Module: Applied Distributed System Programming

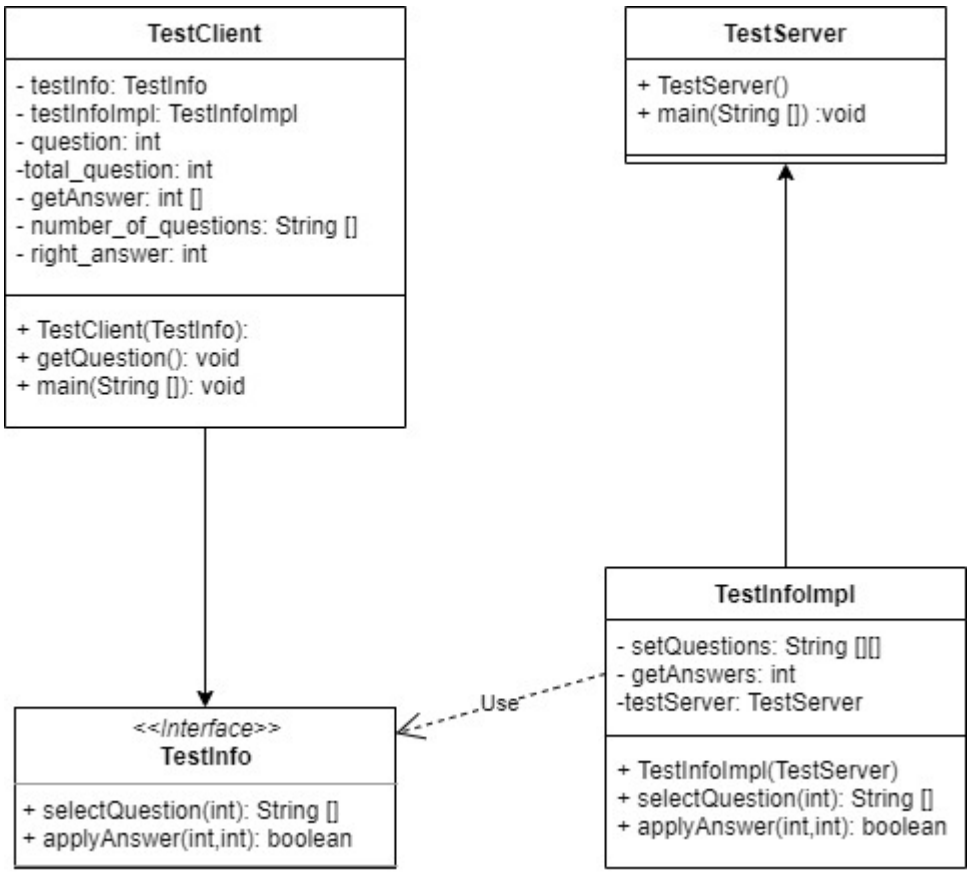
RMI– Based Client/Server communication System in Java

Student ID: W1669150

Name: Valdir Santos Fonseca

Date: 24/03/2020

Class Diagram



Summary Tables

Class TestClient

Fields	private TestInfo testInfo	Attribute that is in use to contact the server and to call the remote methods for example apply the answers
	private TestInfoImpl testInfoImpl	Attribute that is in use to contact the testInfoImpl and to call the methods to select and apply the answer
	private int question	This is the number of the question that is in use and it increments to the next question, the server needs to know this question to release the next one
	private int total_Questions	Number of the questions that has been answer correctly
	private int [] getAnswer	Attribute where is getting the current answer
	private String [] number_of_questions	Questions to be answered
	private int right_answer = 0;	Increment each time the answer is correcr
Constructor	public TestClient()	Construtor to create a new Client
Methods	private void nextQuestion()	Method that returns the next question
	public static void main(String[] args)	The main program that get the successful connection with the server, the launch the client

Class TestServer

Constructor	public class TestServer	Constructor that creates a server with the questions for the client
Methods	public static void main(String[] args)	Main program that start the RMI Server

Class TestInfoImpl

Fields	private String [][] setQuestions	Array that contain all question and the choices to be selected.
	public int [] getAnswers	This Array contain all correct answers for the questions
	private TestServer testServer;	Server object for the activities
Constructor	public TestInfoImpl(TestServer testServer)	Create a constructor that initialize the testInfo for the RMI
Methods	public String[] selectQuestion(int num_of_question)	Method that given the number of the question it returns the question and the choice for the Client
	public boolean applyAnswer(int num_of_question, int select_answer)	Method that given the number of the question and the choice it returns true or false if it's wrong or right

Java Code

TestClient

```
/*
 * @Author Valdir Fonseca
   Student Number: W1669150
 */
package RMI;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.util.Arrays;
import java.util.Scanner;



public class TestClient
{
    private TestInfo testInfo;
    private TestInfoImpl testInfoImpl;
    private int question = 0;
    private int total_Questions = 1;
    private int [] getAnswer = new int[3];
    private String [] number_of_questions = new String[3];
```

```
    private int right_answer = 0;
    Scanner in = new Scanner(System.in);


    /*
       Creating a new Client
    */
    public TestClient()
    {
       try
       {
          //Showing the questions to be answered
          for (int i = 0; i < number_of_questions.length; i++)
          {
            System.out.println(i+1+" - "+Arrays.toString(testInfo.selectQuestion(i)));
          }


          //Method giving the options choices
          for(int i = 0; i < getAnswer.length; i++)
          {
             System.out.println("Please Select your answer with number 1, 2 or 3 as the right
answer!!");
             getAnswer[i] = in.nextInt();
          }
          total_Questions++;
          System.out.println();
```

```
            nextQuestion();
            //Throws a exception in case an occur error
        }catch (RemoteException re) {
            System.out.println();
            System.out.println("RemoteException");
            System.out.println(re);
        }
    }

    private TestClient(TestInfo testInfo) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }
    /*
        Presenting the questions to the user
    */
    private void nextQuestion()
    {
        try {
            String [] actualQuestion = testInfo.selectQuestion(question);

            // present the right answers
            for(int i = 0; i < number_of_questions.length; i++)
            {
                if(getAnswer[i]==testInfoImpl.getAnswers[i])
```

```
                right_answer++;
            }
            // End of the game no more questions will show up
            if (actualQuestion == null) {
                System.out.println("You concluded your Math Test and you achieved " +
right_answer +
                        "correct answers of " + total_Questions + " in Total.");
                System.exit(0);
            }

            total_Questions++;

        } catch (RemoteException re) {
            System.out.println();
            System.out.println("RemoteException");
            System.out.println(re);
        }
    }
    public static void main(String[] args) throws Exception
    {
      TestInfo testInfo1 = (TestInfo) Naming.lookup("rmi://localhost:2000/MathQuestions");
      TestClient cl = new TestClient(testInfo1);
      cl.nextQuestion();
    }
}
```

TestServer

```
/*
 * @Author Valdir Fonseca
   Student Number: W1669150
 */
package RMI;
import java.rmi.Naming;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.awt.*;
import javax.swing.*;



public class TestServer extends JFrame {


     private JTextArea logTextArea = new JTextArea();


    public TestServer ()
    {
        setTitle("ServerTest");
        setLayout(new BorderLayout());
        setSize(500, 500);
        setLocationRelativeTo(null);
```

```
        setDefaultCloseOperation(EXIT_ON_CLOSE);


        add(BorderLayout.CENTER, new JScrollPane(logTextArea));
    }

/**
     * Append the info on the board as soon the server starts
     */
    public void logMessage(String info) {
        logTextArea.append(info + "\n");
    }


    public static void main(String[] args) throws Exception {

         TestServer testServer1 = new TestServer();


        try {
            //Shows the graphical page with the information that the server has initialized
            testServer1.setVisible(true);
            TestInfoImpl testInfo = new TestInfoImpl(testServer1);


            // Connection on the port 2000
             Registry registry = LocateRegistry.createRegistry(2000);
```

```
            LocateRegistry.getRegistry();
            registry.rebind("MathQuestions", testInfo);
            testServer1.logMessage("Server has started...");

        } catch (Exception e)
        {
            testServer1.logMessage("Server has Failed: " + e);
        }
    }
}
```

TestInfoImpl

```
/*
 * @Author Valdir Fonseca
   Student Number: W1669150
 */
package RMI;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import javax.swing.JOptionPane;



public class TestInfoImpl extends java.rmi.server.UnicastRemoteObject implements TestInfo
{


    // List of the Questions
    private String [][] setQuestions = {
        {"Q1: (A + B)*(A + B)", "1. A*A + B*B", "2. A*A + A*B + B*B", "3. A*A + 2*A*B
+ B*B"},
        {"Q2: (A + B)*(A - B)", "1. A*A + 2*B*B", "2. A*A - B*B", "3. A*A -2*A*B +
B*B"},
        {"Q3: sin(x)*sin(x) + cos(x)*cos(x)", "1. 1", "2. 2", "3. 3"}
    };


    //Attribute with the right answers
    public int [] getAnswers = {3,2,1};
```

```
private TestServer testServer;


//The Server Inialize
public TestInfoImpl(TestServer testServer) throws java.rmi.RemoteException
{
    super();
    this.testServer = testServer;
}


/*
    Method with the given number of question will return the question for the user
*/
 public String[] selectQuestion(int num_of_question) throws RemoteException
{
    if((num_of_question < 0) || num_of_question > setQuestions.length)
        return null;


   JOptionPane.showMessageDialog(null, "Message recovered!");
   System.out.println("Message recovered!");
   return setQuestions[num_of_question];
}
```

```
    /*
       Method with the confirmation from the server when the answer has been submitted
     */
    public boolean applyAnswer(int num_of_question, int select_answer) throws
RemoteException
    {
       if((num_of_question > 0) || (num_of_question > setQuestions.length))
       return false;

       System.out.println("Answer inserted!");
       return getAnswers[num_of_question] == select_answer;
    }
}
```

TestInfo

```
/*
 * @Author Valdir Fonseca
   Student Number: W1669150
 */
package RMI;


import java.rmi.Remote;
import java.rmi.RemoteException;


public interface TestInfo extends java.rmi.Remote {


    //Method that receive the question number and return the choice
    public String[] selectQuestion(int num_of_question) throws java.rmi.RemoteException;


    //Given the answer the Server will answer if the question is right or wrong
    public boolean applyAnswer(int num_of_question, int select_answer) throws
java.rmi.RemoteException;
}
```