Distributed and Parallel Computing Lab

CS461 Lab4

Name: Dipean Dasgupta **ID:202151188**

Task: Implementation of Java RMI(Remote Method Invocation)

SubTask1: Define a remote interface that declares remote methods.

In the code snippet below, the **remote interface** **CalculatorService** is defined, which extends the Remote interface from Java's RMI (Remote Method Invocation) package.

Four method signatures are defined: **add (), subtract (), multiply (),** and **divide ().** Each method takes two **double** parameters representing the operands and returns the result of the operation as a **double**.
Each method throws a **RemoteException** to handle communication-related errors that might occur during the remote invocation.

CODE:

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

public interface CalculatorService extends Remote {
    // Method signatures for basic calculator operations
    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;
}
```

Subtask 2: Implement the remote interface in a class.

Using the CalculatorServiceImpl class, which extends UnicastRemoteObject to support remote method invocation, **CalculatorServiceImpl.java** implements the CalculatorService interface. Basic arithmetic operations are carried out by the add, subtract, multiply, and divide methods; divide handles division-by-zero situations by raising an ArithmeticException. When there are network-related problems during remote calls, the class constructor takes care of any potential RemoteException.

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class CalculatorServiceImpl extends UnicastRemoteObject implements
CalculatorService {
```

```java

    protected CalculatorServiceImpl() throws RemoteException {
        super();
    }


    @Override
    public double add(double a, double b) throws RemoteException {
        return a + b;
    }


    @Override
    public double subtract(double a, double b) throws RemoteException {
        return a - b;
    }


    @Override
    public double multiply(double a, double b) throws RemoteException {
        return a * b;
    }


    @Override
    public double divide(double a, double b) throws RemoteException {
        if (b == 0) {
            throw new ArithmeticException("Division by zero is not allowed.");
        }
        return a / b;
    }
}
```

Subtask3: Create an RMI server to register the remote object.

Code of RMIServer defines the RMIServer class, which sets up an RMI server. It creates an RMI registry on port 1099 and binds the CalculatorServiceImpl instance to the name "CalculatorService" at the localhost address. This allows clients to remotely access the calculator service. If any errors occur, they are caught and printed.

CODE:

```
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;

public class RMIServer {
    public static void main(String[] args) {
        try {
            // Create the registry
            LocateRegistry.createRegistry(1099);

            // Create an instance of the CalculatorService implementation
            CalculatorService calculator = new CalculatorServiceImpl();

            // Bind the service to a name
            Naming.rebind("rmi://localhost:1099/CalculatorService", calculator);

            System.out.println("RMI Server is running...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Subtask 4: Develop an RMI client to look up the remote object and invoke its methods.

RMIClient.java implements the RMI client for the calculator service, allowing users to perform arithmetic operations remotely. It first looks up the CalculatorService via RMI and displays a menu for the user to select an operation (add, subtract, multiply, divide, or exit). The client collects user input for the numbers and the chosen operation, calls the corresponding remote method on the server, and displays the result. The program also includes input validation to ensure valid entries and handles division by zero gracefully.

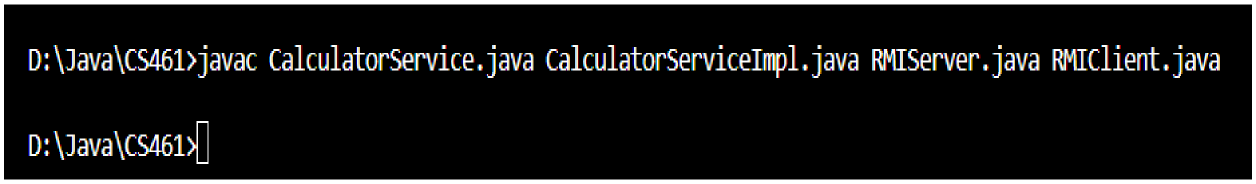
CODE:

```
import java.rmi.Naming;
import java.util.Scanner;

public class RMIClient {
    public static void main(String[] args) {
        try {
            // Lookup the remote calculator service by its name
            CalculatorService calculator = (CalculatorService)
Naming.lookup("rmi://localhost:1099/CalculatorService");
```

```
            Scanner scanner = new Scanner(System.in);
            boolean running = true; // Variable to control the loop

            while (running) {
                // Display the menu to the user
                System.out.println("\nCalculator Menu:");
                System.out.println("1. Add");
                System.out.println("2. Subtract");
                System.out.println("3. Multiply");
                System.out.println("4. Divide");
                System.out.println("5. Exit");
                System.out.print("Choose an operation: ");

                // Ensure the user inputs a valid integer choice
                while (!scanner.hasNextInt()) {
                    System.out.println("Invalid input. Please enter a number.");
                    scanner.next(); // Clear invalid input
                }

                int choice = scanner.nextInt();

                // Check if the user wants to exit
                if (choice == 5) {
                    System.out.println("Exiting...");
                    running = false;
                    continue;
                }

                // Get user input for numbers
                System.out.print("Enter first number: ");
                while (!scanner.hasNextDouble()) {
                    System.out.println("Invalid input. Please enter a valid
number.");
                    scanner.next(); // Clear invalid input
                }
                double num1 = scanner.nextDouble();

                System.out.print("Enter second number: ");
                while (!scanner.hasNextDouble()) {
                    System.out.println("Invalid input. Please enter a valid
number.");
                    scanner.next(); // Clear invalid input
                }
                double num2 = scanner.nextDouble();
                double result = 0;

                // Perform the selected operation
                switch (choice) {
                    case 1:
                        result = calculator.add(num1, num2);
                        System.out.println("Result: " + num1 + " + " + num2 + " =
" + result);
                        break;
                    case 2:
                        result = calculator.subtract(num1, num2);
                        System.out.println("Result: " + num1 + " - " + num2 + " =
" + result);
                        break;
                    case 3:
                        result = calculator.multiply(num1, num2);
                        System.out.println("Result: " + num1 + " * " + num2 + " =
" + result);
                        break;
                    case 4:
                        if (num2 == 0) {
                            System.out.println("Error: Division by zero is not
allowed.");
                        } else {
                            result = calculator.divide(num1, num2);
                            System.out.println("Result: " + num1 + " / " + num2 +
" = " + result);
                        }
                        break;
                    default:
                        System.out.println("Invalid choice. Please select a
number from 1 to 5.");
                }
            }

            scanner.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

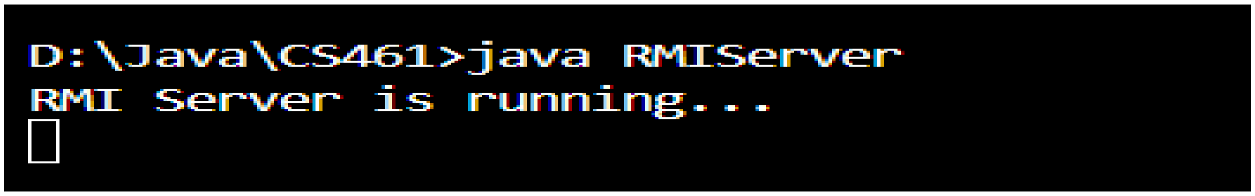
OUTPUT:

All 4 files are run simultaneously.



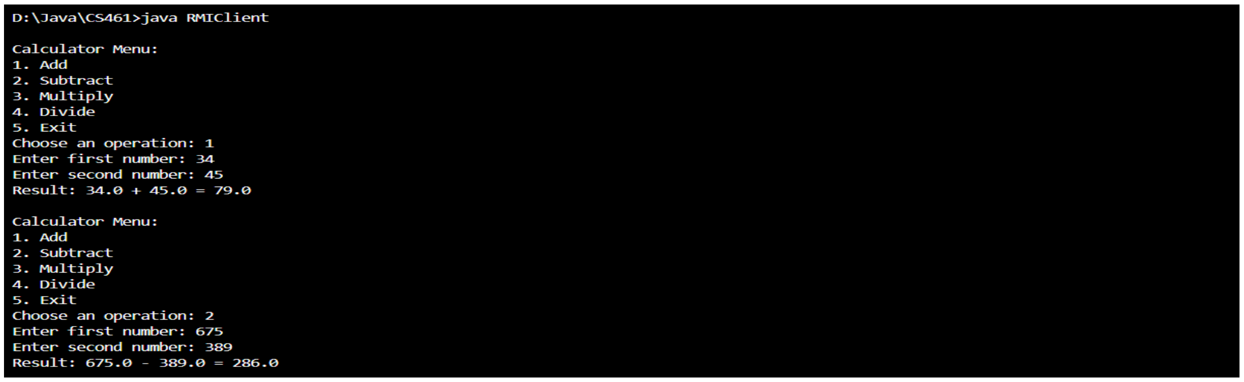
```
D:\Java\CS461>javac CalculatorService.java CalculatorServiceImpl.java RMIServer.java RMIClient.java

D:\Java\CS461>
```

RMI Server is being activated.

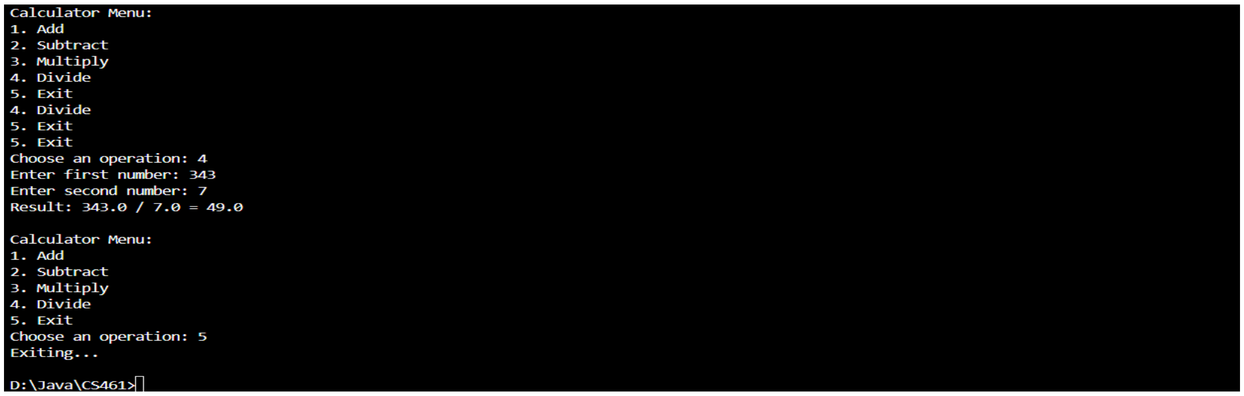


```
D:\Java\CS461>java RMIServer
RMI Server is running...
```

Activating RMIClient. The server asks the client to choose a calculation operation from the menu and enter values accordingly.



```
D:\Java\CS461>java RMIClient

Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Choose an operation: 1
Enter first number: 34
Enter second number: 45
Result: 34.0 + 45.0 = 79.0

Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Choose an operation: 2
Enter first number: 675
Enter second number: 389
Result: 675.0 - 389.0 = 286.0
```



```
Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
4. Divide
5. Exit
5. Exit
Choose an operation: 4
Enter first number: 343
Enter second number: 7
Result: 343.0 / 7.0 = 49.0

Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Choose an operation: 5
Exiting...

D:\Java\CS461>
```

The calculation is carried out by the server and the result is displayed in the client side. Upon choosing exit option, client can choose to disconnect from the server.

---END OF LAB ASSIGNMENT---