Remote Method Invocation (RMI) – Cloud-Based Calculator Application

---

1. Aim

To implement a **Remote Method Invocation (RMI)** based calculator application using Java, where the server is deployed on **AWS EC2** and the client invokes methods remotely.

---

2. Objective

- To understand object-oriented distributed systems using RMI
- To implement remote interfaces and remote objects
- To deploy an RMI server on a cloud platform
- To verify remote method invocation from a client system

---

3. System Requirements

Hardware

- Computer with minimum 4 GB RAM
- Internet connection

Software

- Operating System: Ubuntu (Server), Windows / Kali Linux (Client)
- Programming Language: Java
- Cloud Platform: AWS EC2
- Tools: OpenJDK

---

4. RMI Architecture

- RMI follows an **object-oriented client–server model**
- A remote interface defines the methods
- The server implements the interface and registers the remote object
- The client looks up the remote object using the RMI registry
- Remote methods are invoked using stubs

5. RMI Implementation Details

5.1 Remote Interface

The remote interface defines the following methods:

- add()
- sub()
- mul()
- div()

5.2 Working Principle

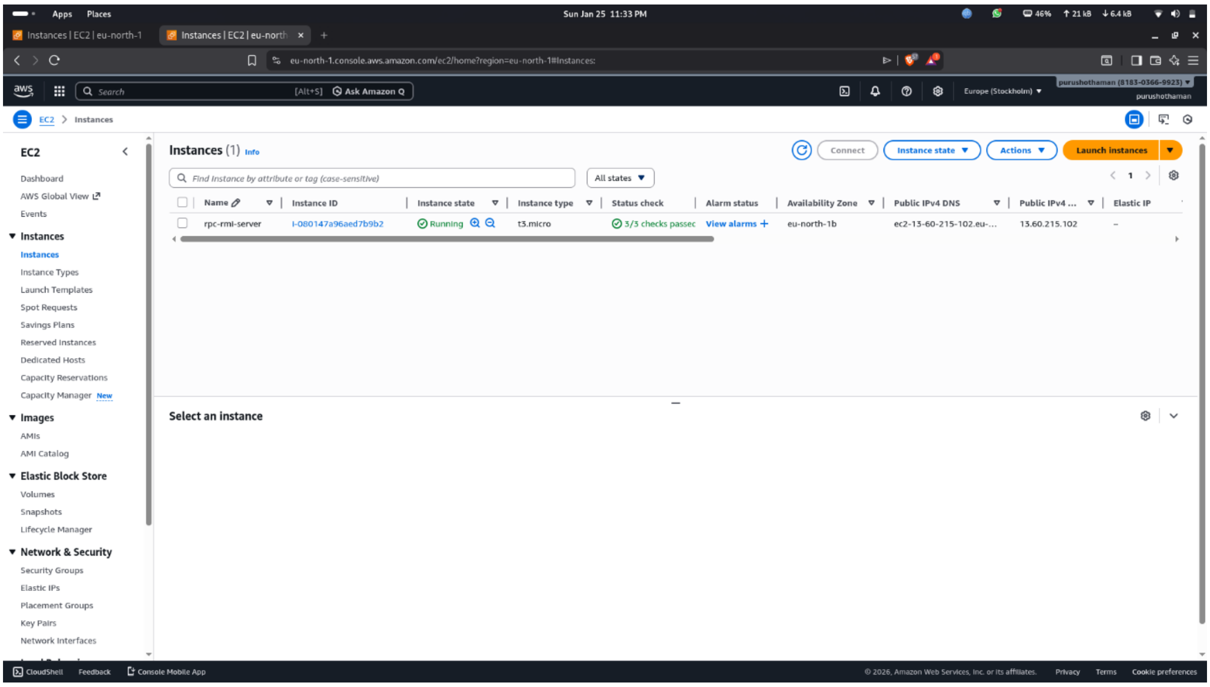
1. Remote interface is created
2. Server implements the remote interface
3. Server registers the object with the RMI registry
4. Client looks up the remote object
5. Client invokes remote methods

6. Cloud Deployment

- RMI server is hosted on **AWS EC2**
- Public IP address is used for registry lookup
- Required ports:
  - 1099 (RMI Registry)
  - 1024–65535 (Dynamic RMI ports)

Attach the following screenshots:

- EC2 instance running
- Security group inbound rules
- RMI server execution

- 

Instances (1)

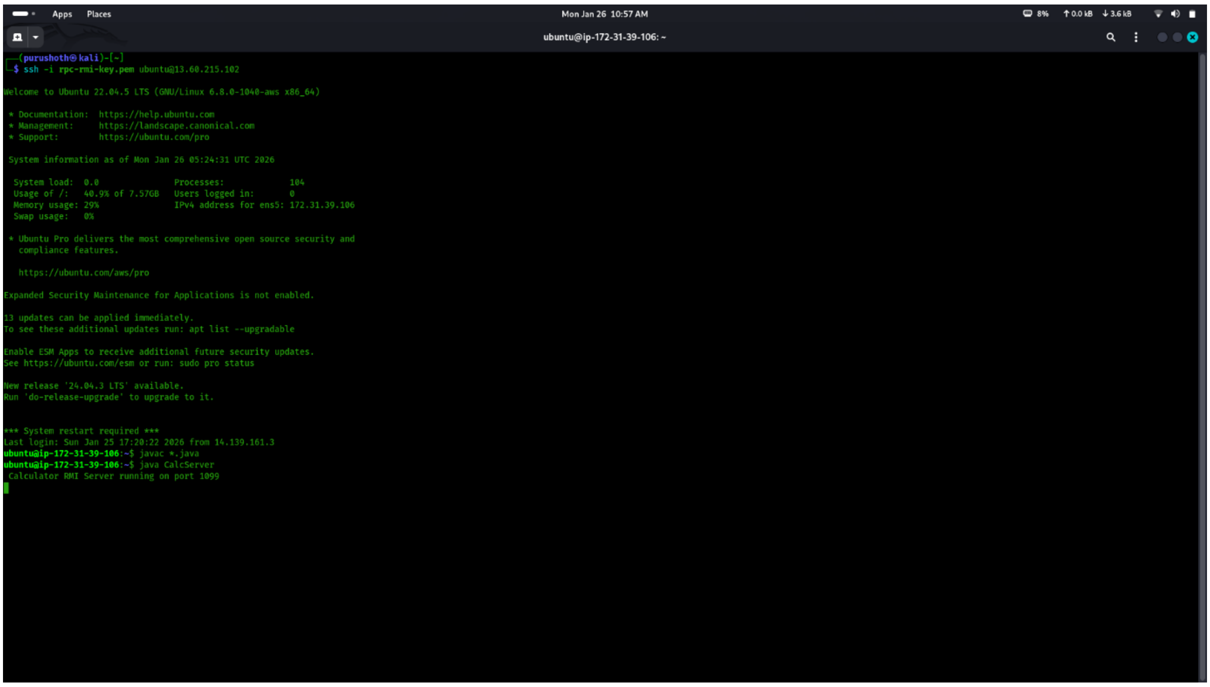
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
rpc-rmi-server	i-080147a96aed7b9b2	Running	t3.micro	3/3 checks passec	View alarms +	eu-north-1b	ec2-13-60-215-102.eu-...	13.60.215.102	–

7. Error Handling

- Remote exceptions are handled using try–catch blocks
- Division by zero is handled safely
- Network connection failures are managed

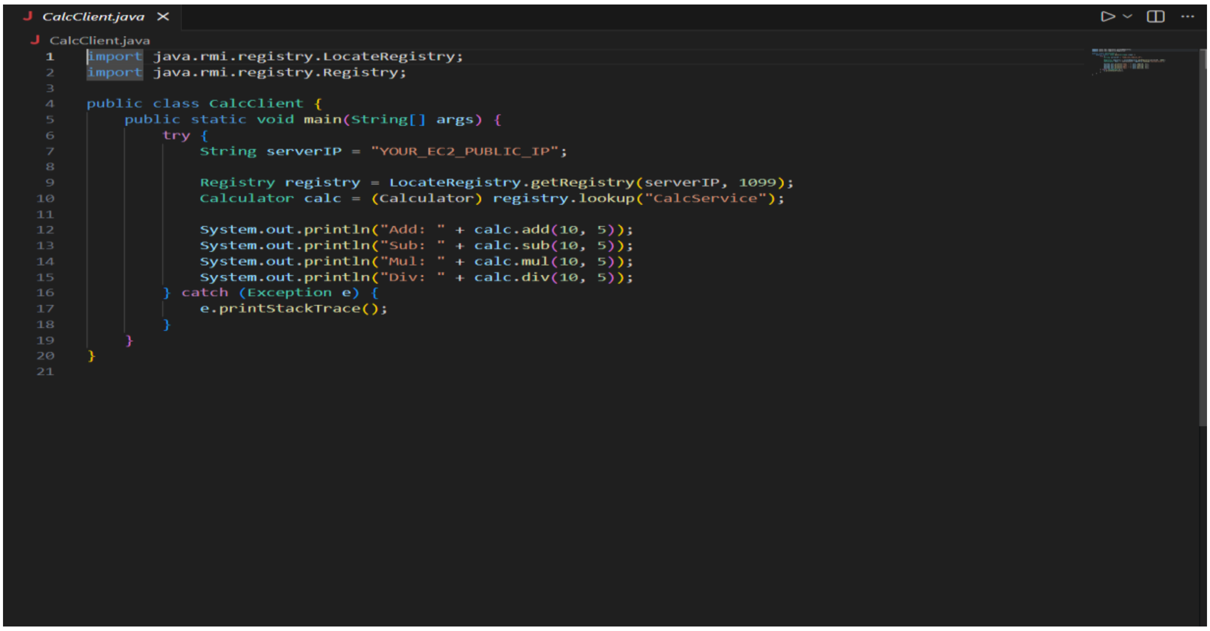
8. Output

- **RMI server running on AWS EC2**

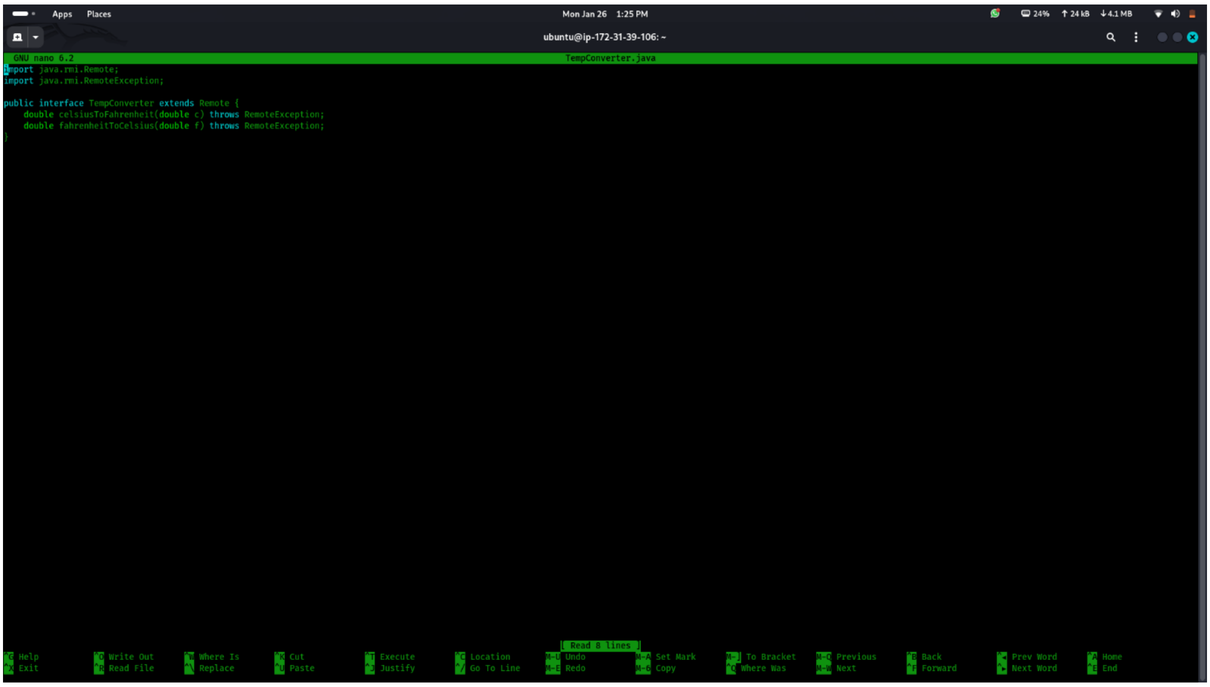


```
┌──(purushoth㉿kali)-[~]
└─$ ssh -i rpc-rmi-key.pem ubuntu@13.60.215.102

Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1040-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Mon Jan 26 05:24:31 UTC 2026

  System load:  0.0               Processes:             104
  Usage of /:   40.9% of 7.57GB   Users logged in:       0
  Memory usage: 29%               IPv4 address for ens5: 172.31.39.106
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

13 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***
Last login: Sun Jan 25 17:20:22 2026 from 14.139.161.3
ubuntu@ip-172-31-39-106:~$ javac *.java
ubuntu@ip-172-31-39-106:~$ java CalcServer
 Calculator RMI Server running on port 1099
```

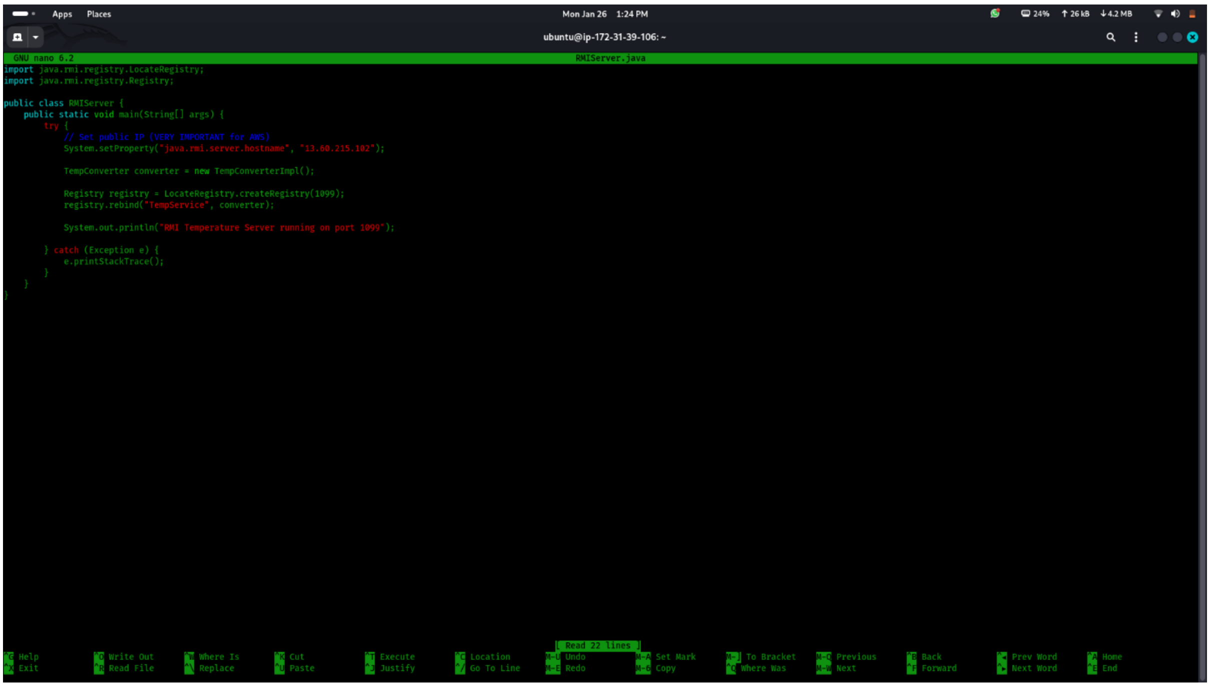
- **Client invoking remote methods**



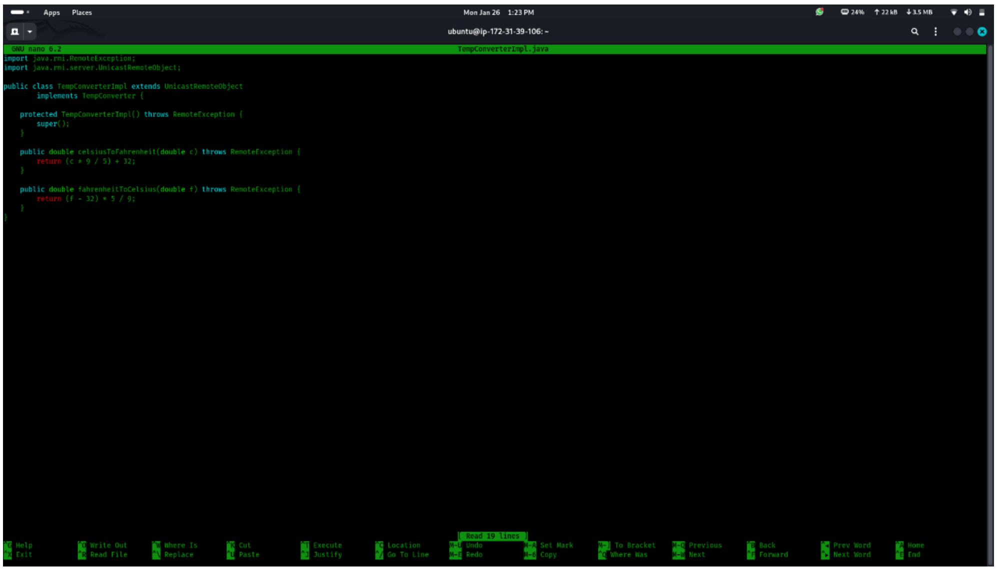
```java
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class CalcClient {
    public static void main(String[] args) {
        try {
            String serverIP = "YOUR_EC2_PUBLIC_IP";

            Registry registry = LocateRegistry.getRegistry(serverIP, 1099);
            Calculator calc = (Calculator) registry.lookup("CalcService");

            System.out.println("Add: " + calc.add(10, 5));
            System.out.println("Sub: " + calc.sub(10, 5));
            System.out.println("Mul: " + calc.mul(10, 5));
            System.out.println("Div: " + calc.div(10, 5));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```



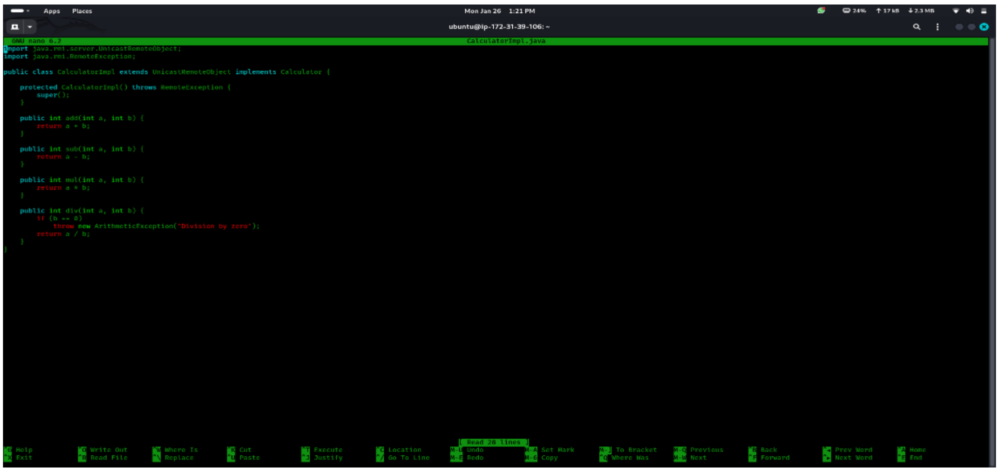
```
  GNU nano 6.2                                                        TempConverter.java
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface TempConverter extends Remote {
    double celsiusToFahrenheit(double c) throws RemoteException;
    double fahrenheitToCelsius(double f) throws RemoteException;
}

                                                                   [ Read 8 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket M-Q Previous   ^B Back        ^◂ Prev Word   ^A Home
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^Q Where Was   M-W Next       ^F Forward     ^▸ Next Word   ^E End
```



```
  GNU nano 6.2                                                        RMIServer.java
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class RMIServer {
    public static void main(String[] args) {
        try {
            // Set public IP (VERY IMPORTANT for AWS)
            System.setProperty("java.rmi.server.hostname", "13.60.215.102");

            TempConverter converter = new TempConverterImpl();

            Registry registry = LocateRegistry.createRegistry(1099);
            registry.rebind("TempService", converter);

            System.out.println("RMI Temperature Server running on port 1099");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

                                                                   [ Read 22 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket M-Q Previous   ^B Back        ^◂ Prev Word   ^A Home
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^Q Where Was   M-W Next       ^F Forward     ^▸ Next Word   ^E End
```

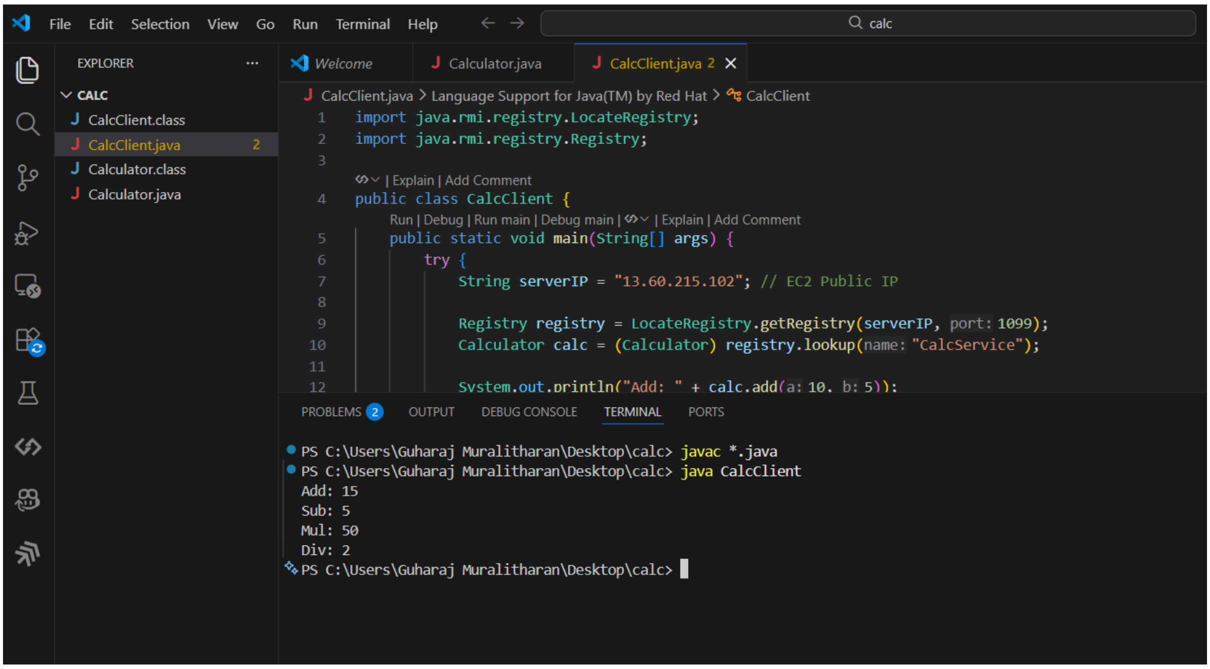


```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class TempConverterImpl extends UnicastRemoteObject
        implements TempConverter {

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

    public double celsiusToFahrenheit(double c) throws RemoteException {
        return (c * 9 / 5) + 32;
    }

    public double fahrenheitToCelsius(double f) throws RemoteException {
        return (f - 32) * 5 / 9;
    }
}
```



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

public class CalculatorImpl extends UnicastRemoteObject implements Calculator {

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

    public int add(int a, int b) {
        return a + b;
    }

    public int sub(int a, int b) {
        return a - b;
    }

    public int mul(int a, int b) {
        return a * b;
    }

    public int div(int a, int b) {
        if (b == 0)
            throw new ArithmeticException("Division by zero");
        return a / b;
    }
}
```

- **Correct calculator results displayed**



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

public class CalcClient {
    public static void main(String[] args) {
        try {
            String serverIP = "13.60.215.102"; // EC2 Public IP

            Registry registry = LocateRegistry.getRegistry(serverIP, port: 1099);
            Calculator calc = (Calculator) registry.lookup(name: "CalcService");

            System.out.println("Add: " + calc.add(a: 10, b: 5));
```

```
PS C:\Users\Guharaj Muralitharan\Desktop\calc> javac *.java
PS C:\Users\Guharaj Muralitharan\Desktop\calc> java CalcClient
Add: 15
Sub: 5
Mul: 50
Div: 2
PS C:\Users\Guharaj Muralitharan\Desktop\calc>
```

9. Result

The RMI-based calculator application was successfully implemented and deployed in a cloud environment. The client accessed server-side objects and obtained correct computation results.

10. Conclusion

This experiment demonstrated Java RMI as an object-oriented approach to distributed computing. Hosting the RMI server on AWS EC2 enhanced understanding of real-world distributed application deployment.