Lu Peng

Interface

```
import java.rmi.*;

public interface Calc extends Remote {
	public int calculate(int opcode, int op1, int op2) throws RemoteException;
	public int exit() throws RemoteException;
}
```

Server implementation

```
import java.rmi.*;
import java.rmi.server.*;


public class CalcServer extends UnicastRemoteObject implements Calc {
	// dummyServer keeps the server in running state
	private java.lang.Object dummySync;

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

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

	public static void main(String[] args) {
		// creation and installation a security manager is not needed,
		// because no dynamic class loading from the client happens in this case
//		System.setSecurityManager(new RMISecurityManager());

		try {
			// register an instance of CalcServer with the RMI Naming Service
			String name = "Calculator";
			CalcServer calculator = new CalcServer( );
			Naming.rebind(name, calculator);
			System.out.println ("Remote solver is ready...");

			// 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
			Naming.unbind(name);
			// remove calculator from server runtime
```

```
				UnicastRemoteObject.unexportObject(calculator, false);
				System.out.println("Server exited");
				//System.exit(0);
			}

			catch (Exception e) {
				System.out.println("Caught an exception while registering: " + e);
				e.printStackTrace( );
			}

		}

		public int calculate(int opcode, int op1, int op2) throws RemoteException {
			// 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;
		}

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

	}
```

Client implementation

```
import java.rmi.*;
import java.rmi.server.*;
import java.util.Scanner;

public class CalcClient {

	public static void main(String[] args) {
		// create and install a security manager if dynamic class loading needed
		// System.setSecurityManager(new RMISecurityManager());

		// get a remote reference to the CalcServer class
		String name = "/Calculator"; // if not on same machine, use "rmi://name_of_remote_host(IP
address)/Service"
		Calc calculator = null;

		try {
			calculator = (Calc) Naming.lookup(name);
		}
		catch(Exception e) {
			System.out.println ("Caught an exception while looking up the server:");
			e.printStackTrace();
			System.exit(0);
		}

		// 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
		try {
			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.");
				}
				input.close();
			}
			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;
		}

}
```