Rnsrajapakshe 27058

Q1.

```
Public class Assignment

{
   public static void main(String[] args)

      {
         String name = “namal rajapakshe”;
         System.out.println(name);

      }

   }
```

Q2.

```
 import java.util.Scanner;
public class Assignment
{



   public static void main(String[] args)
   {
      Scanner rn=new Scanner (System.in);
      int num1,num2,num3,sum;

      System.out.println("enter the first number");
      num1=sn.nextInt();

      System.out.println("enter the secound number");
      num2=sn.nextInt();

     System.out.println("enter the third number");
     num3=sn.nextInt();

     sum= num1+num2+num3;

      System.out.println(" value  "+ sum);
   }

}
```

Q3.

```
 import java.util.Scanner;
   public class Assignment
{
```

```
        public static void main(String[] args)
        {
           Scanner rn=new Scanner (System.in);
           double  Fahrenheit;

           System.out.println("enter Fahrenheit value ");
           Fahrenheit=sn.nextDouble();

           double celsius =  (5.0 / 9.0) *(Fahrenheit - 32);

           System.out.println("your celsius value is"+celsius);
        }

      }
```

Q4.

```
    import java.util.Scanner;
       public class Assignment
       {



        public static void main(String[] args)
        {
           Scanner sn=new Scanner (System.in);
           int num1, num2, num3, sum, product, smallest, largest;
           double average;

            System.out.print("Enter the first number: ");
            num1 = rn.nextInt();

           System.out.print("Enter the second number: ");
           num2 = rn.nextInt();

           System.out.print("Enter the third number: ");
           num3 = rn.nextInt();

           // Calculate the sum
            sum = num1 + num2 + num3;
           System.out.println("Sum: " + sum);


            average = sum / 3.0;
           System.out.println("Average: " + average);


            product = num1 * num2 * num3;
```

```
        System.out.println("Product: " + product);

         smallest = num1;
        if (num2 < smallest)
        {
           smallest = num2;
        }
        if (num3 < smallest)
              {
           smallest = num3;
        }
        System.out.println("Smallest: " + smallest);


        largest = num1;
        if (num2 > largest) {
           largest = num2;
            }
        if (num3 > largest) {
           largest = num3;
        }
        System.out.println("Largest: " + largest);


    }
}
```

Q5.

```
package assignment;
import java.text.DecimalFormat;
import java.util.Scanner;

public class Assignment
{
   public static void main(String[] args)
   {
       int[] grades = new int[20];
      int count = 0;
      int grade;
      Scanner rn=new Scanner (System.in);
      System.out.println("enter up to 20 integer value grades");

      while (count<grades.length)
      {
          grade = rn.nextInt();
```

```
            if (grade==-1)
            {
               break;
            }
            grades[count]=grade;
            count++;
         }

        double average = calculateAvg(grades, count);


         DecimalFormat decimalFormat = new DecimalFormat("0.00");
         String formattedAverage = decimalFormat.format(average);

         System.out.println("Average of grades: " + formattedAverage);
      }
        private static double calculateAvg(int[] grades, int count) {
         if (count == 0) {
            return 0;
         }

         int sum = 0;
         for (int x = 0; x < count; x++) {
            sum += grades[x];
         }
         return (double) sum / count;
   }
```

Q6.

```
      public class DataSet
   {


      public static void main(String[] args)
      {
         Date date = new Date (15.02.2022);

         date.displayDate();

         date.setDay(15);
         date.setMonth(2);
         date.setYear(2022);

         date.displayDate();
      }

   }
```

Date class:

```
public class Date
{
 private int day;
 private int month;
 private int year;


 public Date ( int day,int month, int year)
 {
    this.day=day;
    this.month = month;
    this.year = year;

 }



 public int getDay()
    {
      return day;
    }
 public void setDay(int day)
    {
      this.day = day;
    }


  public int getMonth()
    {
      return month;
    }

   public void setMonth(int month)
    {
      this.month = month;
    }


   public int getYear()
    {
      return year;
    }

   public void setYear(int year)
   {
     this.year = year;
```

```
    }


    public void displayDate()
    {
      System.out.println(day + "." + month + "." + year);
    }
  }
```

Q7.

```
  public class item
{
  String description;
    int location;

  public item (int location,String description)
  {
    this.location=location;
    this.description=description;
  }

  // getter and setter
  public int getlocation()
  {
    return location;
  }



  public void setlocation(int location)
  {
    this.location= location;
  }

  public String getdescription()
  {
     return description;
  }

  public void setdescription( String description)
  {
   this.description= description;
  }


}
```

```
public  class Monster extends item
{
   public Monster (int location,String description)
   {
      super(location,description);
   }

}
```

class

```
{
   public static void main(String[] args)
   {
     item rn=new item(1,"this is item");
      System.out.println("item location "+ rn.getlocation());

       System.out.println("Item description: " + rn.getdescription());


      rn.setlocation(2);
      rn.setdescription("Updated description.");
      System.out.println("Updated item location is: " + rn.getlocation());
      System.out.println("Updated item description is: " + rn.getdescription());

      Monster monster = new Monster(3, "This is  monster.");
      System.out.println("Monster location: " + monster.getlocation());
      System.out.println("Monster description: " + monster.getdescription());


      monster.setlocation(4);
      monster.setdescription("Updated monster description.");
      System.out.println("Updated monster location is: " + monster.getlocation());
      System.out.println("Updated monster description is: " + monster.getdescription());


   }

}
```

Q8.

1.
```
public class SavingsAcc
{

   private static double annualInterestRate;
```

```
    private double sBalance;

    public SavingsAcc(double sBalance)
  {
        this.sBalance = sBalance;
    }

    public void calMInterest() {
       double mInterest = sBalance * annualInterestRate / 12;
       sBalance += mInterest;
    }

    public static void modifyInterestRate(double newinterestRate) {
       annualInterestRate = newinterestRate;
    }

    public double getSBalance() {
       return sBalance;
    }
}
```

- public clas TestSAccount
  {

```
    public static void main(String[] args) {

   SAccount  sn1 = new SAccount(2000.00);
   SAccount sn2 = new SAccount(3000.00);


    SAccount.modifyInterestRate(0.04);


     sn1.calculateMInterest();
      sn2.calculateMInterest();

   System.out.println("Month 1 - Balances  is (4% interest rate):");
   System.out.println("Saver1 balance is : $" + sn1.getSBalance());
   System.out.println("Saver2 balance is : $" + sn2.getSBalance());


   SAccount.modifyInterestRate(0.05);
```

```
        sn1.calculateMInterest();
        sn2.calculateMInterest();

        System.out.println("Month 2 - Balances is (5% interest rate):");
        System.out.println("Saver1 balance is : $" + sn1.getSBalance());
        System.out.println("Saver2 balance is : $" + sn2.getSBalance());
    }
}
```

Q9.

```
public class Q9
{


    public static void main(String[] args)
    {
        sedan scs = new sedan();
        scs.speed = 200;
        scs.regularPrice = 20000;
        scs.color = "Red";
        scs.length = 25;

        ford scf1 = new ford();
        scf1.speed = 180;
        scf1.regularPrice = 30000;
        scf1.color = "Blue";
        scf1.year = 2019;
        scf1.manufacturerDiscount = 2000;


         ford scf2 = new ford();
        scf2.speed = 220;
        scf2.regularPrice = 40000;
        scf2.color = "Silver";
        scf2.year = 2021;
        scf2.manufacturerDiscount = 3000;

        car scc = new car();
        scc.speed = 150;
        scc.regularPrice = 25000;
        scc.color = "Black";
```

```
        System.out.println("Sale Price of Sedan: $" + scs.getSlPrice());
        System.out.println("Sale Price of Ford 1: $" + scf1.getSlPrice());
        System.out.println("Sale Price of Ford 2: $" + scf2.getSlPrice());
        System.out.println("Sale Price of Car: $" + scc.getSlPrice());
    }

}
```

- ```
  public class car
  {
    int speed;
    double regularPrice;
    String color;

    public double getSlPrice()
    {
      return regularPrice;
    }
  }
  ```

- ```
  public class ford extends car
    int year;
    int manufacturerDiscount;

    @Override
    public double getSlPrice() {
      return super.getSlPrice() - manufacturerDiscount;
    }

  }
  ```

- ```
  public class sedan extends car
  {
    int length;

    @Override
    public double getSlPrice()
    {
      if (length > 20) {
        return regularPrice * 0.95;
      } else {
        return regularPrice * 0.9;
      }
    }
  }
  ```
  ```

- public class truck extends car
  {
  int weight;

  @Override
  public double getSlPrice() {
  if (weight > 2000) {
  return regularPrice * 0.9;
  } else {
  return regularPrice * 0.8;
  }
  }
  }

Q10.

```
public class Q10
{
   public static void main(String[] args)
    {
        shape scc = new circle();
       scc.draw();
       scc.erase();

       shape sct = new triangle();
       sct.draw();
       sct.erase();

       shape scsq = new square();
       scsq.draw();
       scsq.erase();
    }


}
```

- public class circle extends shape
  {
  @Override
  public void draw()
  {
  System.out.println("Drawing a circle");
  }

  @Override
  public void erase()
  {
  System.out.println("Erasing a circle");
  }

```
}
```

- ```
  public class shape
  {
     public void draw()
     {
        System.out.println("Drawing a shape");
     }

     public void erase()
     {
        System.out.println("Erasing a shape");
     }
  }
  ```

- ```
  public class square  extends shape
  {
     @Override
     public void draw()
     {
        System.out.println("Drawing a square");
     }

     @Override
     public void erase()
     {
        System.out.println("Erasing a square");
     }
  }
  ```

- ```
  public class triangle extends shape
  {
     @Override
     public void draw()
     {
        System.out.println("Drawing a triangle");
     }

     @Override
     public void erase()
     {
        System.out.println("Erasing a triangle");
     }
  }
  ```
  ```

Q11.

```
Interface A
 {
   void method 1();
   void method 2();
}

Class my implement A
{
 @override
  Public void method1()
{
  System.out.println(“method()1 implementation in my”);
}
@override
   Public void method2()
{
  System.out.println(“method() implementation in my”);
}
}
```

- *Multiple inheritance*

```
  Interface x
{
   void methodX();
}

   Interface y
{
   void methodY();
}

class my implement X,Y
{
@override
Public void methodX
{
   System.out.println(“method() implementation in my”);
}
@override
Public void methodX
```

```
{
  System.out.println("method() implementation in my");
}

}
```

- *Interface "test"*

```
Interface test
{
 Int square(int nu);
}

class arithmetic implements test
{
@overrride
Public int square(int nu)
{
Return nu*nu;
}

}
```

- *ToTestint class*

```
class ToTestint
{
public  static void main(String [] args)
{
 Arithmetic arithobj = new Arithmetic();
 int no = 6 ;
int result = arithobj.square(nu);
system.out.println("square of " +nu+" is"+result);
}
}
```

2. Q12.

```
Public class negative
{
Public static void main (String [] args)
{
Int [] arr;
try {
int size = -5;
if (size < 0)
```

```
{
Throw new NegativeArratSizeException(“array size cannot be negative “);
}
Arr=new int [size];
System.out.println(“array is created “);
}
Catch (NegativeArratSizeException ex){
System.out.println(“Exception caught:”+ ex.getMessage();
}
}
}
```

- *Multiple catch statements*

```
public class MultipleCatch {
   public static void main(String[] args) {
      try {
         int[] arr = new int[8];
         System.out.println(arr[16]);
      } catch (ArrayIndexOutOfBoundsException ex) {
         System.out.println("Array Index Out of Bounds Exception is: " +
exp.getMessage());
      } catch (ArithmeticException ex) {
         System.out.println("Arithmetic Exception is: " + ex.getMessage());
      } catch (Exception ex) {
         System.out.println("Generic Exception is: " + ex.getMessage());
      }
   }
```

- *Subclass exception precedence over base class*

```
public class BaseException extends Exception {
   public BaseException(String message) {
      super(message);
   }
}

public class SubException extends BaseException {
   public SubException(String message) {
      super(message);
   }
}

public class SubClassException{
   public static void main(String[] args) {
      try {
```

```
            throw new SubException("Subclass  occurred.");
        } catch (SubException ex) {
            System.out.println("Caught SubException: " + ex.getMessage());
        } catch (BaseException ex) {
            System.out.println("Caught BaseException: " + ex.getMessage());
        }
    }
}
```

- *try/catch with finally clause:*

```
public class FinallyEx {
    public static void main(String[] args) {
        try {
            int result = 16 / 0; // Attempt  divide by zero
            System.out.println("Result is: " + result);
        } catch (ArithmeticException ex) {
            System.out.println("Arithmetic Exception is : " + ex.getMessage());
        } finally {
            System.out.println("Finally block executed.");
        }
    }
}
```

- *usage of the throws clause:*

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class ThrowsEx {
    public static void main(String[] args) {
        try {
            int result = readNumber();
            System.out.println("Result is: " + result);
        } catch (NumberFormatException ex) {
            System.out.println("Number Format Exception is : " + ex.getMessage());
        } catch (IOException ex) {
            System.out.println("IOException is : " + ex.getMessage());
        }
    }

    public static int readNumber() throws IOException, NumberFormatException {
        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
        System.out.print("Enter the number: ");
        String input = reader.readLine();
```

```
            return Integer.parseInt(input);
        }
    }
```

- *User defined exception:*

```
public class CustomException extends Exception {
    public CustomException(String message) {
        super(message);
    }
}

public class UDException {
    public static void main(String[] args) {
        try {
            int age = 13;
            if (age < 18) {
                throw new CustomException("You must be  at least 18 years old.");
            }
            System.out.println(" You are eligible.");
        } catch (CustomException ex) {
            System.out.println("Custom Exception: " + ex.getMessage());
        }
    }
}
```

Q13.

```
public class MyRunnable implements Runnable {
    @Override
    public void run() {
        try {
            System.out.println("Child Thread  starting...");
            Thread.sleep(500);
            System.out.println("Child Thread  ending...");
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
    }

    public static void main(String[] args) {

        MyRunnable myRunnable = new MyRunnable();


        Thread thread = new Thread(myRunnable);
```

```
    thread.start();

    System.out.println("Main Thread is executing concurrently...");
  }
}
```

- .

```
public class MyThread extends Thread {
  public MyThread() {
    super();
    start();
  }

  @Override
  public void run() {
    try {
      System.out.println("Child Thread is starting...");
      Thread.sleep(500);
      System.out.println("Child Thread is ending...");
    } catch (InterruptedException ex) {
      ex.printStackTrace();
    }
  }

  public static void main(String[] args) {

    System.out.println("Main Thread is executing...");


  }
}
```

Q14.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class BasicCalculator extends JFrame implements ActionListener {
  private JTextField displayField;
  private String currentInput;
  private double currentValue;
  private char lastOperator;

  public BasicCalculator() {
    currentInput = "";
```

```
    currentValue = 0;
    lastOperator = ' ';

    // Create GUI components
    displayField = new JTextField(15);
    displayField.setEditable(false);
    displayField.setHorizontalAlignment(JTextField.RIGHT);

    JButton[] numberButtons = new JButton[10];
    for (int i = 0; i < 10; i++) {
       numberButtons[i] = new JButton(String.valueOf(i));
       numberButtons[i].addActionListener(this);
    }

    JButton addButton = new JButton("+");
    JButton subtractButton = new JButton("-");
    JButton multiplyButton = new JButton("*");
    JButton divideButton = new JButton("/");
    JButton equalsButton = new JButton("=");
    JButton clearButton = new JButton("C");

    addButton.addActionListener(this);
    subtractButton.addActionListener(this);
    multiplyButton.addActionListener(this);
    divideButton.addActionListener(this);
    equalsButton.addActionListener(this);
    clearButton.addActionListener(this);

    // Create the panel and set the layout
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(4, 4, 10, 10));

    // Add components to the panel
    panel.add(numberButtons[1]);
    panel.add(numberButtons[2]);
    panel.add(numberButtons[3]);
    panel.add(addButton);
    panel.add(numberButtons[4]);
    panel.add(numberButtons[5]);
    panel.add(numberButtons[6]);
    panel.add(subtractButton);
    panel.add(numberButtons[7]);
    panel.add(numberButtons[8]);
    panel.add(numberButtons[9]);
    panel.add(multiplyButton);
    panel.add(clearButton);
    panel.add(numberButtons[0]);
    panel.add(equalsButton);
```

```
        panel.add(divideButton);

        // Add the components to the frame
        this.add(displayField, BorderLayout.NORTH);
        this.add(panel, BorderLayout.CENTER);

        // Set frame properties
        this.setTitle("Basic Calculator");
        this.setSize(300, 300);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        String actionCommand = e.getActionCommand();
        char inputChar = actionCommand.charAt(0);

        if (Character.isDigit(inputChar)) {
            currentInput += inputChar;
            displayField.setText(currentInput);
        } else if (inputChar == 'C') {
            currentInput = "";
            currentValue = 0;
            lastOperator = ' ';
            displayField.setText("");
        } else if (inputChar == '=') {
            calculateResult();
        } else {
            if (!currentInput.isEmpty()) {
                calculateResult();
                lastOperator = inputChar;
            }
        }
    }

    private void calculateResult() {
        double inputNumber = Double.parseDouble(currentInput);

        switch (lastOperator) {
            case '+':
                currentValue += inputNumber;
                break;
            case '-':
                currentValue -= inputNumber;
                break;
            case '*':
                currentValue *= inputNumber;
                break;
```

```
            case '/':
                if (inputNumber != 0) {
                    currentValue /= inputNumber;
                } else {
                    displayField.setText("Error: Cannot divide by zero.");
                    currentInput = "";
                    return;
                }
                break;
            default:
                currentValue = inputNumber;
        }

        displayField.setText(String.valueOf(currentValue));
        currentInput = "";
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new BasicCalculator());
    }
}
```