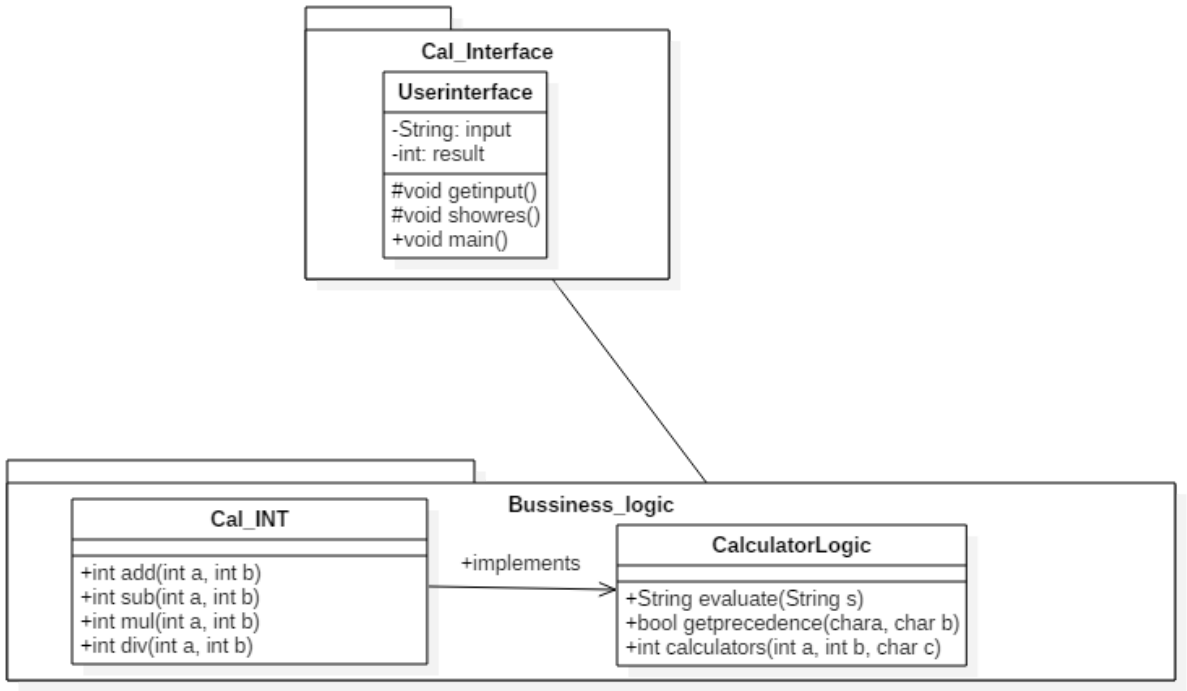
Tutorial No. 1

Q.1 implement calculator program using basic operations addition,subtraction,multiplication,division based on monolithic and client-server architecture.

Class diagram:

Monolithic:



Implementation:

```
//userinterface
package Monolithic;
import basic_calculator.*;
import java.util.*;
public class Userinterface {
```

```
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner scan=new Scanner(System.in);
		String c="Y";
		while(c.equals("Y"))
		{
		System.out.println("Enter the expression : ");
		String exp;
		exp=scan.nextLine();
		CalculatorLogic calculator=new CalculatorLogic();
		int result=calculator.evaluate(exp);
		showResult(result);
		System.out.print("Do You not continue:(Y/N)");
		c=scan.nextLine();
		}
		scan.close();
	}
	static void showResult(int res){
		System.out.println("Answer is "+res);
	}
}
```

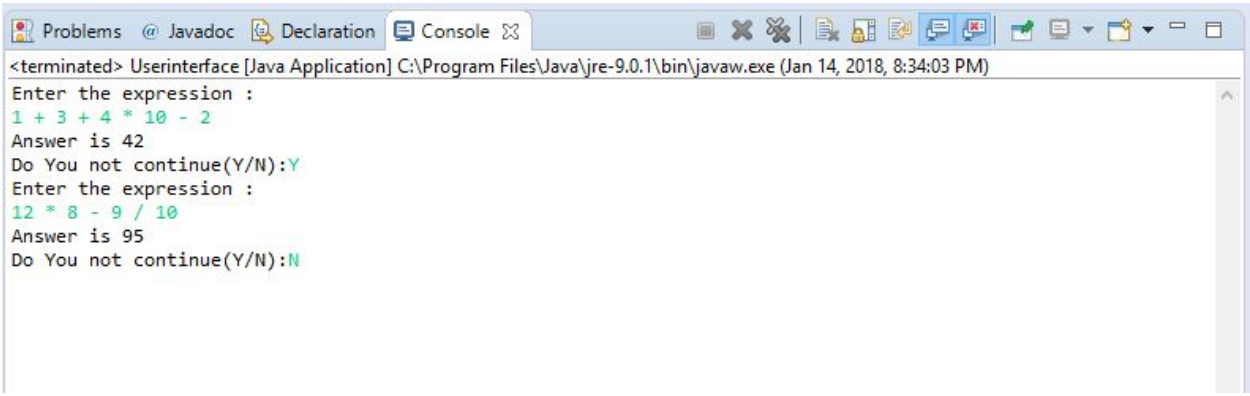
```
//business logic

package basic_calculator;
import java.util.Stack;
import java.util.*;

public class CalculatorLogic {
```

```
        static public int add(int a,int b){
                return a+b;
        }
        static public int sub(int a,int b){
                return a-b;
        }
        static public int mult(int a,int b){
                return a*b;
        }
        static public int div(int a,int b){
                return a/b;
        }
        public static int evaluate(String expression)
        {
                char[] tokens = expression.toCharArray();
                Stack<Integer> values = new Stack<Integer>();
                Stack<Character> 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(Integer.parseInt(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();
    }
    public static boolean hasPrecedence(char op1, char op2)
    {
        if (op2 == '(' || op2 == ')')
                return false;
        if ((op1 == '*' || op1 == '/') && (op2 == '+' || op2 == '-'))
                return false;
        else
                return true;
    }
    public static int applyOp(char op, int b, int a)
    {
        switch (op)
        {
```

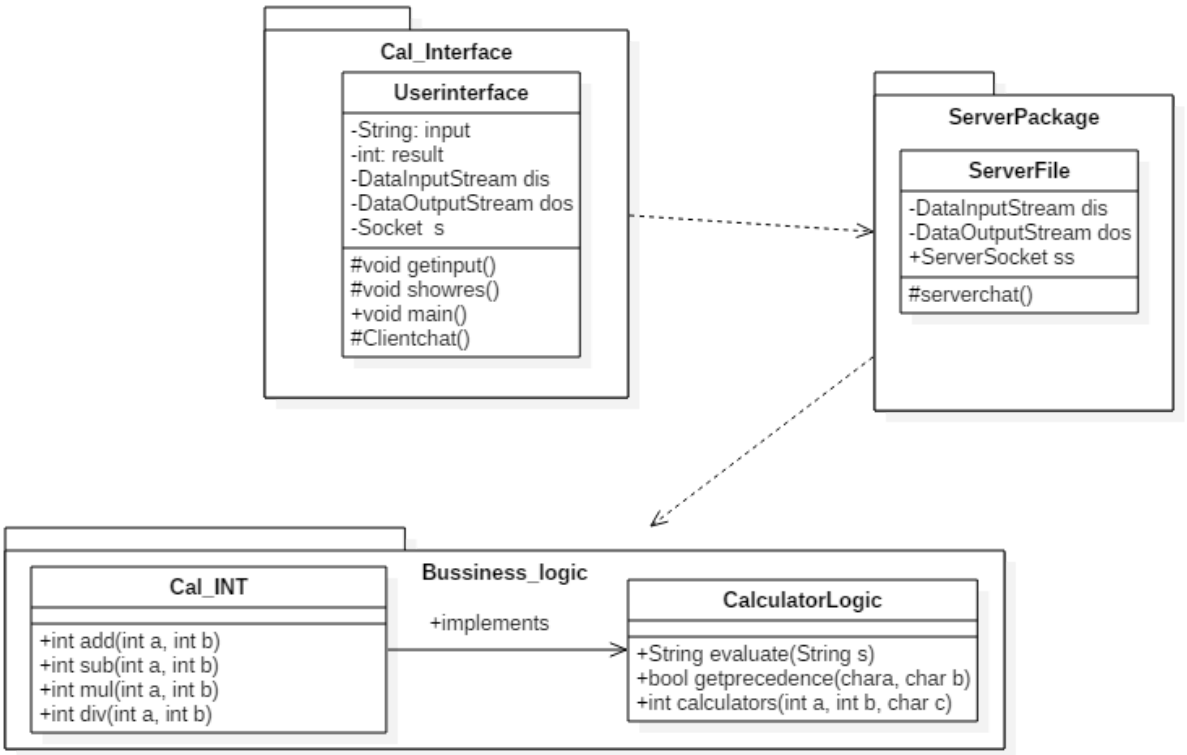
```
            case '+':
                return add(a,b);
            case '-':
                return sub(a,b);
            case '*':
                return mult(a,b);
            case '/':
                if (b != 0)
                    return div(b, a);
            }
            return 0;
      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

output:



Class Diagram:

Client-server:



Implementation:

```
//user interface client
package ClientPackage;
import java.io.*;
import java.net.*;
import java.util.*;

public class ClientFile {

        Socket s;
    DataInputStream din;
    DataOutputStream dout;

    public static void main(String as[])
    {
        new ClientFile();
```

```
}


public ClientFile()
{
    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);
    }
}
public 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 file
package ServerPackage;
import java.io.*;
import basic_calculator.*;
import java.net.*;
import java.util.Stack;
import java.util.*;

public class ServerPart {
        ServerSocket ss;
        Socket s;
        DataInputStream dis;
        DataOutputStream dos;

        public static void main(String[] args) {
                new ServerPart();
        }
        public ServerPart()
        {
                try
                {
                        System.out.println("Server Started");
                        ss=new ServerSocket(10);
                        s=ss.accept();
                        System.out.println(s);
                        System.out.println("Client Connected");
                        dis= new DataInputStream(s.getInputStream());
                        dos= new DataOutputStream(s.getOutputStream());
```

```
                ServerChat();
            }
            catch(Exception e)
            {
                System.out.println(e);
            }
        }

        public void ServerChat() throws IOException
        {
            do{
                int c=0;
                String resmsg;
                CalculatorLogic cal=new CalculatorLogic();
                String expression=dis.readUTF();
                int answer=cal.evaluate(expression);
                String res=Integer.toString(answer);
                dos.writeUTF(res);
                String conmsg="Do you want to continue:Y/N";
                dos.writeUTF(conmsg);

                dos.flush();
            }while(dis.readUTF().equals("Y"));
        }
        }
```

//Business logic

-same as monolithic

<u>Output:</u>

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

```
Problems  @ Javadoc  Declaration  Console
<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
```