Client.java –

```
import java.rmi.Naming;


public class Client {
   public static void main(String[] args) {
       try {
           String url = "//localhost:1098/ArithmeticService"; // Use the same port as specified in the
server
           Server arithmeticService = (Server) Naming.lookup(url);


           int a = 10;
           int b = 5;


           System.out.println("Adding: " + a + " + " + b + " = " + arithmeticService.add(a, b));
           System.out.println("Subtracting: " + a + " - " + b + " = " + arithmeticService.subtract(a, b));
           System.out.println("Multiplying: " + a + " * " + b + " = " + arithmeticService.multiply(a, b));
           System.out.println("Dividing: " + a + " / " + b + " = " + arithmeticService.divide(a, b));
       } catch (Exception e) {
           e.printStackTrace();
       }
   }
}
```

Server.java-

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


public interface Server extends Remote {
   int add(int a, int b) throws RemoteException;
```

```
    int subtract(int a, int b) throws RemoteException;

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

    double divide(int a, int b) throws RemoteException;
}
```

ServerImpl.java –

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

public class ServerImpl extends UnicastRemoteObject implements Server {
    public ServerImpl() throws RemoteException {
        super();
    }

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

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

    @Override
    public int multiply(int a, int b) throws RemoteException {
        return a * b;
    }
    @Override
    public double divide(int a, int b) throws RemoteException {
        if (b == 0) {
            throw new RemoteException("Cannot divide by zero");
        }
        return (double) a / b;
    }

    public static void main(String[] args) {
        try {
            int registryPort = 1098; // Use a different port
            Server arithmeticService = new ServerImpl();
            java.rmi.registry.LocateRegistry.createRegistry(registryPort);
            java.rmi.Naming.rebind("//localhost:" + registryPort + "/ArithmeticService",
arithmeticService);
            System.out.println("ArithmeticService is ready on port " + registryPort + ".");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```