Tutorial No. 3

Q.1 extend basic calculator with scientific calculations

   Contains following operations

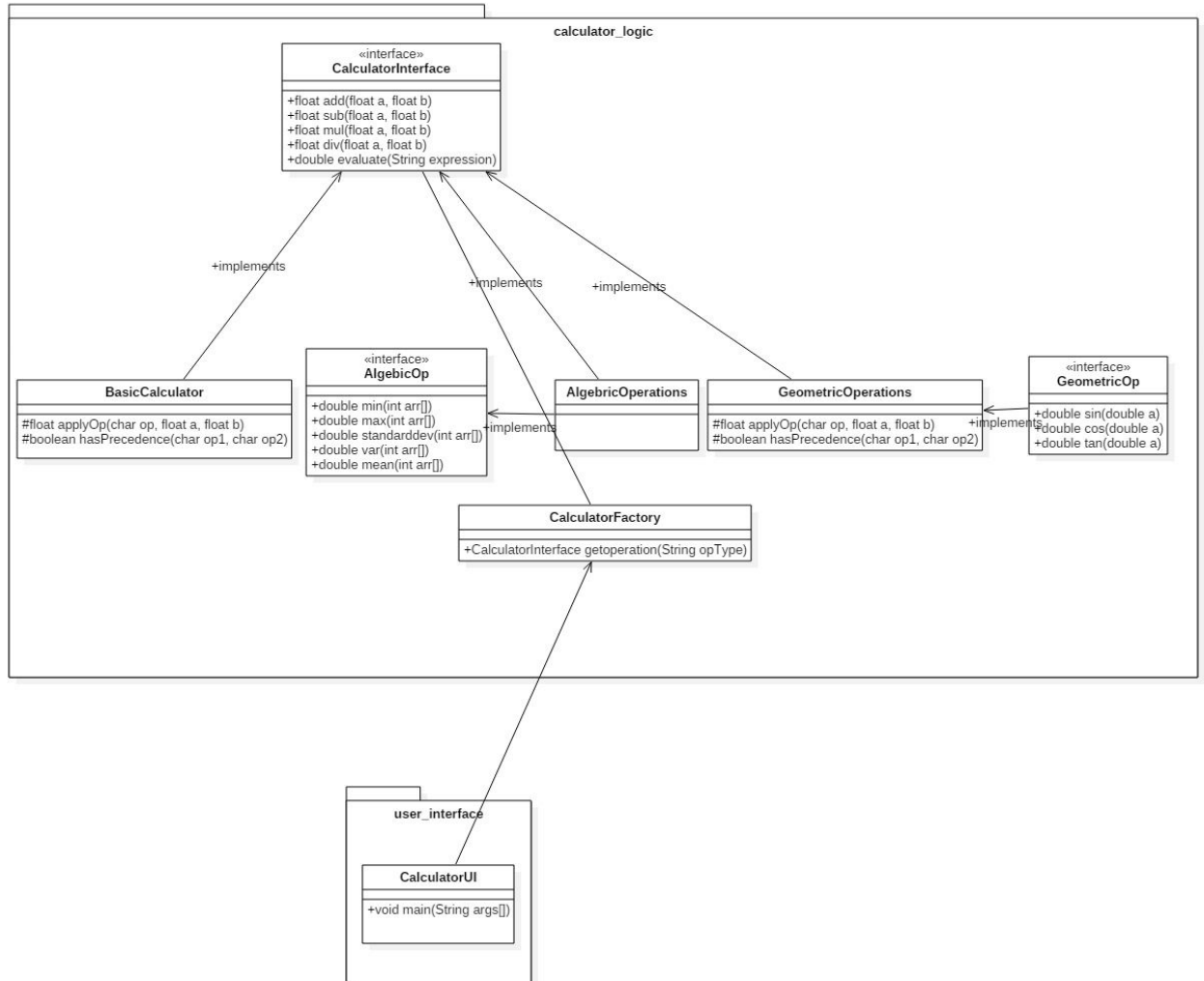Geometric :

  Sin ,cos,tan

Statistics:

 Min,max,mean,standard deviation,variance

Class diagram:

calculator_logic

«interface»
CalculatorInterface

+float add(float a, float b)
+float sub(float a, float b)
+float mul(float a, float b)
+float div(float a, float b)
+double evaluate(String expression)

+implements   +implements   +implements

BasicCalculator

#float applyOp(char op, float a, float b)
#boolean hasPrecedence(char op1, char op2)

«interface»
AlgebicOp

+double min(int arr[])
+double max(int arr[])
+double standarddev(int arr[])
+double var(int arr[])
+double mean(int arr[])

+implements

AlgebricOperations

GeometricOperations

#float applyOp(char op, float a, float b)
#boolean hasPrecedence(char op1, char op2)

+implements

«interface»
GeometricOp

+double sin(double a)
+double cos(double a)
+double tan(double a)

CalculatorFactory

+CalculatorInterface getoperation(String opType)

user_interface

CalculatorUI

+void main(String args[])

Implementation
//user interfaces
package user_interface;
import calculator_logic.*;
import java.util.Scanner;
public class CalculatorUI {
        public static void main(String[] args) {
            Scanner scan=new Scanner(System.in);
                String c="Y";
                while(c.equals("Y"))
                {
                System.out.println("Enter the expression : ");

```java
                String exp;
                exp=scan.nextLine();
                CalculatorFactory calFactory=new CalculatorFactory();
                CalculatorInterface cal=calFactory.getOperation(exp);
                double res=cal.evaluate(exp);
                System.out.println("Answer is "+res);
                System.out.print("Do You not continue(Y/N):");
                c=scan.nextLine();
                }
                scan.close();
        }
}


//interfaces
package calculator_logic;

public interface CalculatorInterface {
        public float add(float a,float b);
        public float sub(float a,float b);
        public float mult(float a,float b);
        public float div(float a,float b);
        public double evaluate(String expression);
}

public interface GeometricOp {
        public double sin(double a);
        public double cos(double a);
        public double tan(double a);

}
public interface AlgebricOp {
                public  double min(int arr[]);
                public  double max(int arr[]);
                public  double standarddev(int arr[]);
                public  double var(int arr[]);
                public  double mean(int arr[]);

}

//Concrete  implementation
//arithmetic
package calculator_logic;
```

```java
import java.util.*;
public class AlgebricOperations implements AlgebricOp,CalculatorInterface {

	@Override
	public double min(int[] arr) {
		// TODO Auto-generated method stub
		int min=arr[0];
		for(int k=1;k<arr.length;k++)
		{
			if(arr[k]<min)
				min=arr[k];
		}
		return min;
	}

	@Override
	public double max(int[] arr) {
		// TODO Auto-generated method stub
		int max=arr[0];
		for(int k=1;k<arr.length;k++)
		{
			if(arr[k]>max)
				max=arr[k];
		}
		return max;
	}

	@Override
	public double standarddev(int[] arr) {
		// TODO Auto-generated method stub
		double sum = 0.0, standardDeviation = 0.0;

		for(double num : arr) {
			sum += num;
		}

		double mean = sum/10;

		for(double num: arr) {
			standardDeviation += Math.pow(num - mean, 2);
		}
		return Math.sqrt(standardDeviation/10);
	}
```

```java
@Override
public double var(int[] arr) {
        // TODO Auto-generated method stub
        double sd=standarddev(arr);
        return Math.pow(sd,2);
}


@Override
public double mean(int[] arr) {
        // TODO Auto-generated method stub
        double sum=0.0;
        for(double num : arr) {
    sum += num;
  }
        int length=arr.length;
        double mean=sum/length;
        return mean;
}
public double evaluate(String expression){
        StringTokenizer token;
        ArrayList<Integer> arrlist;
        token=new StringTokenizer(expression,"(");
        String tempt=token.nextToken();
        String temp=token.nextToken();
        token=new StringTokenizer(temp,")");
        temp=token.nextToken();
        token=new StringTokenizer(temp,",");
        arrlist=new ArrayList<>();
        while(token.hasMoreTokens()){
                arrlist.add(Integer.parseInt(token.nextToken()));
        }
        int arr[]=new int[arrlist.size()];
        for(int i=0;i<arrlist.size();i++){
                arr[i]=(int)arrlist.get(i);
        }
        if(tempt.equals("min")){
                return min(arr);
        }
        else if(tempt.equals("max")){
                return max(arr);
        }
        else if(tempt.equals("mean")){
```

```java
                    return mean(arr);
            }
            else if(tempt.equals("sd")){
                    return standarddev(arr);
            }
            else if(tempt.equals("var")){
                    return var(arr);
            }
            return 0.0;
        }

        @Override
        public float add(float a, float b) {
            // TODO Auto-generated method stub
            return 0;
        }

        @Override
        public float sub(float a, float b) {
            // TODO Auto-generated method stub
            return 0;
        }

        @Override
        public float mult(float a, float b) {
            // TODO Auto-generated method stub
            return 0;
        }

        @Override
        public float div(float a, float b) {
            // TODO Auto-generated method stub
            return 0;
        }
}

//geometric
package calculator_logic;
import java.util.Stack;
public class GeometricOperations implements CalculatorInterface,GeometricOp {

        @Override
        public float add(float a, float b) {
```

```java
        // TODO Auto-generated method stub
        return a+b;
}

@Override
public float sub(float a, float b) {
        // TODO Auto-generated method stub
        return a-b;
}

@Override
public float mult(float a, float b) {
        // TODO Auto-generated method stub
        return a*b;
}

@Override
public float div(float a, float b) {
        // TODO Auto-generated method stub
        return b/a;
}

@Override
public double sin(double a) {
        // TODO Auto-generated method stub
        return Math.sin(a);
}

@Override
public double cos(double a) {
        // TODO Auto-generated method stub
        return Math.cos(a);
}

@Override
public double tan(double a) {
        // TODO Auto-generated method stub
        return Math.tan(a);
}

public double evaluate(String expression)
{
        char[] tokens;
```

```java
Stack<Double> values;
Stack<Character> ops;
tokens = expression.toCharArray();
values = new Stack<>();
ops = new Stack<Character>();
for (int i = 0; i < tokens.length; i++)
{
        if (tokens[i] == ' ')
                continue;
        if (tokens[i] >= '0' && tokens[i] <= '9')
        {
                StringBuffer sbuf = new StringBuffer();
                while (i < tokens.length && tokens[i] >= '0' && tokens[i] <= '9')
                        sbuf.append(tokens[i++]);
                values.push(Double.parseDouble(sbuf.toString()));
        }
        else if(tokens[i] == 's'){
                i=i+3;                          //sin45 + cos45
                StringBuffer temp = new StringBuffer();
                while(i<tokens.length && tokens[i] >= '0' && tokens[i]<='9'){
                        temp.append(tokens[i++]);
                }

values.push(Double.parseDouble(""+sin(Double.parseDouble(temp.toString()))));
        }
        else if(tokens[i] == 'c'){
                i=i+3;                          //sin45 + cos45
                StringBuffer temp = new StringBuffer();
                while(i<tokens.length && tokens[i] >= '0' && tokens[i]<='9'){
                        temp.append(tokens[i++]);
                }

values.push(Double.parseDouble(""+cos(Double.parseDouble(temp.toString()))));
        }
        else if(tokens[i] == 't'){
                i=i+3;                          //sin45 + cos45
                StringBuffer temp = new StringBuffer();
                while(i<tokens.length && tokens[i] >= '0' && tokens[i]<='9'){
                        temp.append(tokens[i++]);
                }

values.push(Double.parseDouble(""+tan(Double.parseDouble(temp.toString()))));
        }
```

```java
                else if (tokens[i] == '(')
                        ops.push(tokens[i]);
                else if (tokens[i] == ')')
                {
                        while (ops.peek() != '(')
                        values.push(applyOp(ops.pop(), values.pop(), values.pop()));
                        ops.pop();
                }
                else if (tokens[i] == '+' || tokens[i] == '-' ||
                                tokens[i] == '*' || tokens[i] == '/')
                {
                        while (!ops.empty() && hasPrecedence(tokens[i], ops.peek()))
                        values.push(applyOp(ops.pop(), values.pop(), values.pop()));
                        ops.push(tokens[i]);
                }
        }
        while (!ops.empty())
                values.push(applyOp(ops.pop(), values.pop(), values.pop()));
        return values.pop();
}
protected boolean hasPrecedence(char op1, char op2)
{
        if (op2 == '(' || op2 == ')')
                return false;
        if ((op1 == '*' || op1 == '/') && (op2 == '+' || op2 == '-'))
                return false;
        else
                return true;
}
protected double applyOp(char op, double b, double a)
{
        float a1=Float.parseFloat(a+"");
        float b1=Float.parseFloat(b+"");
        switch (op)
        {
        case '+':
                return add(a1,b1);
        case '-':
                return sub(a1,b1);
        case '*':
                return mult(a1,b1);
        case '/':
                if (b1 != 0)
```

```java
                        return div(a1,b1);
            }
            return 0;
        }
}

//basic calculator
package calculator_logic;
import java.util.Stack;
public class BasicCalculator implements CalculatorInterface {

        @Override
        public float add(float a, float b) {
                // TODO Auto-generated method stub
                return a+b;
        }

        @Override
        public float sub(float a, float b) {
                // TODO Auto-generated method stub
                return a-b;
        }

        @Override
        public float mult(float a, float b) {
                // TODO Auto-generated method stub
                return a*b;
        }

        @Override
        public float div(float a, float b) {
                // TODO Auto-generated method stub
                return b/a;
        }

        public double evaluate(String expression)
        {
                Stack<Float> values;
                Stack<Character> ops;
                char[] tokens;
                tokens = expression.toCharArray();
                values = new Stack<>();
                ops = new Stack<Character>();
```

```java
		for (int i = 0; i < tokens.length; i++)
		{
			if (tokens[i] == ' ')
				continue;
			if (tokens[i] >= '0' && tokens[i] <= '9')
			{
				StringBuffer sbuf = new StringBuffer();
				while (i < tokens.length && tokens[i] >= '0' && tokens[i] <= '9')
					sbuf.append(tokens[i++]);
				values.push(Float.parseFloat(sbuf.toString()));
			}
			else if (tokens[i] == '(')
				ops.push(tokens[i]);
			else if (tokens[i] == ')')
			{
				while (ops.peek() != '(')
				values.push(applyOp(ops.pop(), values.pop(), values.pop()));
				ops.pop();
			}
			else if (tokens[i] == '+' || tokens[i] == '-' ||
						tokens[i] == '*' || tokens[i] == '/')
			{
				while (!ops.empty() && hasPrecedence(tokens[i], ops.peek()))
				values.push(applyOp(ops.pop(), values.pop(), values.pop()));
				ops.push(tokens[i]);
			}
		}
		while (!ops.empty())
			values.push(applyOp(ops.pop(), values.pop(), values.pop()));
		return values.pop();
}
protected static boolean hasPrecedence(char op1, char op2)
{
	if (op2 == '(' || op2 == ')')
		return false;
	if ((op1 == '*' || op1 == '/') && (op2 == '+' || op2 == '-'))
		return false;
	else
		return true;
}
protected float applyOp(char op, float b, float a)
{
	switch (op)
```

```java
            {
            case '+':
                    return add(a,b);
            case '-':
                    return sub(a,b);
            case '*':
                    return mult(a,b);
            case '/':
                    if (b != 0)
                            return div(a,b);
            }
            return 0;
        }

}

//factory class
package calculator_logic;

public class CalculatorFactory {
        public CalculatorInterface getOperation(String opType){
            if(opType == null){
              return null;
            }
            if(opType.contains("sin")|| opType.contains("cos")||opType.contains("tan")){
              return new GeometricOperations();
            }
            else if(opType.contains("min")||opType.contains("max")||
opType.contains("min")||opType.contains("mean")||opType.contains("sd")||opType.contains("var
"))
            {
              return new AlgebricOperations();
            }
            else{
                return new BasicCalculator();
            }
          }
}
```