শিক্ষা নিয়ে গড়বো দেশ তথ্য-প্রযুক্তির বাংলাদেশ

Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh

LAB REPORT-08

COURSE NO.- PROG 112
COURSE TITLE- OBJECT ORIENTED PROGRAMMING SESSIONAL

SUBMITTED BY

Mehrin Farzana
ID: 2101013
Department of IRE
Session :2021-2022
Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh

SUBMITTED TO

Md.Toukir Ahmed
Lecturer
Department of IRE
Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh

Date of Submission: 16 October, 2023

Problem No.: 01

Problem Name: Snake game developed in Java.

Code:

SnakeGame Class:

```
package snakegame;

public class SnakeGame {


   public static void main(String[] args) {
      new GameFrame();
   }

}
```

GameFrame class:

```
package snakegame;

import javax.swing.JFrame;

public class GameFrame extends JFrame{

        GameFrame(){

                this.add(new GamePanel());
                this.setTitle("Snake");
                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                this.setResizable(false);
                this.pack();
                this.setVisible(true);
                this.setLocationRelativeTo(null);

        }
}
```

GamePanel class:

```
package snakegame;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.Random;

public class GamePanel extends JPanel implements ActionListener{

	static final int SCREEN_WIDTH = 1300;
	static final int SCREEN_HEIGHT = 750;
	static final int UNIT_SIZE = 50;
	static final int GAME_UNITS = (SCREEN_WIDTH*SCREEN_HEIGHT)/(UNIT_SIZE*UNIT_SIZE);
	static final int DELAY = 175;
	final int x[] = new int[GAME_UNITS];
	final int y[] = new int[GAME_UNITS];
	int bodyParts = 6;
	int applesEaten;
	int appleX;
	int appleY;
	char direction = 'R';
	boolean running = false;
	Timer timer;
	Random random;

	GamePanel(){
		random = new Random();
		this.setPreferredSize(new Dimension(SCREEN_WIDTH,SCREEN_HEIGHT));
		this.setBackground(Color.black);
		this.setFocusable(true);
		this.addKeyListener(new MyKeyAdapter());
		startGame();
	}
	public void startGame() {
		newApple();
		running = true;
		timer = new Timer(DELAY,this);
		timer.start();
	}
	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		draw(g);
	}
	public void draw(Graphics g) {

		if(running) {
```

```
                        /*
                        for(int i=0;i<SCREEN_HEIGHT/UNIT_SIZE;i++) {
                                g.drawLine(i*UNIT_SIZE, 0, i*UNIT_SIZE,
SCREEN_HEIGHT);
                                g.drawLine(0, i*UNIT_SIZE, SCREEN_WIDTH,
i*UNIT_SIZE);
                        }
                        */
                        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);
                                        g.fillRect(x[i], y[i], UNIT_SIZE, UNIT_SIZE);
                                }
                                else {
                                        g.setColor(new Color(45,180,0));
                                        //g.setColor(new
Color(random.nextInt(255),random.nextInt(255),random.nextInt(255)));
                                        g.fillRect(x[i], y[i], UNIT_SIZE, UNIT_SIZE);
                                }
                        }
                        g.setColor(Color.red);
                        g.setFont( new Font("Ink Free",Font.BOLD, 40));
                        FontMetrics metrics = getFontMetrics(g.getFont());
                        g.drawString("Score: "+applesEaten, (SCREEN_WIDTH -
metrics.stringWidth("Score: "+applesEaten))/2, g.getFont().getSize());
                }
                else {
                        gameOver(g);
                }

        }
        public void newApple(){
                appleX                                                  =
random.nextInt((int)(SCREEN_WIDTH/UNIT_SIZE))*UNIT_SIZE;
                appleY                                                  =
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] = y[0] - UNIT_SIZE;
                        break;
```

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

    }
    public void checkApple() {
            if((x[0] == appleX) && (y[0] == appleY)) {
                bodyParts++;
                applesEaten++;
                newApple();
            }
    }
    public void checkCollisions() {
            //checks 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) {
            //Score
            g.setColor(Color.red);
```

```
                g.setFont( new Font("Ink Free",Font.BOLD, 40));
                FontMetrics metrics1 = getFontMetrics(g.getFont());
                g.drawString("Score: "+applesEaten, (SCREEN_WIDTH -
metrics1.stringWidth("Score: "+applesEaten))/2, g.getFont().getSize());
                //Game Over text
                g.setColor(Color.red);
                g.setFont( new Font("Ink Free",Font.BOLD, 75));
                FontMetrics metrics2 = getFontMetrics(g.getFont());
                g.drawString("Game Over", (SCREEN_WIDTH -
metrics2.stringWidth("Game Over"))/2, SCREEN_HEIGHT/2);
        }
        @Override
        public void actionPerformed(ActionEvent e) {

                if(running) {
                        move();
                        checkApple();
                        checkCollisions();
                }
                repaint();
        }

        public 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;
                        }
                }
        }
}
```

Output:

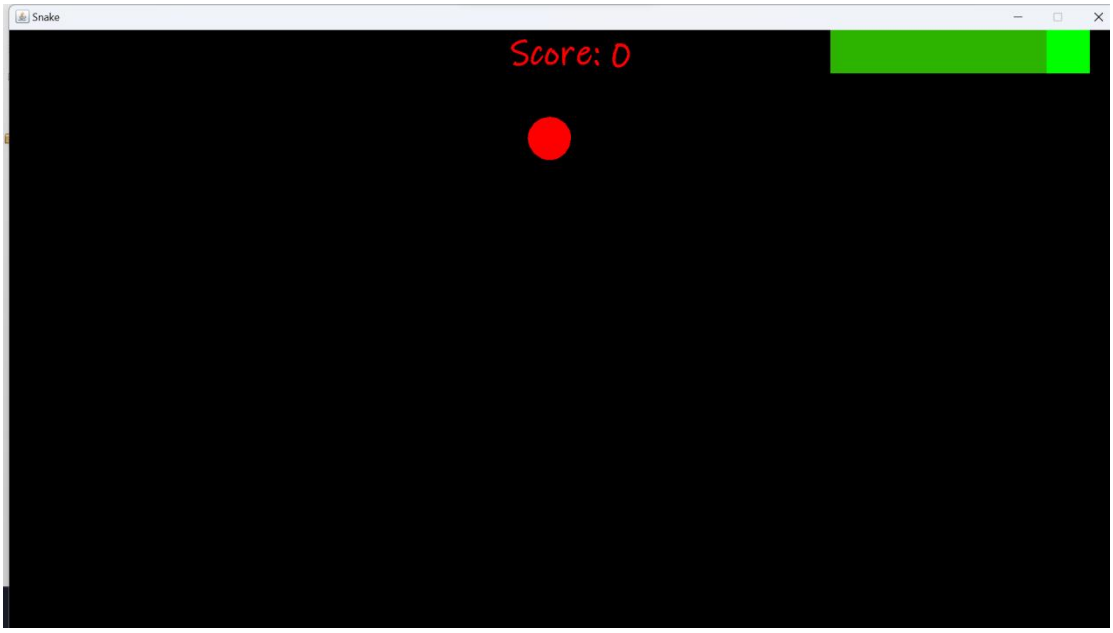

Fig 1.1: Output window.

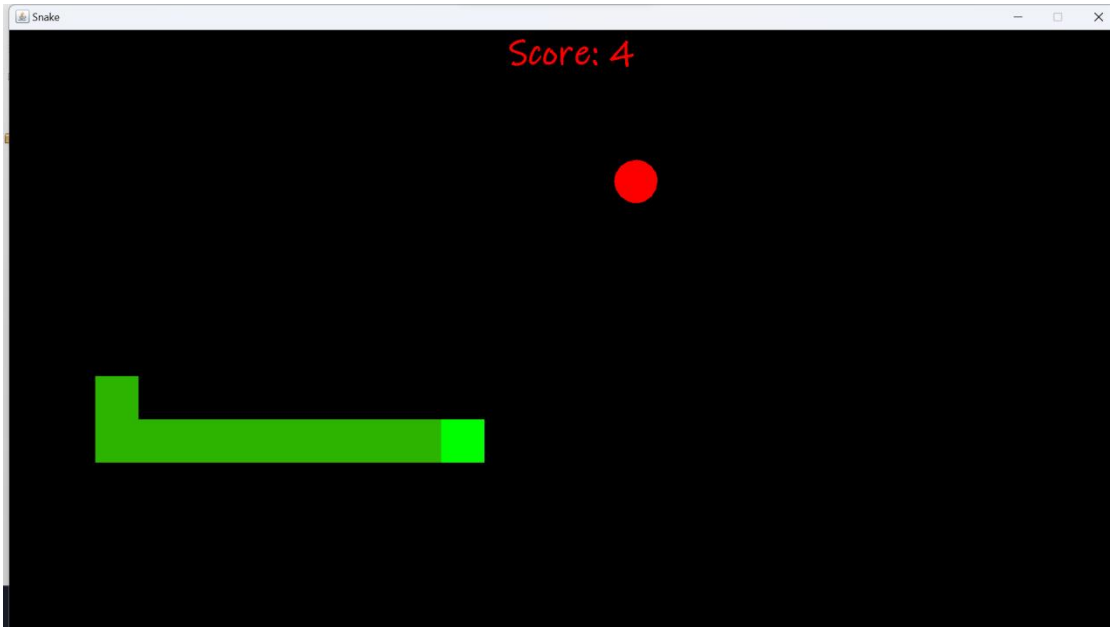

Fig 1.2: Output window.

Fig 1.3: Output window.

Explanation:

This game implements graphics and key actions. Uses inheritnce and method overriding and other features of OOP like encapsulation, abstraction, polymorphism.

Problem No.: 02

Problem Name: Abstract lass in java.

Code:

```
package test;

interface DogInterface{
   //all methods are abstract here
   public void bark();     //cannot have method body
   //{
      // System.out.println("Bark!");
   //}
   public abstract void favFood();

}

abstract class Dog{
   public void bark(){
      System.out.println("Bark!");
   }
   public abstract void favFood();

}

class Chihuahua extends Dog{
   //must define favFood method
   @Override
   public void favFood(){
      System.out.println("I love Biscuits!");
   }

}
class Winnie implements DogInterface{
   //must implement all methods
   public void bark(){
      System.out.println("Bark!");
   }
   public void favFood(){
      System.out.println("I love Biscuits!");
   }
}
```

```
public class Test {

    public static void main(String[] args) {
        //Dog d = new Dog();   //Cannnt instantite an abstract class
        Chihuahua c = new Chihuahua();
        c.bark();
        c.favFood();

    }
}
```

Output:

```
run:
Bark!
I love Biscuits!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig 2.1: Output on console.

Explanation:

This code shows the properties of abstract class that is , it cannot be instantiated and abstract methods cant have bodies and the child class must implement all abstract classes.

It also shows that interfaces assume that all methods in it are abstract that must be implemented in child class.

Problem No.: 03

Problem Name: Interface in java.

Code:

```
package test;

interface DogInterface{
   //all methods are abstract here
   public void bark();     //cannot have method body
   //{
      // System.out.println("Bark!");
   //}
   public abstract void favFood();

}

class Winnie implements DogInterface{
   //must implement all methods
   public void bark(){
      System.out.println("Bark!");
   }
   public void favFood(){
      System.out.println("I love Biscuits!");
   }
}
public class Test {

   public static void main(String[] args) {
      //Dog d = new Dog();   //Cannnt instantite an abstract class
      Winnie c = new Winnie();
      c.bark();
      c.favFood();

   }
}
```

Output:

```
run:
Bark!
I love Biscuits!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig 3.1: Output window.

Explanation:

This code shows the properties of interface that is , it assumes that all methods in it are abstract that must be implemented in child class.