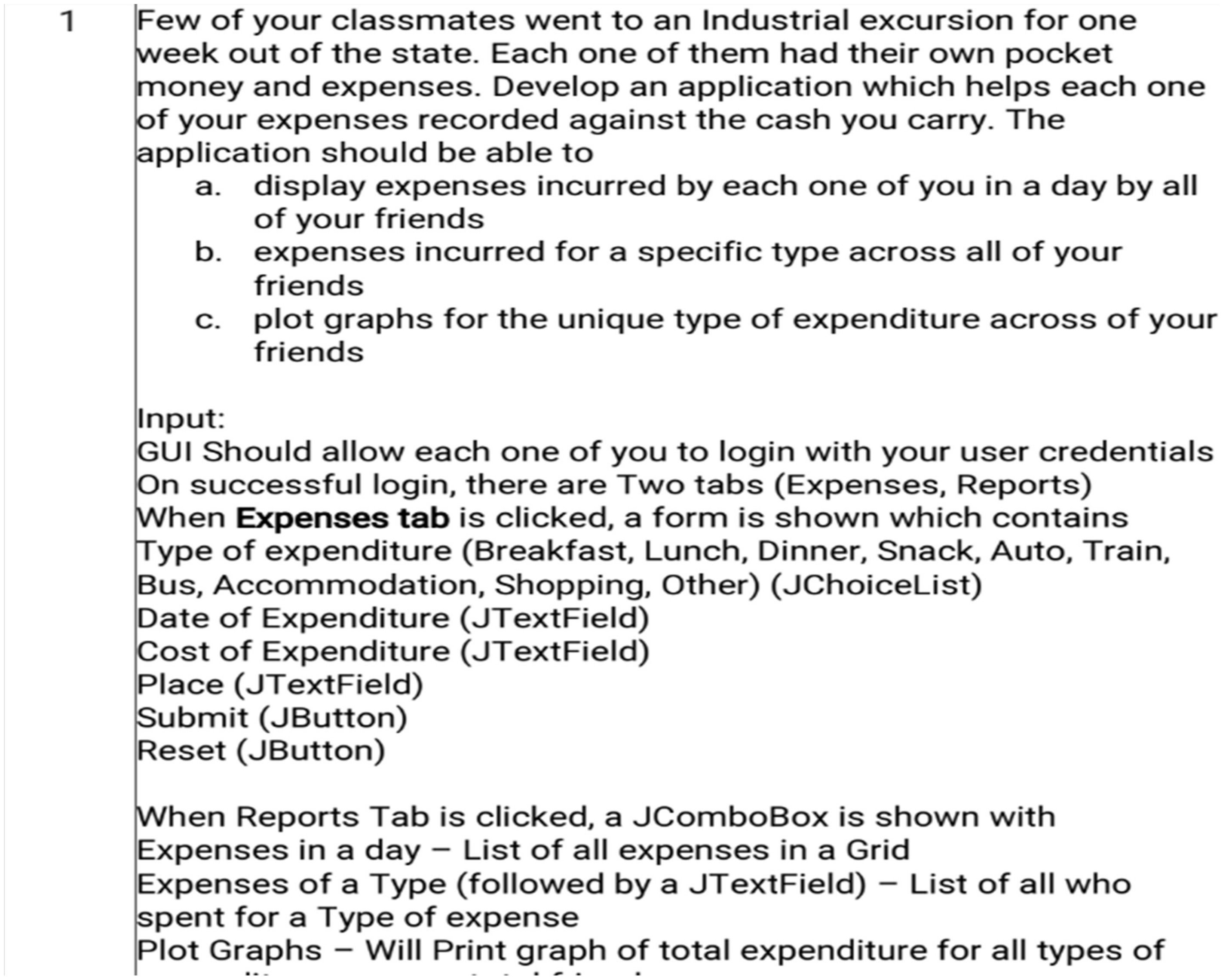
JAVA ASSIGNMENT-III

Name : S Sai suhrut.

Roll no. 1602-18-733-044

CSE-A



1 Few of your classmates went to an Industrial excursion for one week out of the state. Each one of them had their own pocket money and expenses. Develop an application which helps each one of your expenses recorded against the cash you carry. The application should be able to

a. display expenses incurred by each one of you in a day by all of your friends
b. expenses incurred for a specific type across all of your friends
c. plot graphs for the unique type of expenditure across of your friends

Input:
GUI Should allow each one of you to login with your user credentials
On successful login, there are Two tabs (Expenses, Reports)
When **Expenses tab** is clicked, a form is shown which contains
Type of expenditure (Breakfast, Lunch, Dinner, Snack, Auto, Train, Bus, Accommodation, Shopping, Other) (JChoiceList)
Date of Expenditure (JTextField)
Cost of Expenditure (JTextField)
Place (JTextField)
Submit (JButton)
Reset (JButton)

When Reports Tab is clicked, a JComboBox is shown with
Expenses in a day – List of all expenses in a Grid
Expenses of a Type (followed by a JTextField) – List of all who spent for a Type of expense
Plot Graphs – Will Print graph of total expenditure for all types of

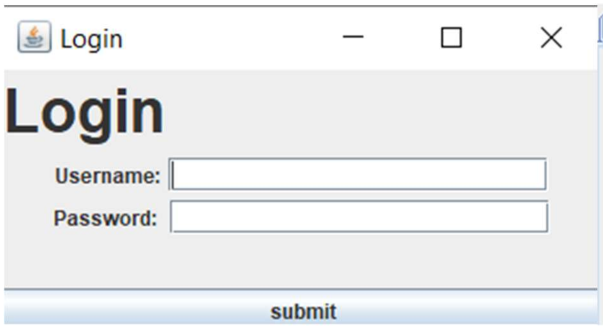
```java
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.regex.Pattern;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTabbedPane;
import javax.swing.JTable;
import javax.swing.JTextField;

/**
 * @author suhru
 *
 */
public class q11  implements ActionListener,ItemListener{
     JComboBox<String> type,reporttype;
     JFrame login,main;
     JButton submit=null,reset,submit1;
     JTextField user,dateofexpenditure,costofexpenditure,place;
     JPasswordField pass= new JPasswordField(20);
     String selectedtype=null;
     private HashMap <String,String>a=new HashMap<String,String>();
     private HashMap<String,ArrayList<ArrayList>> b=new
HashMap<String,ArrayList<ArrayList>>();
     public static void main(String args[]) {
          q11 q= new q11();
          q.initializer();
          q.loginframe();
     }
     public void loginframe() {
          login=new JFrame("Login");
          JLabel loginlabel=new JLabel("Login");
          login.setLayout(new BorderLayout());
          login.add(loginlabel,BorderLayout.NORTH);
          loginlabel.setFont(loginlabel.getFont().deriveFont (36.0f));
          JPanel details=new JPanel();
          details.add(new JLabel("Username:"));
          user=new JTextField(20);
          details.add(user);
          details.add(new JLabel("Password: "));
          details.add(pass);
          login.add(details);
          login.setSize(500,500);
          //        details.setBounds(0, 250, 250, 250);
          login.setVisible(true);
          submit= new JButton("submit");
          submit.addActionListener(this);
          login.add(submit,BorderLayout.SOUTH);
     }
```

```
    public void initializer() {
        a.put("admin","admin");
        a.put("suhrusai","suhrusai");
        String[] expenditure_types=
{"Breakfast","Lunch","Dinner","Snack","Auto","Train","Bus","Accommodat
ion","Shopping","Other"};
        type= new JComboBox<String>(expenditure_types);
        String[] report_type= {"Expenses in a day","Expenses of a
type","Plot Graphs"};
        this.reporttype=new JComboBox<String>(report_type);
    }
    public void mainframe() {
        main=new JFrame("Expenditure manager");
        main.setSize(500,500);
        main.setVisible(true);
        JTabbedPane tabbedpane = new JTabbedPane();
        JPanel expenses= new JPanel();
        JPanel reports= new JPanel();
        //Expenses panel
        expenses.add(new JLabel("Type of Expenditure: "));
        expenses.add(type);
        expenses.add(new JLabel("Date of Expenditure: "));
        dateofexpenditure=new JTextField(20);
        expenses.add(dateofexpenditure);
        costofexpenditure=new JTextField(20);
        expenses.add(new JLabel("Cost of Expenditure: "));
        expenses.add(costofexpenditure);
        expenses.add(new JLabel("              Place: "));
        place=new JTextField(20);
        expenses.add(place);
        submit1=new JButton("submit");
        submit1.addActionListener(this);
        reset= new JButton("Reset");
        expenses.add(submit1);
        expenses.add(reset);
        type.addItemListener(this);
        //EOF Expenses panel

        //Reports panel
        reports.add(reporttype);
        JPanel expday=new JPanel();
        JPanel exptype=new JPanel();
        JPanel graph=new JPanel();
```

```
		String names[]= {"Type of Expediture","Date of 
Expenditure"," Cost of Expenditure","Place"};
		ArrayList <String[]>temp=new ArrayList();
		for(Map.Entry<String,ArrayList<ArrayList>> i:b.entrySet()) {
			if(i.getKey().equals(user.getText()))
				temp.add((String[]) i.getValue().toArray());
		}
		JTable jt=new JTable();
		expday.add(jt);
		reporttype.addItemListener(new ItemListener(){

			@Override
			public void itemStateChanged(ItemEvent ie) {
				// TODO Auto-generated method stub
				if(ie.getStateChange()==1) {
					if(ie.getItem().toString().equals("Expenses in 
a day")) {
						expday.setVisible(true);
						exptype.setVisible(false);
						graph.setVisible(false);
					}
				}
			}
		});
		/*
		 * Expenses in a day
		 */
		expenses.add(expday);
		expenses.add(exptype);
		expenses.add(graph);
		tabbedpane.add("Hello",expenses);
		tabbedpane.add("Reports",reports);
		main.add(tabbedpane);
	}
	public boolean validator(String username,String password) {
		String Password=a.get(username);
		if(Password!=null && Password.equals(password)) {
			return true;
		}
		return false;
	}
	@SuppressWarnings("deprecation")
	@Override
```

```
     public void actionPerformed(ActionEvent ae) {
          // TODO Auto-generated method stub
          if(ae.getSource()==submit) {
               if(!validator(user.getText(), pass.~~getText~~())) {
                    JOptionPane.showMessageDialog(login, "Wrong
credentials", "Login error", JOptionPane.ERROR_MESSAGE);
                    return ;
               }
               this.login.setVisible(false);
               this.mainframe();
          }
          else if(ae.getSource()==submit1) {
               ArrayList <String>temp=new ArrayList<String>();
               if(selectedtype==null) {
                    JOptionPane.showMessageDialog(login, "Select and
option", "Invalid option", JOptionPane.ERROR_MESSAGE);
                    return;
               }
               temp.add(selectedtype);
               if(Pattern.matches("^(0[1-9]|[12][0-9]|3[01])[- /.](0[1-
9]|1[012])[- /.][1-9][1-9]", this.dateofexpenditure.getText())) {
                    temp.add(this.dateofexpenditure.getText());

               }
               else {
                    JOptionPane.showMessageDialog(login, "Invalid
Date", "Invalid Date", JOptionPane.ERROR_MESSAGE);
                    return;
               }
               if(Pattern.matches("[0-9]*",
this.costofexpenditure.getText())) {
                    temp.add(this.costofexpenditure.getText());
               }
               else {
                    JOptionPane.showMessageDialog(login, "Invalid
cost", "Invalid Cost", JOptionPane.ERROR_MESSAGE);
                    return;
               }
               if(this.place.getText().trim().contentEquals("")) {
                    JOptionPane.showMessageDialog(login, "Blank Field",
"Blank Field", JOptionPane.ERROR_MESSAGE);
                    return;
               }
```

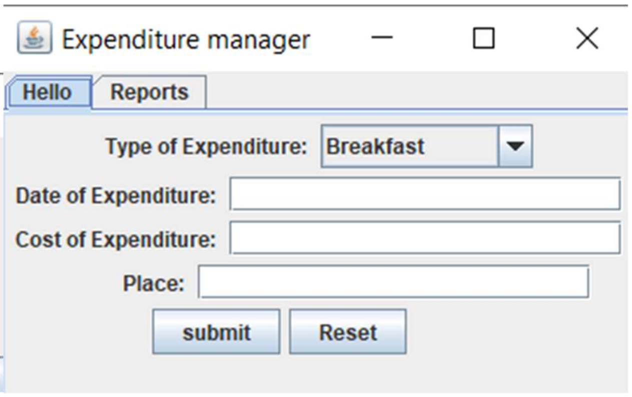
```
                else {
                    temp.add(this.place.getText());
                }
                if(b.get(user.getText())==null) {
                    ArrayList<ArrayList> temp1=new ArrayList();
                    temp1.add(temp);
                    b.put(user.getText(), temp1);
                }
                else {
                    ArrayList<ArrayList> temp1=new
ArrayList(b.get(user.getText()));
                    temp1.add(temp);
                    b.put(user.getText(), temp1);
                }

                /*
                 * Printing values;
                 */
                for(Map.Entry<String,ArrayList<ArrayList>>
i:b.entrySet()) {
                    System.out.println(i.getKey()+"
"+i.getValue());
                }
            }
        }
        @Override
        public void itemStateChanged(ItemEvent ie) {
            // TODO Auto-generated method stub
            if(ie.getStateChange()==1) {
                selectedtype=ie.getItem().toString();
            }
        }

}
```


Login

Username:

Password:

submit


Expenditure manager

Hello | Reports

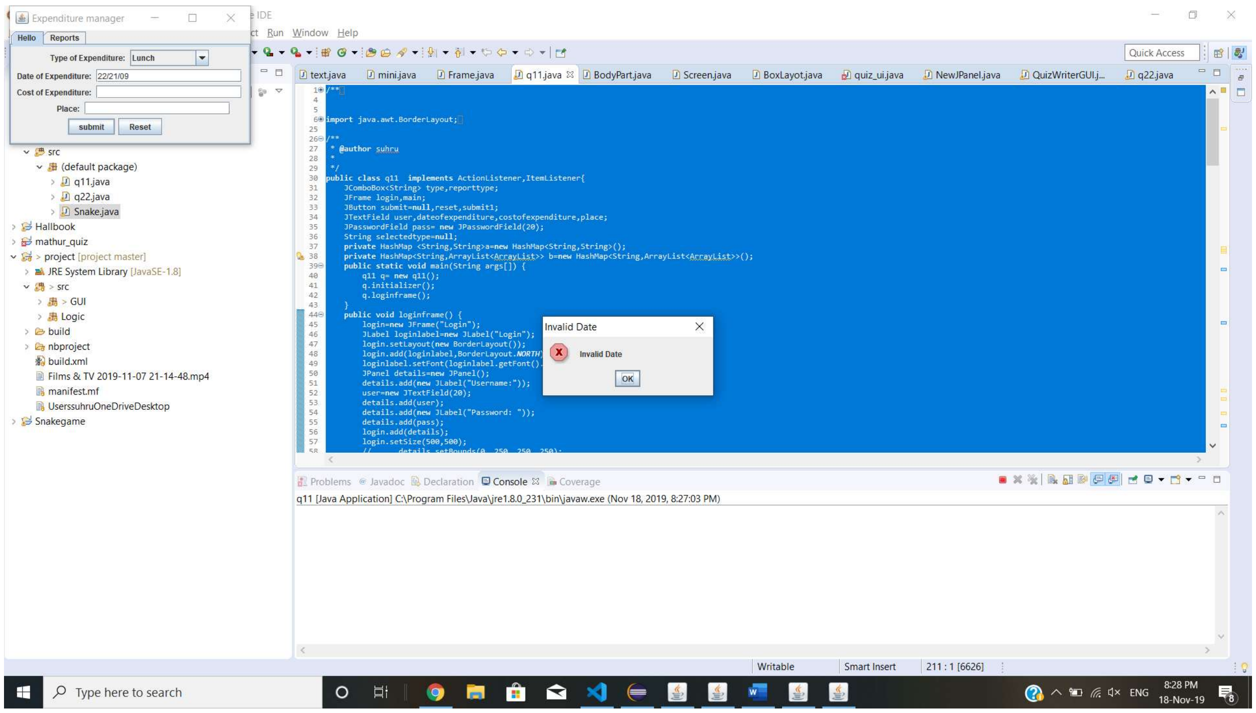
Type of Expenditure: Breakfast

Date of Expenditure:

Cost of Expenditure:

Place:

submit | Reset


Expenditure manager

Hello | Reports

Type of Expenditure: Lunch

Date of Expenditure: 22/21/09

Cost of Expenditure:

Place:

submit | Reset

```
/**

import java.awt.BorderLayout;

/**
 * @author suhru
 *
 */
public class q11  implements ActionListener,ItemListener{
    JComboBox<String> type,reporttype;
    JFrame login,main;
    JButton submit=null,reset,submit1;
    JTextField user,dateofexpenditure,costofexpenditure,place;
    JPasswordField pass= new JPasswordField(20);
    String selectedtype=null;
    private HashMap <String,String>a=new HashMap<String,String>();
    private HashMap<String,ArrayList<ArrayList>> b=new HashMap<String,ArrayList<ArrayList>>();
    public static void main(String args[]) {
        q11 q= new q11();
        q.initializer();
        q.loginframe();
    }
    public void loginframe() {
        login=new JFrame("Login");
        JLabel loginlabel=new JLabel("Login");
        login.setLayout(new BorderLayout());
        login.add(loginlabel,BorderLayout.NORTH
        loginlabel.setFont(loginlabel.getFont().
        JPanel details=new JPanel();
        details.add(new JLabel("Username:"));
        user=new JTextField(20);
        details.add(user);
        details.add(new JLabel("Password: "));
        details.add(pass);
        login.add(details);
        login.setSize(500,500);
    //      details.setBounds(0, 250, 250, 250);
```

Invalid Date

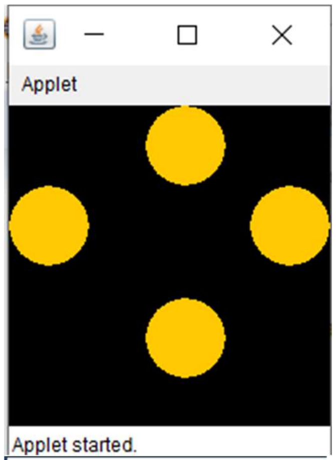
Invalid Date

OK

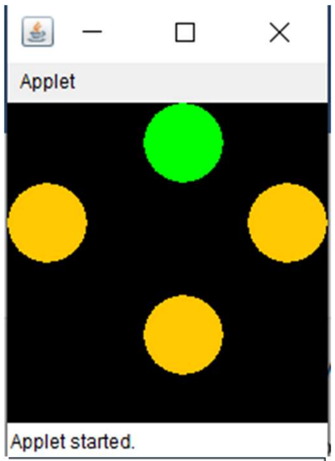
```
q11 [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (Nov 18, 2019, 8:27:03 PM)
```

2	Develop an animation to showcase the working of a traffic signal using JApplets and Graphics classes. Maintain individual timers for each directions Input: (Signal Cycle : West-North-East-South) Orange Time for all directions: 5 secs Green Time for Traffic coming from East: 10 secs Green Time for Traffic coming from West: 10 secs Green Time for Traffic coming from North: 15 secs Green Time for Traffic coming from South: 15 secs

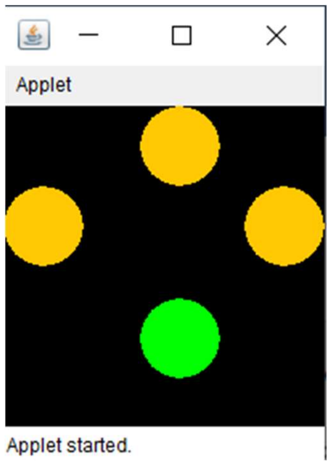
```java
import java.awt.Color;
import java.awt.Graphics;
import java.util.Timer;
import java.util.TimerTask;

import javax.swing.JApplet;
public class q22 extends JApplet implements Runnable{
    long initialtime;
    int time1=0,time2=0,time3=0,time4=0;
    public void init() {
        // TODO Auto-generated method stub
        initialtime=System.currentTimeMillis();

    }
    public void start() {
        Timer timer1 = new Timer();
        timer1.scheduleAtFixedRate(new TimerTask() {
            public void run() {
                repaint();
                time1++;
            }
        },000,1000);
        Timer timer2 = new Timer();
        timer2.scheduleAtFixedRate(new TimerTask() {
            public void run() {
                repaint();
                time2++;
            }
        },000,1000);
        Timer timer3 = new Timer();
        timer3.scheduleAtFixedRate(new TimerTask() {
            public void run() {
                repaint();
                time3++;
            }
        },00,1000);
        Timer timer4 = new Timer();
        timer4.scheduleAtFixedRate(new TimerTask() {
            public void run() {
                repaint();
                time4++;
            }
        },0,1000);
    }
    public void paint(Graphics g) {
        //g.fillOval(arg0, arg1, arg2, arg3);
        getContentPane().setBackground(Color.black);
        if(time1%55<=5) {
            g.setColor(Color.ORANGE);
            g.fillOval(85,0, 50, 50);
            g.fillOval(0, 50, 50, 50);
            g.fillOval(150, 50, 50, 50);
            g.fillOval(85,120, 50, 50);
        }
        else if(time2%55<=15) {
            g.setColor(Color.ORANGE);
            g.fillOval(85,0, 50, 50);
            g.fillOval(0, 50, 50, 50);
            g.setColor(Color.GREEN);
            g.fillOval(150, 50, 50, 50);
            g.setColor(Color.ORANGE);
            g.fillOval(85,120, 50, 50);
        }
        else if(time3%55<=25) {
            g.setColor(Color.ORANGE);
            g.fillOval(85,0, 50, 50);
            g.setColor(Color.GREEN);
            g.fillOval(0, 50, 50, 50);
            g.setColor(Color.ORANGE);
            g.fillOval(150, 50, 50, 50);
            g.fillOval(85,120, 50, 50);
        }
        else if(time4%55<=40) {
            g.setColor(Color.GREEN);
            g.fillOval(85,0, 50, 50);
            g.setColor(Color.ORANGE);
            g.fillOval(0, 50, 50, 50);
            g.fillOval(150, 50, 50, 50);
            g.fillOval(85,120, 50, 50);
        }
        else if(time2%55<=55) {
            g.setColor(Color.ORANGE);
            g.fillOval(85,0, 50, 50);
            g.fillOval(0, 50, 50, 50);
            g.fillOval(150, 50, 50, 50);
            g.setColor(Color.GREEN);
            g.fillOval(85,120, 50, 50);
        }

    }
    public void run() {
        while(true) {

        }
    }

}
```



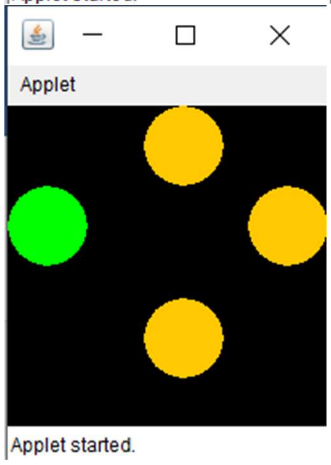
Applet started.



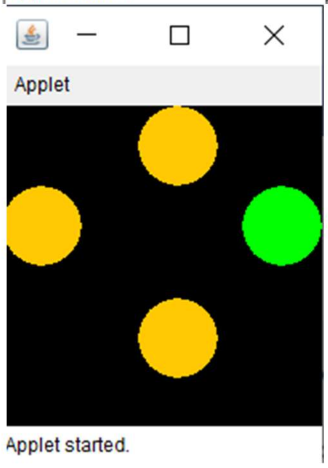
Applet started.



Applet started.



Applet started.



Applet started.

3	Develop a Snake game in JFrame, which has only one level and has the following properties 1. A cube is made visible at a random time slot in the JFrame dimension 2. Snake starts to grow when it hits the cube 3. Game ends when the snake hits the walls. 4. Snake moves with the keys 5. Snake speed is constant

Apple.java

```
package game;
```

```java
import java.awt.Color;
import java.awt.Graphics;

public class Apple {

    private int x, y, width, height;

    public Apple(int x, int y, int tileSize) {
        this.x = x;
        this.y = y;
        width = tileSize;
        height = tileSize;
    }
    public void tick() {

    }
    public void draw(Graphics g) {
        g.setColor(Color.GRAY);
        g.fillRect(x * width , y * height, width, height);
    }

    public int getx() {
        return x;
    }
    public void setx(int x) {
        this.x = x;
    }
    public int gety() {
        return y;
    }
    public void sety(int y) {
        this.y = y;
    }

}
```

BodyPart.java

```java
package game;

import java.awt.Color;
import java.awt.FlowLayout;
import javax.swing.BoxLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class Frame {
   JLabel score =new JLabel("Score: "+0);
    public Frame() {

        JFrame frame = new JFrame();
        Screen screen = new Screen();
        frame.setLayout(new FlowLayout());
        frame.add(score);
        frame.add(screen);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setTitle("Snake");
        frame.setResizable(false);
        frame.pack();
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
        Thread t=new Thread() {
         public void run() {
              while(true)
              score.setText("Score: "+
String.valueOf(screen.score()));
         }
        };
        t.start();

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

Frame.java

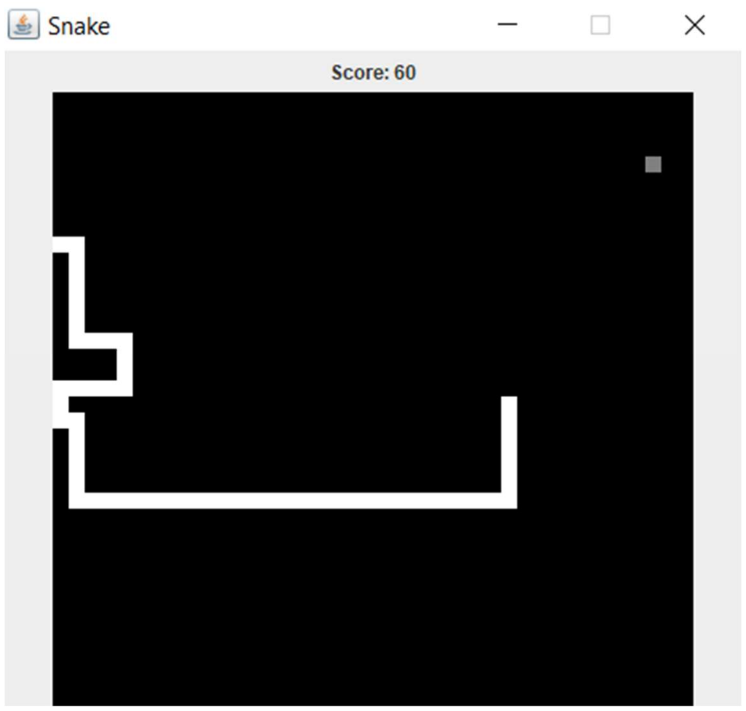
```
package game;

import java.awt.Color;
import java.awt.FlowLayout;

import javax.swing.BoxLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;
```

```
public class Frame {
   JLabel score =new JLabel("Score: "+0);
    public Frame() {

        JFrame frame = new JFrame();
        Screen screen = new Screen();
        frame.setLayout(new FlowLayout());
        frame.add(score);
        frame.add(screen);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setTitle("Snake");
        frame.setResizable(false);
        frame.pack();
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
        Thread t=new Thread() {
         public void run() {
              while(true)
              score.setText("Score: "+
String.valueOf(screen.score()));
         }
        };
        t.start();

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

__Screen.java__

```
package game;


import java.awt.Color;

import java.awt.Dimension;

import java.awt.Graphics;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.util.ArrayList;
import java.util.Random;

import javax.swing.JPanel;

public class Screen extends JPanel implements Runnable, KeyListener {

   private static final long serialVersionUID = 1L;

   public static final int WIDTH = 400, HEIGHT = 400;

   private Thread t;
   private boolean running = false;

   private BodyPart b;
   private ArrayList<BodyPart> snake;

   private Apple apple;
   private ArrayList<Apple> apples;

   private Random r;

   private int x = 10, y = 10;
   private int size = 5;

   private boolean right = true, left = false, up = false, down =false;
   public Screen() {
      setFocusable(true);

      addKeyListener(this);
```

```
    setPreferredSize(new Dimension(WIDTH, HEIGHT));

    r = new Random();

    snake = new ArrayList<BodyPart>();
    apples = new ArrayList<Apple>();

    start();
}
public int score() {
      return size;
}
public void move() {
    if (snake.size() == 0) {
        b = new BodyPart(x, y, 10);
        snake.add(b);
    }
    if(apples.size() == 0) {
        int x = r.nextInt(40);
        int y = r.nextInt(10);

        apple = new Apple(x, y, 10);
        apples.add(apple);
    }

    for(int i = 0; i < apples.size(); i++) {
        if(x == apples.get(i).getx() &&
                y == apples.get(i).gety()) {
            size++;
            apples.remove(i);
            i++;
        }
    }

    for(int i =0; i < snake.size(); i++) {
        if(x == snake.get(i).getx() &&
                y == snake.get(i).gety()) {
            if(i != snake.size() - 1) {
                stop();
            }
        }
    }
    if(x < 0 || x > 39 || y < 0 || y > 39) {
        stop();
    }
   try {
                Thread.sleep(100);
        } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        if(right) x++;
        if(left) x--;
        if(up) y--;
        if(down) y++;

        b = new BodyPart(x, y, 10);
        snake.add(b);
        if(snake.size() > size) {
            snake.remove(0);
        }
    }

    public void paint(Graphics g) {
    g.clearRect(0, 0, WIDTH, HEIGHT);
    g.setColor(Color.BLACK);
    g.fillRect(0, 0, WIDTH, HEIGHT);
    g.setColor(Color.WHITE);
    g.setColor(Color.WHITE);
    g.drawString("Score : "+String.valueOf(snake.size()),0,0);
     for (int i = 0; i < snake.size(); i++) {
        snake.get(i).draw(g);
    }
    for(int i = 0; i < apples.size(); i++) {
        apples.get(i).draw(g);
    }
    }

    public void start() {
    running = true;
    t = new Thread(this);
    t.start();
    }

    public void stop() {
    running = false;
    try {
        t.join();
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
   setBackground(Color.WHITE);
    }

    public void run() {
    while (running) {
        move();
        repaint();
    }
    }

    @Override
    public void keyPressed(KeyEvent e) {
    int key = e.getKeyCode();
    if(key == KeyEvent.VK_RIGHT && !left) {
        up = false;
        down = false;
        right = true;
    }
    if(key == KeyEvent.VK_LEFT && !right) {
        up = false;
        down = false;
        left = true;
    }
    if(key == KeyEvent.VK_UP && !down) {
        left = false;
        right = false;
        up = true;
    }
    if(key == KeyEvent.VK_DOWN && !up) {
        left = false;
        right = false;
        down = true;
    }
    }
    @Override
    public void keyReleased(KeyEvent arg0) {
    }
    public void keyTyped(KeyEvent arg0) {
    }
}
```


Snake
Score: 60