);
		jf.setResizable(false);
		jf.setVisible(true);
		jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		jf.add(gameplay);
	}
}
```

The **Gameplay.java** file has the following code.

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.Random;

import javax.swing.ImageIcon;
import javax.swing.JPanel;
import javax.swing.Timer;

public class Gameplay extends JPanel implements KeyListener, ActionListener {
	private int[] xLength, yLength;

	private int[] enemyX = { 25, 50, 75, 100, 125, 150, 175, 200, 225, 250,
275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625, 650,
675, 700, 725, 750, 775, 800, 825, 850 };
	private int[] enemyY = { 75, 100, 125, 150, 175, 200, 225, 250, 275, 300,
325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625 };

	private ImageIcon titleImage, upMouth, rightMouth, downMouth, leftMouth,
snakeBody, enemyImage;

	private boolean up, right, down, left, gameOver;

	private Timer timer;
	private Random random = new Random();

	private final int delay = 100;
	private int length, moves, score, x = random.nextInt(34), y = 
random.nextInt(23);

	public Gameplay() {
		titleImage = new 
ImageIcon(Gameplay.class.getResource("/assets/snaketitle.jpg"));
		upMouth = new 
ImageIcon(Gameplay.class.getResource("/assets/upmouth.png"));
		rightMouth = new 
ImageIcon(Gameplay.class.getResource("/assets/rightmouth.png"));
		downMouth = new 
ImageIcon(Gameplay.class.getResource("/assets/downmouth.png"));
		leftMouth = new 
ImageIcon(Gameplay.class.getResource("/assets/leftmouth.png"));
		snakeBody = new 
ImageIcon(Gameplay.class.getResource("/assets/snakeimage.png"));
		enemyImage = new 
ImageIcon(Gameplay.class.getResource("/assets/enemy.png"));
		xLength = new int[750];
		yLength = new int[750];
		up = right = down = left = gameOver = false;
		moves = score = 0;
		length = 3;

		addKeyListener(this);
		setFocusable(true);
		setFocusTraversalKeysEnabled(false);
		timer = new Timer(delay, this);
		timer.start();
	}

	@Override
	public void paint(Graphics g) {
		if (moves == 0) {
			xLength[0] = 100;
			xLength[1] = 75;
			xLength[2] = 50;

			yLength[0] = yLength[1] = yLength[2] = 100;
		}

		g.setColor(Color.white);
		g.drawRect(24, 10, 851, 55);

		titleImage.paintIcon(this, g, 25, 11);

		g.setColor(Color.WHITE);
		g.drawRect(24, 74, 851, 577);

		g.setColor(Color.black);
		g.fillRect(25, 75, 850, 575);

		g.setColor(Color.white);
		g.setFont(new Font("arial", Font.PLAIN, 14));
		g.drawString("Scores: " + score, 780, 30);

		g.setColor(Color.white);
		g.setFont(new Font("arial", Font.PLAIN, 14));
		g.drawString("Length: " + length, 780, 50);
		rightMouth.paintIcon(this, g, xLength[0], yLength[0]);

		for (int i = 0; i < length; i++) {
			if (i == 0 && up) {
				upMouth.paintIcon(this, g, xLength[i], yLength[i]);
			}
			if (i == 0 && right) {
				rightMouth.paintIcon(this, g, xLength[i], yLength[i]);
			}
			if (i == 0 && down) {
				downMouth.paintIcon(this, g, xLength[i], yLength[i]);
			}
			if (i == 0 && left) {
				leftMouth.paintIcon(this, g, xLength[i], yLength[i]);
			}
			if (i != 0) {
				snakeBody.paintIcon(this, g, xLength[i], yLength[i]);
			}
		}

		if ((enemyX[x] == xLength[0] && enemyY[y] == yLength[0])) {
			score++;
			length++;
			x = random.nextInt(34);
			y = random.nextInt(23);
		}
		enemyImage.paintIcon(this, g, enemyX[x], enemyY[y]);

		for (int i = 1; i < length; i++) {
			if (xLength[i] == xLength[0] && yLength[i] == yLength[0]) {
				up = right = down = left = false;
				gameOver = true;
				g.setColor(Color.white);
				g.setFont(new Font("arial", Font.BOLD, 50));
				g.drawString("GAME OVER", 300, 300);
				g.setFont(new Font("arial", Font.BOLD, 20));
				g.drawString("SPACE TO RESTART", 350, 340);
			}
		}

		g.dispose();
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		// TODO Auto-generated method stub
		timer.start();

		if (right) {
			for (int i = length - 1; i >= 0; i--)
				yLength[i + 1] = yLength[i];

			for (int i = length; i >= 0; i--) {
				if (i == 0)
					xLength[i] = xLength[i] + 25;
				else
					xLength[i] = xLength[i - 1];
```

```
                    if (xLength[i] > 850)
                        xLength[i] = 25;
                }
                repaint();
            }

            if (left) {
                for (int i = length - 1; i >= 0; i--)
                    yLength[i + 1] = yLength[i];

                for (int i = length; i >= 0; i--) {
                    if (i == 0)
                        xLength[i] = xLength[i] - 25;
                    else
                        xLength[i] = xLength[i - 1];

                    if (xLength[i] < 25)
                        xLength[i] = 850;
                }
                repaint();
            }

            if (up) {
                for (int i = length - 1; i >= 0; i--)
                    xLength[i + 1] = xLength[i];

                for (int i = length; i >= 0; i--) {
                    if (i == 0)
                        yLength[i] = yLength[i] - 25;
                    else
                        yLength[i] = yLength[i - 1];

                    if (yLength[i] < 75)
                        yLength[i] = 625;
                }
                repaint();
            }

            if (down) {
                for (int i = length - 1; i >= 0; i--)
                    xLength[i + 1] = xLength[i];

                for (int i = length; i >= 0; i--) {
                    if (i == 0)
                        yLength[i] = yLength[i] + 25;
                    else
                        yLength[i] = yLength[i - 1];

                    if (yLength[i] > 625)
                        yLength[i] = 75;
                }
                repaint();
            }
        }

        @Override
        public void keyPressed(KeyEvent e) {
            // TODO Auto-generated method stub
            if (e.getKeyCode() == KeyEvent.VK_SPACE) {
```

```
			moves = 0;
			score = 0;
			length = 3;
			gameOver = false;
			repaint();
		}

		if (e.getKeyCode() == KeyEvent.VK_RIGHT && !gameOver) {
			moves++;
			right = true;
			if (!left)
				right = true;
			else {
				right = false;
				left = true;
			}
			up = false;
			down = false;
		}

		if (e.getKeyCode() == KeyEvent.VK_LEFT && !gameOver) {
			moves++;
			left = true;
			if (!right)
				left = true;
			else {
				left = false;
				right = true;
			}
			up = false;
			down = false;
		}

		if (e.getKeyCode() == KeyEvent.VK_UP && !gameOver) {
			moves++;
			up = true;
			if (!down)
				up = true;
			else {
				up = false;
				down = true;
			}
			left = false;
			right = false;
		}

		if (e.getKeyCode() == KeyEvent.VK_DOWN && !gameOver) {
			moves++;
			down = true;
			if (!up)
				down = true;
			else {
				down = false;
				up = true;
			}
			right = false;
			left = false;
		}
```

```
	}

	@Override
	public void keyReleased(KeyEvent e) {
		// TODO Auto-generated method stub

	}

	@Override
	public void keyTyped(KeyEvent e) {
		// TODO Auto-generated method stub

	}
}
```

OUTPUT

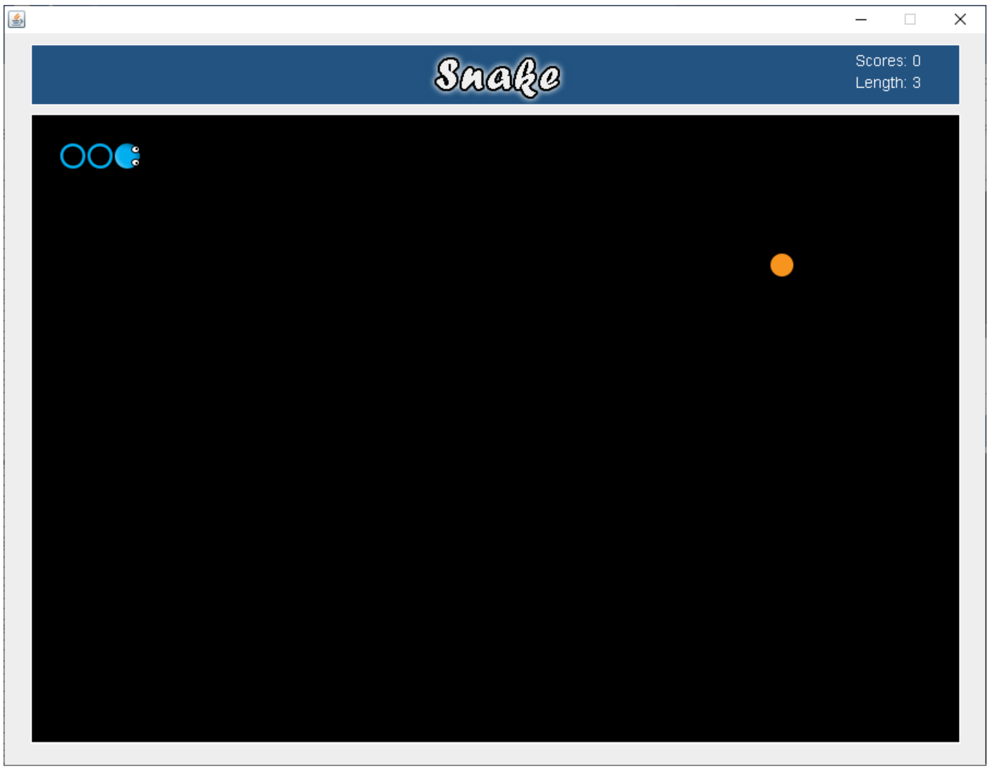


Figure 1 Initial state of the game

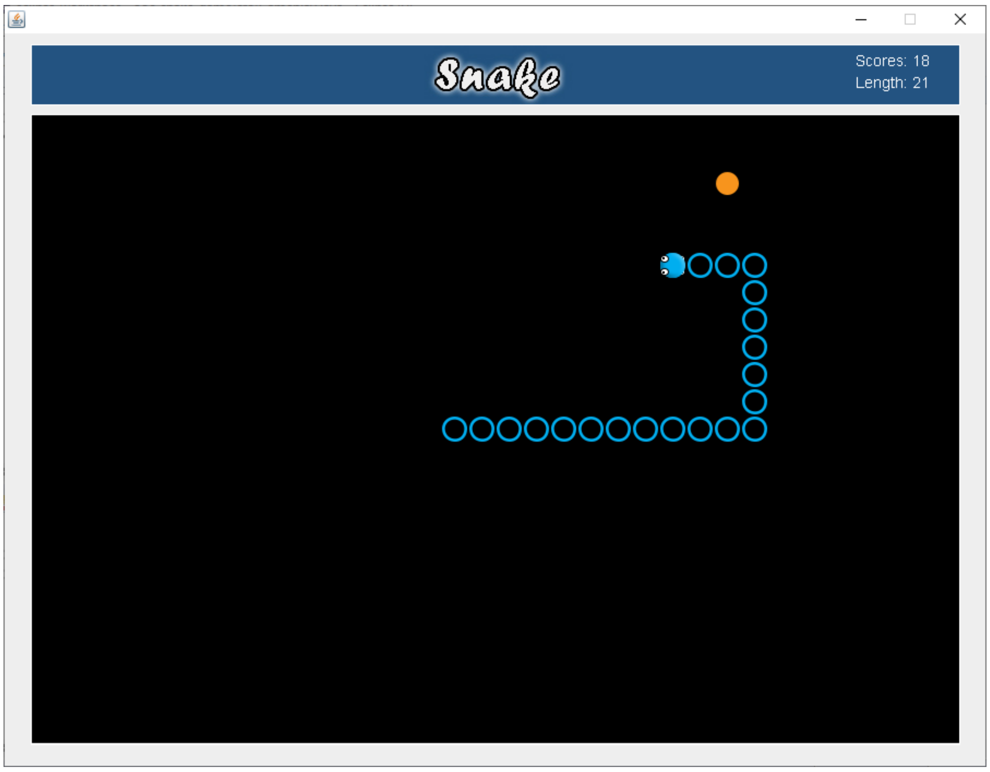


Figure 2 While the game is being played

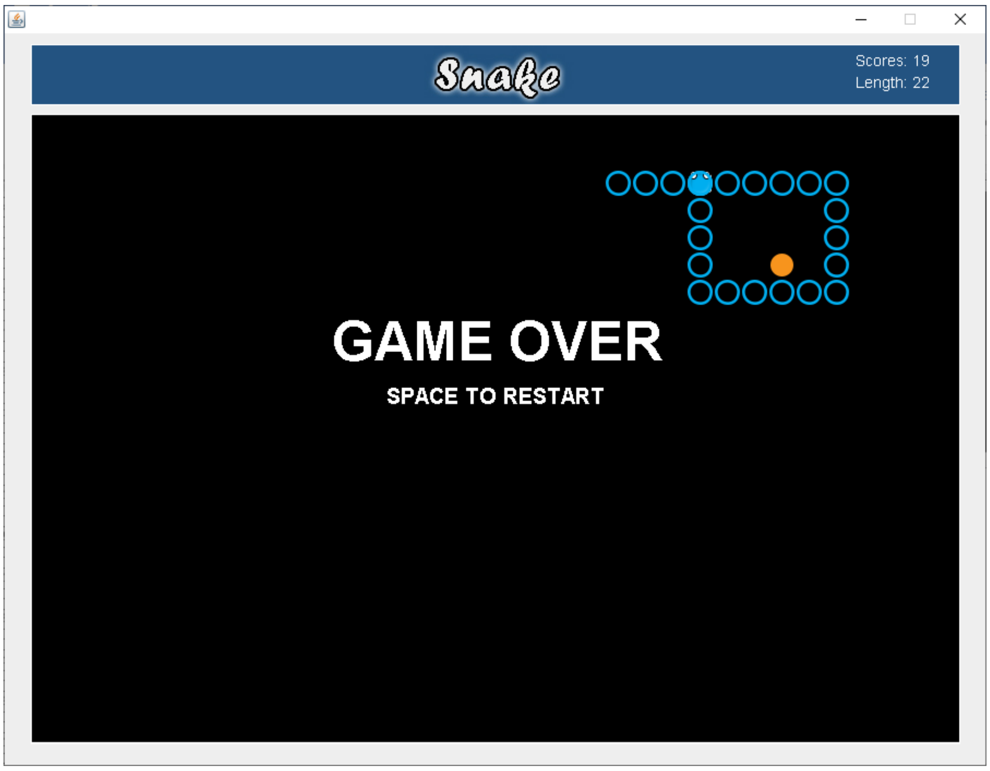


Figure 3 When the game stops

NOTE: The game cannot be run on a web browser as no latest browser supports Java Applets.