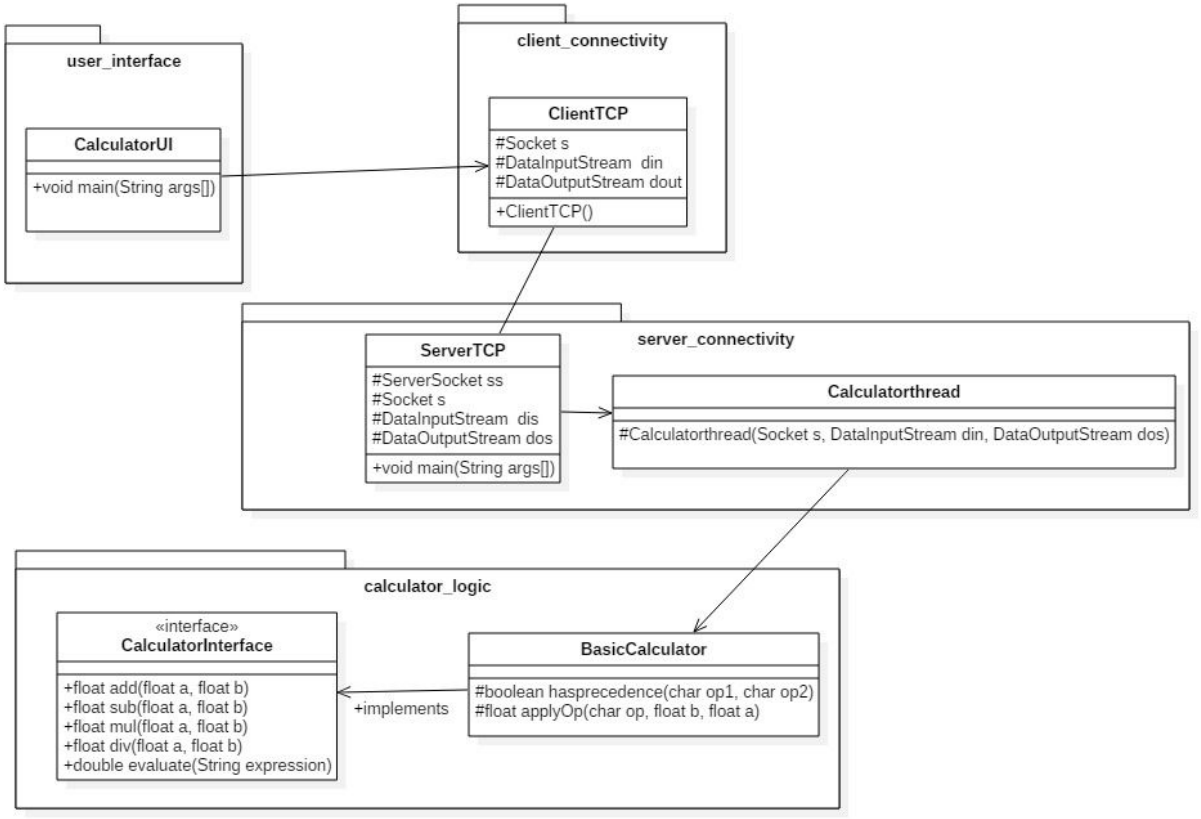
Tutorial No. 2

Q.1 implement calculator program using basic operations addition,subtraction,multiplication,division based on monolithic and client-server architecture using thread.(different thread should be allocated to different client).

Class Diagram:

Client-server:

fig.class Diagram: Calculator_clientserverthread_basic


user_interface
CalculatorUI
+void main(String args[])
client_connectivity
ClientTCP
#Socket s
#DataInputStream din
#DataOutputStream dout
+ClientTCP()
server_connectivity
ServerTCP
#ServerSocket ss
#Socket s
#DataInputStream dis
#DataOutputStream dos
+void main(String args[])
Calculatorthread
#Calculatorthread(Socket s, DataInputStream din, DataOutputStream dos)
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
BasicCalculator
#boolean hasprecedence(char op1, char op2)
#float applyOp(char op, float b, float a)

Implementation:

```
//user interface client
package user_interface;
```

```
import client_connectivity.*;
public class CalculatorUI {
        public static void main(String[] args) {
           new ClientTCP();
        }

}


//client control file
package client_connectivity;
import java.io.*;
import java.net.*;
import java.util.*;
public class ClientTCP {
        protected Socket s;
    protected DataInputStream din;
    protected DataOutputStream dout;
    public ClientTCP()
    {
    try
    {

       s=new Socket("localhost",10);
       //System.out.println(s);
       din= new DataInputStream(s.getInputStream());
       dout= new DataOutputStream(s.getOutputStream());
       ClientChat();
    }
    catch(Exception e)
    {
       System.out.println(e);
    }
```

```
  }
  void ClientChat() throws IOException
  {
    String choice;


    do{
        Scanner scan=new Scanner(System.in);
        System.out.println("Enter the Expression : ");
        String exp=scan.nextLine();
        dout.writeUTF(exp);
        System.out.println(din.readUTF());
        String servermsg=din.readUTF();
        System.out.println(servermsg);
        choice=scan.next();
        dout.writeUTF(choice);
        dout.flush();
    }while(choice.equals("Y"));
  }
}

//server control
package server_connectivity;
import calculator_logic.*;
import java.io.*;
import java.net.*;
public class ServerTCP {
    protected ServerSocket ss;
    protected Socket s;
    protected DataInputStream dis;
    protected DataOutputStream dos;
    public static void main(String[] args) {
        new ServerTCP();
    }
    public ServerTCP()
    {
        try {
            ServerSocket ss=new ServerSocket(10);
            while(true)
            {

            Socket s= null;
        try
        {
            s=ss.accept();
            System.out.println("A new client is connected : " + s);
            dis= new DataInputStream(s.getInputStream());
            dos= new DataOutputStream(s.getOutputStream());
            System.out.println("Assigning new thread for this client");
      Thread t = new Calculatorthread(s, dis, dos);
      t.start();
        }
        catch(Exception e){
            System.out.println(e);
        }
            }
        }catch(Exception e){};
    }
    protected class Calculatorthread extends Thread{
        Socket s;
        DataInputStream din;
        DataOutputStream dos;
        public Calculatorthread(Socket s,DataInputStream
din,DataOutputStream dos){
            this.s=s;
            this.din=din;
            this.dos=dos;
        }
        public void run(){
            try{
                do{
                    String exp=din.readUTF();
                    BasicCalculator cal=new BasicCalculator();
                    dos.writeUTF(cal.evaluate(exp)+"");
                    dos.writeUTF("Do you want to
continue(Y/N)");
                }while(din.readUTF().equals("Y"));
            }
            catch(Exception e){}
        }
}
}
```

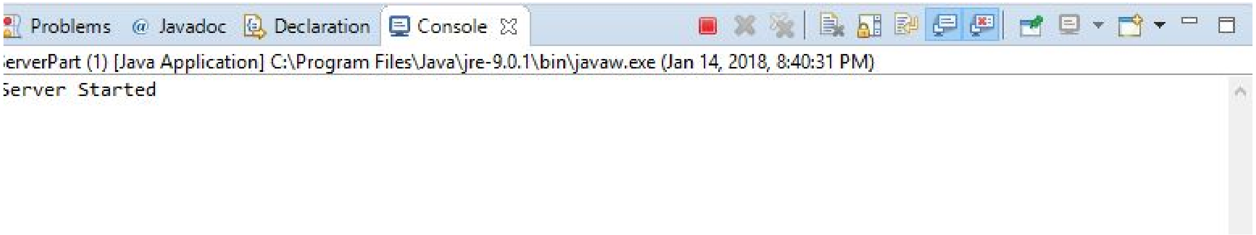
```
//Business logic
//interfaces
package calculator_logic;

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

```
//concrete implementation
 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 mul(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>();
                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()));
```

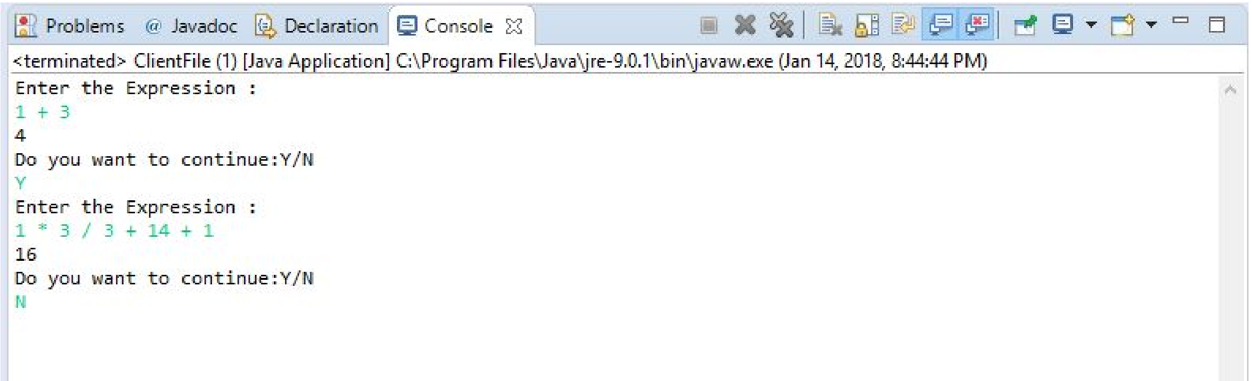
```
                ops.push(tokens[i]);
            }
        }
        while (!ops.empty())
            values.push(applyOp(ops.pop(), values.pop(),
values.pop()));
        return values.pop();
    }
    protected static boolean hasPrecedence(char op1, char op2)
    {
        if (op2 == '(' || op2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

```
}
```

Output:



```
erverPart (1) [Java Application] C:\Program Files\Java\jre-9.0.1\bin\javaw.exe (Jan 14, 2018, 8:40:31 PM)
erver Started
```



```
<terminated> ClientFile (1) [Java Application] C:\Program Files\Java\jre-9.0.1\bin\javaw.exe (Jan 14, 2018, 8:44:44 PM)
Enter the Expression :
1 + 3
4
Do you want to continue:Y/N
Y
Enter the Expression :
1 * 3 / 3 + 14 + 1
16
Do you want to continue:Y/N
N
```