**PRACTICAL NO. 3**

**Design a Graphical User Interface (GUI) based calculator. (scientific or standard). Operations should be performed using both mouse and keyboard.**

**Code Files :-**

1. **Server.java file**

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class CalculatorServer {

public static void main(String[] args) {

// TODO Auto-generated method stub

try {

Calculator calculator = new CalculatorImplementation();

Registry registry = LocateRegistry.createRegistry(1099);

registry.rebind("Calculator", calculator);

System.out.println("Calculator Server is ready.");

}

catch (Exception e) {

// TODO: handle exception

e.printStackTrace();

}

}

}

1. **Client.java file**

import java.awt.BorderLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JTextField;

public class CalculatorClient extends JFrame {

private static final long serialVersionUID = 1L;

private JTextField display;

private double num1, num2, result;

private String operator;

public CalculatorClient() {

setTitle("Calculator");

setSize(300, 400);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

display = new JTextField();

display.setSize(100, 50);

display.setEditable(false);

add(display, BorderLayout.NORTH);

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(4, 4));

String[] buttons = {

"7", "8", "9", "/",

"4", "5", "6", "\*",

"1", "2", "3", "-",

"0", "C", "=", "+"

};

for (String text : buttons) {

JButton button = new JButton(text);

button.addActionListener(new ButtonClickListener());

panel.add(button);

}

add(panel, BorderLayout.CENTER);

setVisible(true);

}

private class ButtonClickListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

String command = e.getActionCommand();

try {

Registry registry = LocateRegistry.getRegistry("localhost", 1099);

Calculator calculator = (Calculator) registry.lookup("Calculator");

switch (command) {

case "C":

display.setText("");

break;

case "=":

num2 = Double.parseDouble(display.getText());

switch (operator) {

case "+":

result = calculator.add(num1, num2);

break;

case "-":

result = calculator.subtract(num1, num2);

break;

case "\*":

result = calculator.multiply(num1, num2);

break;

case "/":

result = calculator.divide(num1, num2);

break;

}

display.setText(String.valueOf(result));

break;

default:

if ("+-\*/".contains(command)) {

operator = command;

num1 = Double.parseDouble(display.getText());

display.setText("");

} else {

display.setText(display.getText() + command);

}

break;

}

} catch (Exception ex) {

display.setText("Error");

}

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

new CalculatorClient();

}

1. **CalcOperation.java**

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class CalculatorImplementation extends UnicastRemoteObject implements Calculator {

private static final long serialVersionUID = 1L;

public CalculatorImplementation() throws RemoteException {

super();

}

@Override

public double add(double a, double b) throws RemoteException {

// TODO Auto-generated method stub

return a+b;

}

@Override

public double subtract(double a, double b) throws RemoteException {

// TODO Auto-generated method stub

return a-b;

}

@Override

public double multiply(double a, double b) throws RemoteException {

// TODO Auto-generated method stub

return a\*b;

}

@Override

public double divide(double a, double b) throws RemoteException {

// TODO Auto-generated method stub

return a/b;

}

}

1. **Calculator.java(interface)**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface Calculator extends Remote {

double add(double a, double b) throws RemoteException;

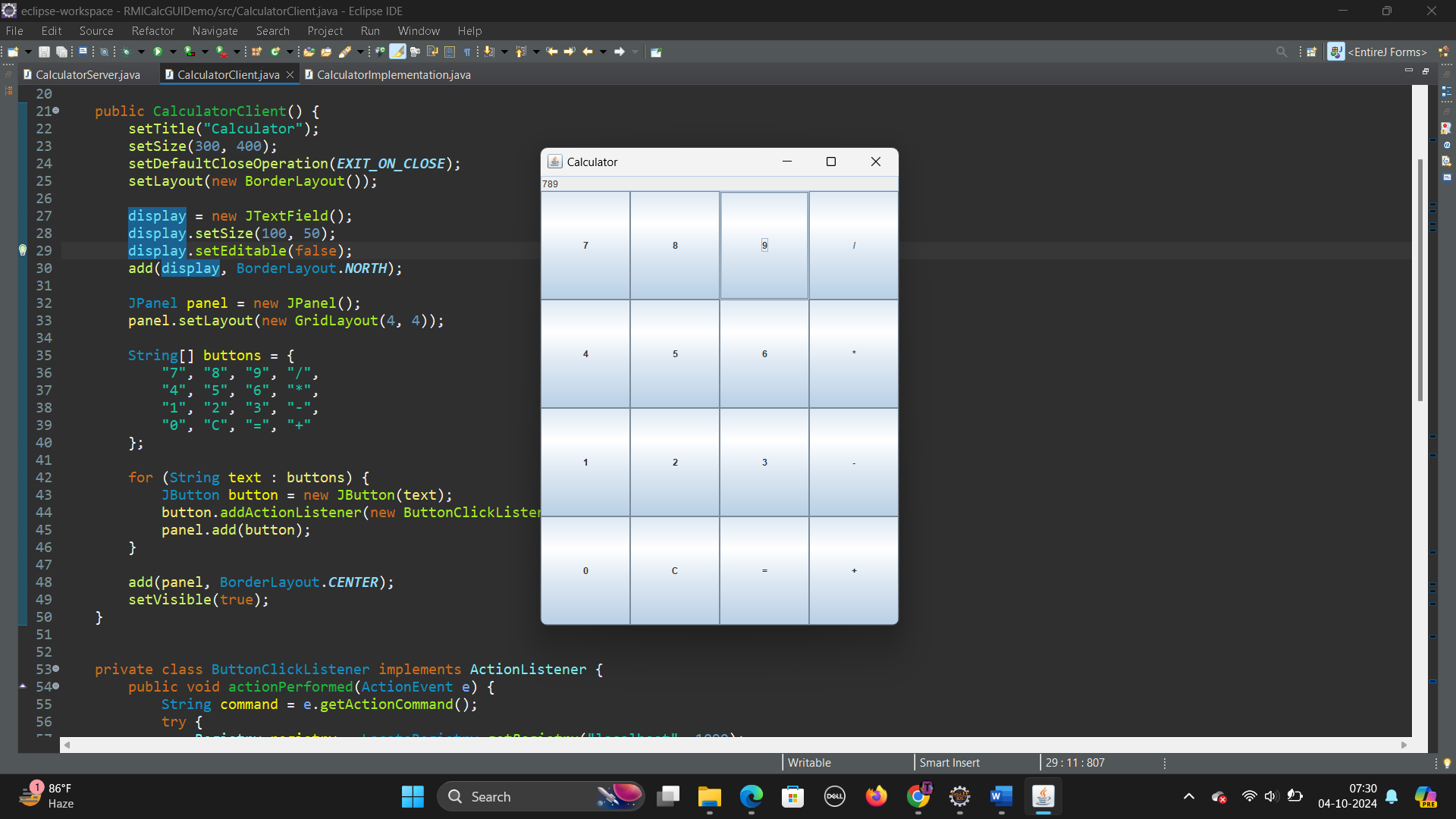
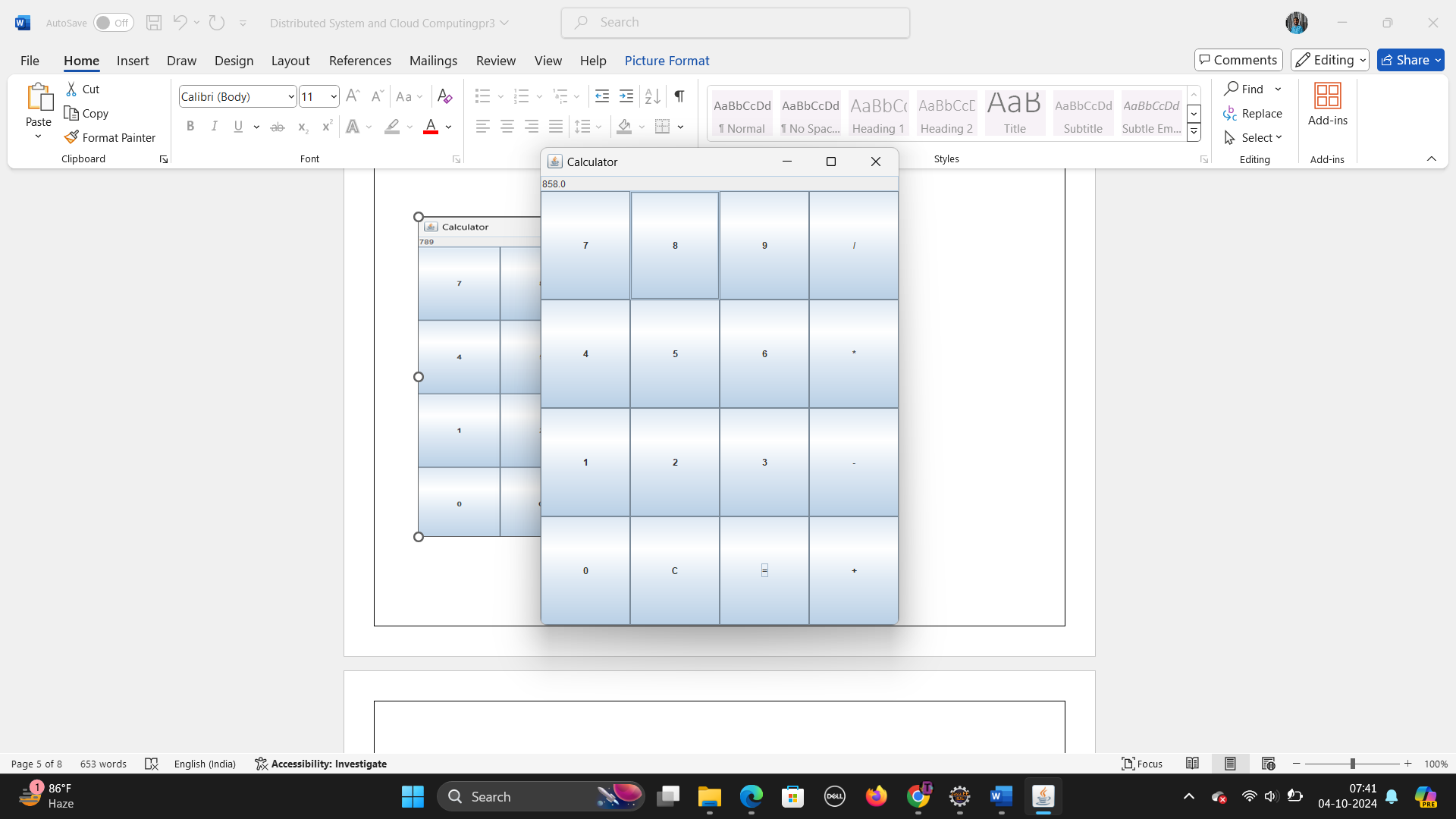
double subtract(double a, double b) throws RemoteException;

double multiply(double a, double b) throws RemoteException;

double divide(double a, double b) throws RemoteException;

}

**Output :-**

Addition

**2. Retrieve day, time and date function from server to client. This program should display server day, date and time.**

**Code Files :-**

1. **DateTimeServiceClient.java**

package RMIDemo;

import java.rmi.Naming;

import java.util.Date;

public class DateTimeClient {

public static void main(String[] args) {

try {

DateTimeService dateTimeService = (DateTimeService) Naming.*lookup*("rmi://localhost:1900/DateTimeService");

Date serverDateTime = dateTimeService.getCurrentDateTime();

System.***out***.println("Current Date and Time from Server: " + serverDateTime.toString());

String serverDay = dateTimeService.getCurrentDay();

System.***out***.println("Current Day from Server: " + serverDay);

} catch (Exception e) {

e.printStackTrace();

}

}

}

1. **DateTimeServer.java**

package RMIDemo;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class DateTimeServer {

public static void main(String[] args) {

try {

DateTimeService dateTimeService = new DateTimeServiceImpl();

LocateRegistry.*createRegistry*(1900);

Naming.*rebind*("rmi://localhost:1900/DateTimeService", dateTimeService);

System.***out***.println("DateTimeServer is running...");

} catch (Exception e) {

e.printStackTrace();

}

}

}

1. **DateTimeServiceImpl.java**

package RMIDemo;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Locale;

public class DateTimeServiceImpl extends UnicastRemoteObject implements DateTimeService {

private static final long ***serialVersionUID*** = 1L;

protected DateTimeServiceImpl() throws RemoteException {

super();

}

*@Override*

public Date getCurrentDateTime() throws RemoteException {

return new Date();

}

*@Override*

public String getCurrentDay() throws RemoteException {

SimpleDateFormat dayFormat = new SimpleDateFormat("EEEE", Locale.***ENGLISH***);

return dayFormat.format(new Date());

}

}

1. **DateTimeService.java (Interface)**

package RMIDemo;

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.Date;

public interface DateTimeService extends Remote {

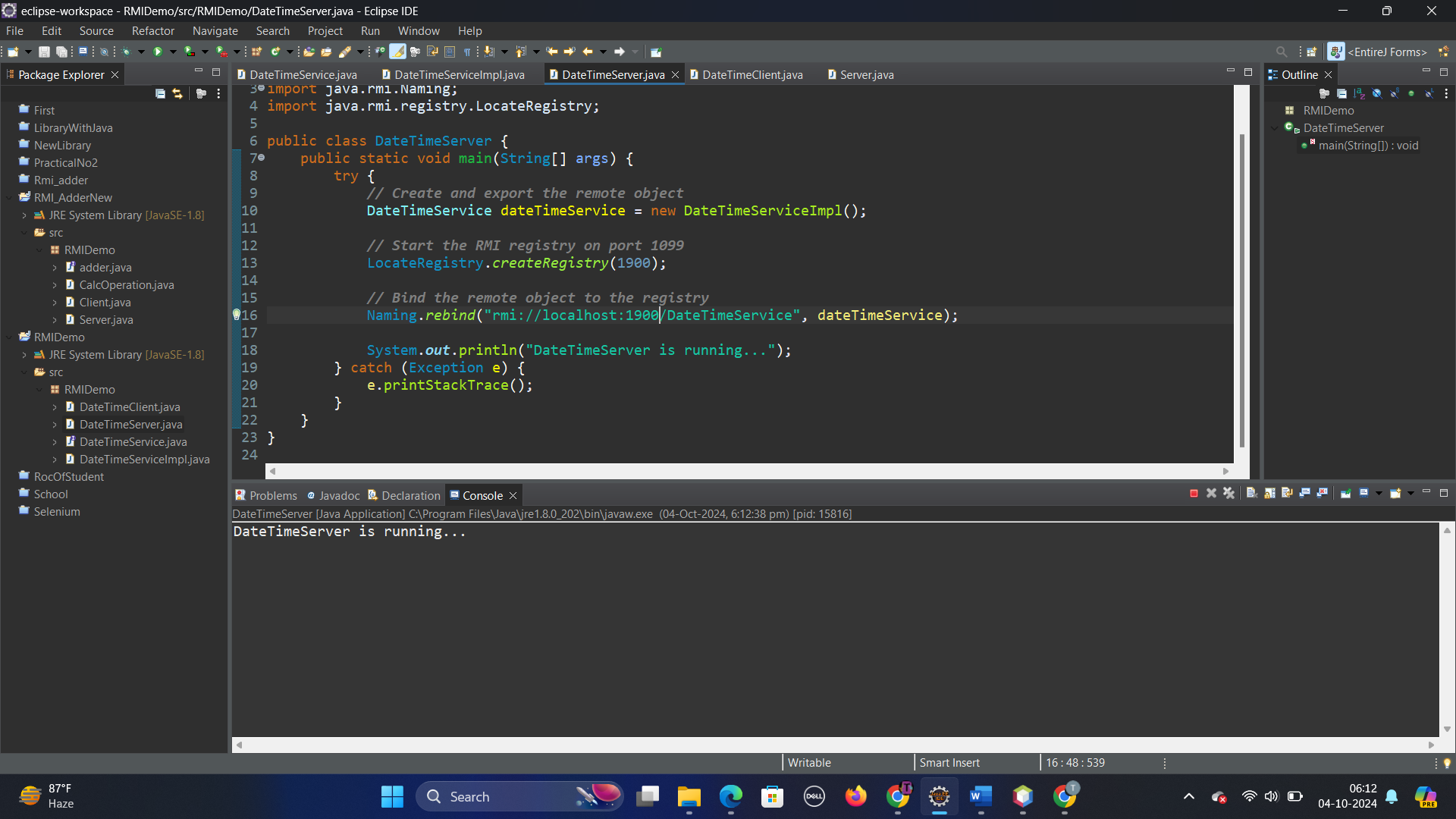
Date getCurrentDateTime() throws RemoteException;

String getCurrentDay() throws RemoteException;

}

**Output :-**

* Server



* Client

