Server implementation

```
package _final;

import java.lang.*;
import java.util.Properties;
import java.io.*;

import org.omg.CORBA.*;
import org.omg.CORBA.Object;
import org.omg.CosNaming.*;

public class CORBACalcImpl extends _CalcImplBase {
        // dummyServer keeps the server in running state
        private java.lang.Object dummySync;

        // Use the integer value of the operator to get the requested operation
        public int calculate(int opcode, int op1, int op2) {
                // Result to be returned
                int result = 0;

                // switch block to do the calculation
                switch(opcode) {
                case '+': result = op1 + op2; break;
                case '-': result = op1 - op2; break;
                case '*': result = op1 * op2; break;
                case '/': result = op1 / op2; break;
                case '%': result = op1 % op2; break;
                }

                return result;
        }

        public void setDummySync(java.lang.Object dummySync) {
                this.dummySync = dummySync;
        }

        // dummySync revokes waiting state to let server main() proceed
        public int exit() {
                System.out.println("CalcServer Exiting ...");
                synchronized(dummySync) {
                        dummySync.notify();
                }
                return 0;
        }

        public static void main(String[] args) {
                try {
                        // create and initialize the ORB
                        System.out.println("Initializing ORB ...");
                        Properties props = new Properties();
                        props.put("org.omg.CORBA.ORBInitialPort", "1050");
                        ORB orb = ORB.init(args, props);
```

```
                    // create a Calculator and register it with the ORB
                    System.out.println("Connecting solver to ORB...");
                    CORBACalcImpl calculator = new CORBACalcImpl( );
                    orb.connect(calculator);

                    // get the Naming service reference from the ORB
                    System.out.println("Getting reference to Naming Service..");
                    org.omg.CORBA.Object ncObj = orb.resolve_initial_references("NameService");
                    NamingContext ncRef = NamingContextHelper.narrow(ncObj);

                    // bind the Calculator object to a name
                    System.out.println("Registering Calculator with Naming service.");
                    NameComponent comp = new NameComponent("Calculator", "");
                    NameComponent path[] = {comp};
                    // Creates a binding of the name "Calculator" and the object "calculator"
                    ncRef.bind(path, calculator);

                    // wait for client request
                    System.out.println("Waiting for clients...");

                    // dummySyc waits to halt the proceeding of the program and
                    // lets the server handle client request
                    java.lang.Object dummySync = new java.lang.Object();
                    calculator.setDummySync(dummySync);
                    synchronized(dummySync) {
                              dummySync.wait( );
                    }

                    // remove name binding of calculator and shut down ORB
                    ncRef.unbind(path);
                    orb.shutdown(false);
                    System.out.println("Server exited");
          }

          catch (Exception e) {
                    System.out.println("Server error: " + e);
                    e.printStackTrace(System.out);
          }


     }

}
```

Client implementation

```
package _final;

import java.util.*;

import org.omg.CORBA.*;
import org.omg.CosNaming.*;

public class CORBACalcClient {

    public static void main(String[] args) {
        try {
            // create an ORB
            Properties props = new Properties();
            props.put("org.omg.CORBA.ORBInitialPort", "1050");
            ORB orb = ORB.init (args, props);

            // get a reference to the Naming service object
            org.omg.CORBA.Object ncObj = orb.resolve_initial_references("NameService");
            NamingContext nc = NamingContextHelper.narrow (ncObj );

            // get a reference to the calculator object on the remote host
            NameComponent comp = new NameComponent("Calculator", "");
            NameComponent path[] = {comp};
            org.omg.CORBA.Object calcObj= nc.resolve(path);
            Calc calculator = CalcHelper.narrow(calcObj);

            // ask the user to enter a string for calculation
            System.out.println("Enter operator, operand1 and operand2, "
                    + "separated by whitespaces, eg. \"+ 23 15\". \nTo exit, enter \"Exit\".");

            // use a while loop to keep sending requests to server
            Scanner input = new Scanner(System.in);
            while(true) {
                System.out.print("Client: ");
                // read string input
                String s = input.nextLine();

                // if s is "Exit", call exit() on server and client exits
                if(s.equalsIgnoreCase("exit")) {
                    System.out.println("Server: I am out");
                    calculator.exit();
                    break;
                }

                // check validity of input string and parse
                int[] intArray = new int[3];
                int result;
                if(isValid(s, intArray)) {
                        result = calculator.calculate(intArray[0], intArray[1], intArray[2]);
                        System.out.println("Server: " + result);
                }
                else
```

```
                                        System.out.println("Invalid input. Enter again.");
                        }

                }

                catch (Exception e ) {
                        System.out.println("Client error: " + e);
                        e.printStackTrace(System.out);
                        }

                System.out.println("Client exited");
        }

        // transform the input string into an array of three integers
        public static boolean isValid(String s, int[] intArray) {
                String[] tokens = s.split(" ");

                // check length of the string array
                if(tokens.length != 3)
                        return false;

                // check if operator is valid
                if(tokens[0].length() != 1) return false;
                if(!"+-*/%".contains(tokens[0])) return false;
                intArray[0] = tokens[0].charAt(0); // take the int value of the operator char

                // check if the operands are valid
                try {
                        intArray[1] = Integer.parseInt(tokens[1]);
                        intArray[2] = Integer.parseInt(tokens[2]);
                }
                catch(NumberFormatException nfe) {
                        return false;
                }

                return true;
        }

}
```