Lab 3 – CSE 660 – Spring 2011

Erwin Soekianto – Chenwei Xu

This lab consists of 3 parts:

- RPC (Remote Procedure Call)
- Java RMI (Remote Method Invocation)
- Android Application Development (Remote UDT Calculator)

Part 1: RPC

RPC was first implemented at Frontier at 1998.

What is RPC?

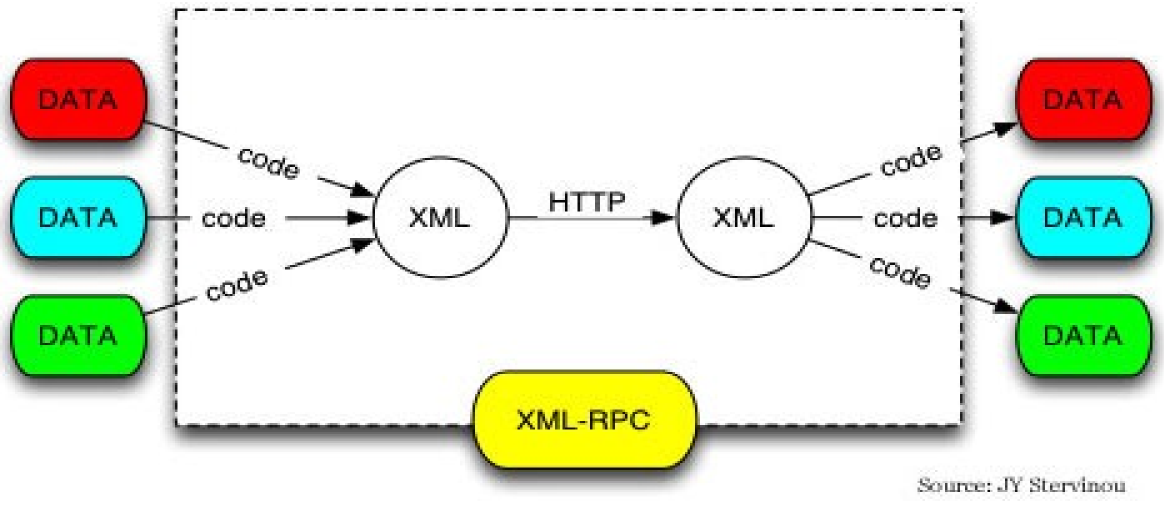
It's a spec and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet. [1]

It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned. [1]

RPC is very compatible, which can be implemented with various operating systems, programming languages, dynamic and static environments, open source and commercial, for Perl, Python, Java, Frontier, C/C++, Lisp, PHP, Microsoft .NET, Rebol, Real Basic, Tcl, Delphi, WebObjects and Zope, and more are coming all the time.

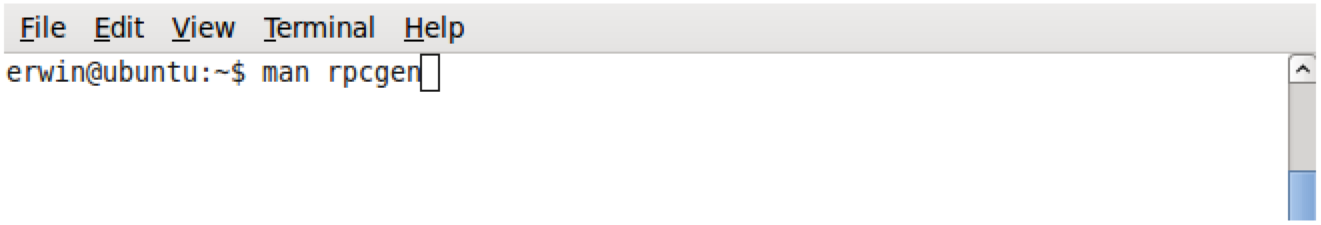
But the one that we are using for this lab is in C and Linux operating system.

And this is how the XML-RPC works in diagram below.



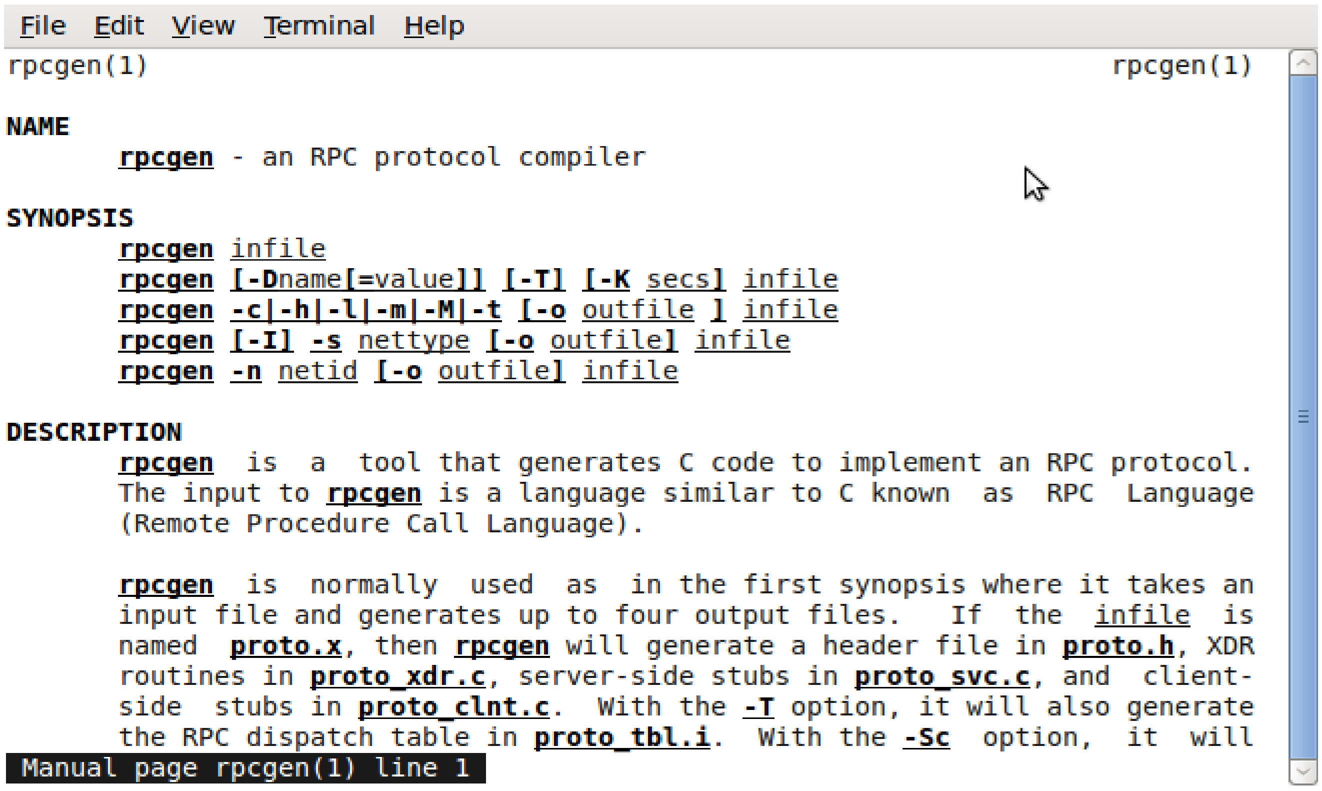
Installation and configuration:

- To check whether you have RPC installed on you machine by using $**man rpcgen** command on your terminal, and most of the modern linux OS have RPC installed by deafult.



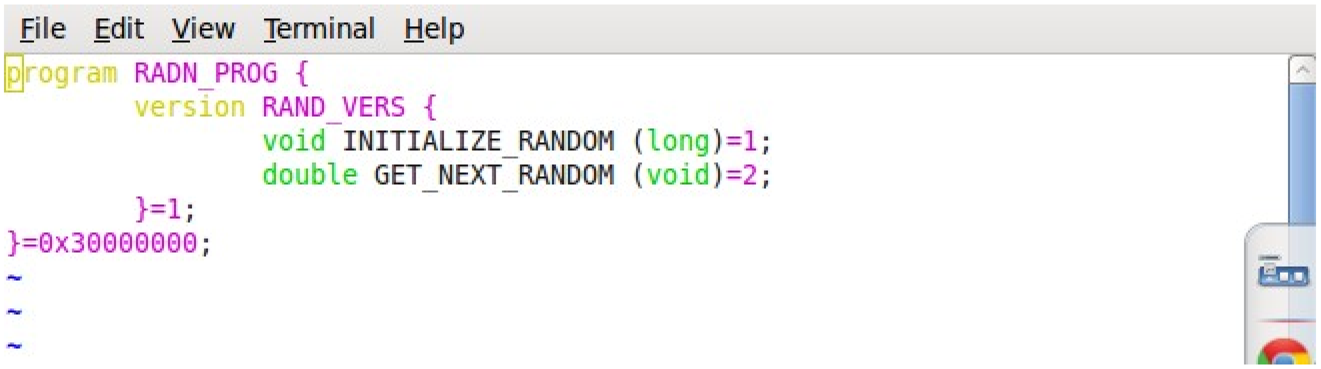
```
erwin@ubuntu:~$ man rpcgen
```

- And if you see the result something like this, that means you already have RPC installed and ready to use and you can also learn from that documentation on how to use RPC
-



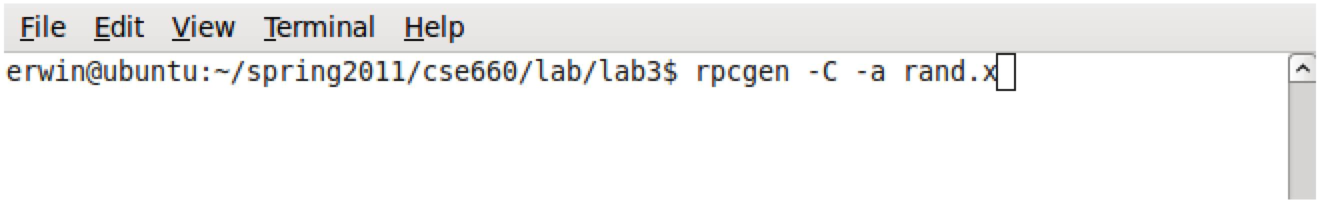
```
rpcgen(1)                                                           rpcgen(1)

NAME
       rpcgen - an RPC protocol compiler

SYNOPSIS
       rpcgen infile
       rpcgen [-Dname[=value]] [-T] [-K secs] infile
       rpcgen -c|-h|-l|-m|-M|-t [-o outfile ] infile
       rpcgen [-I] -s nettype [-o outfile] infile
       rpcgen -n netid [-o outfile] infile

DESCRIPTION
       rpcgen  is  a  tool that generates C code to implement an RPC protocol.
       The input to rpcgen is a language similar to C known  as  RPC  Language
       (Remote Procedure Call Language).

       rpcgen  is  normally  used  as  in the first synopsis where it takes an
       input file and generates up to four output files.   If  the  infile  is
       named  proto.x, then rpcgen will generate a header file in proto.h, XDR
       routines in proto_xdr.c, server-side stubs in proto_svc.c, and  client-
       side  stubs in proto_clnt.c.  With the -T option, it will also generate
       the RPC dispatch table in proto_tbl.i.  With the -Sc  option,  it  will
 Manual page rpcgen(1) line 1
```

- And create a new file name name rand.x with the following content.

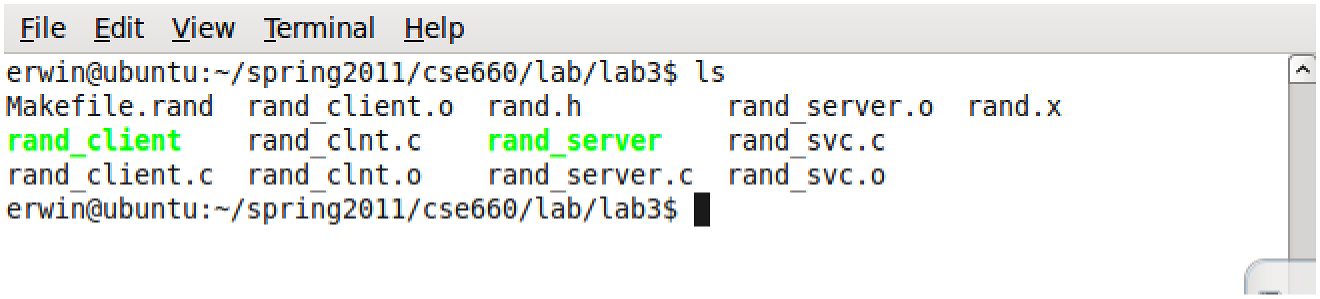


```
program RADN_PROG {
        version RAND_VERS {
                void INITIALIZE_RANDOM (long)=1;
                double GET_NEXT_RANDOM (void)=2;
        }=1;
}=0x30000000;
~
~
~
```

- And generated the program template by typing this command $**rpcgen -C -a rand.x,**
  You should have generated the files:
  Makefile.rand, rand_clnt.c, rand_server.c, rand_client.c, rand.h, rand_svc.c



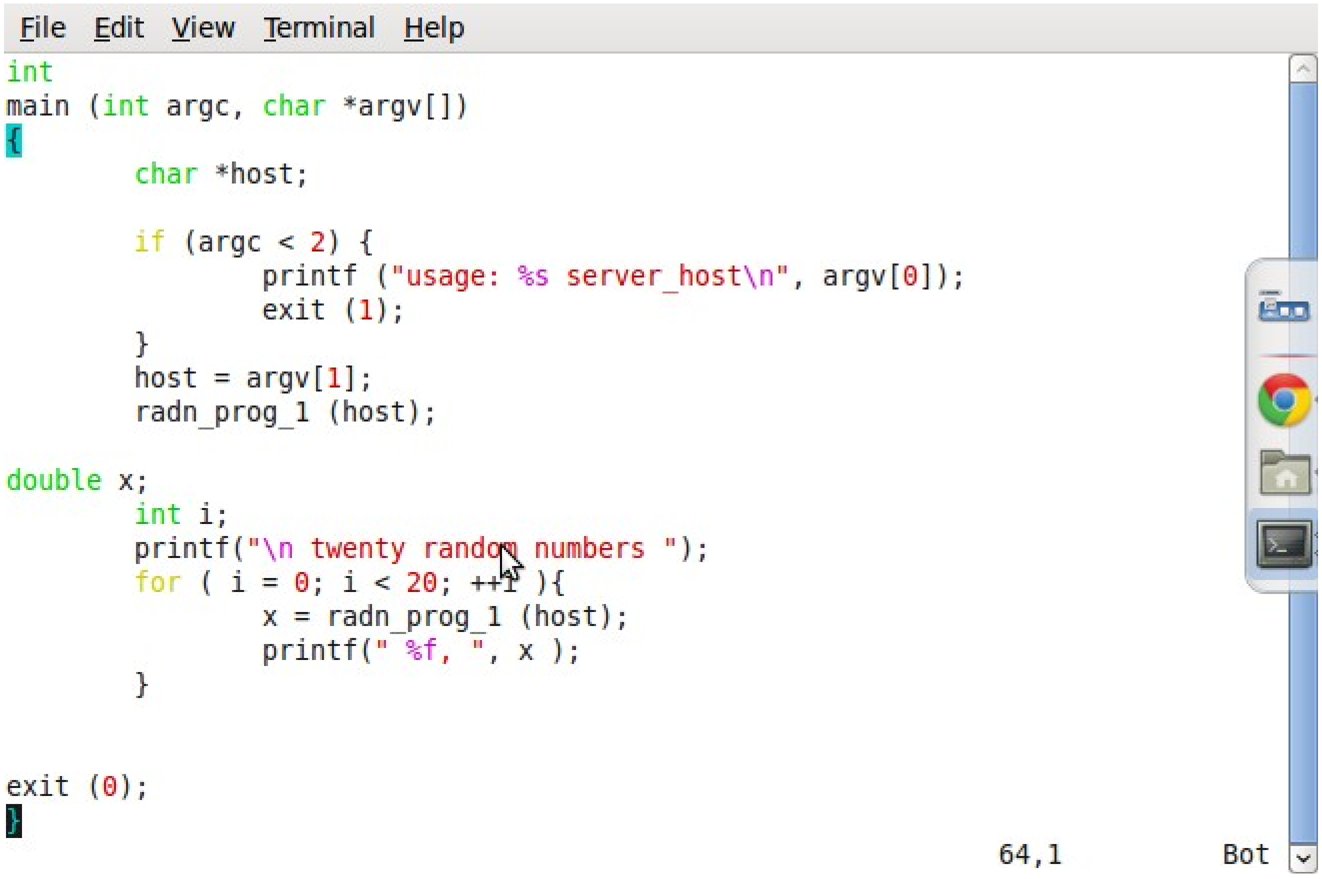
```
erwin@ubuntu:~/spring2011/cse660/lab/lab3$ rpcgen -C -a rand.x
```



```
erwin@ubuntu:~/spring2011/cse660/lab/lab3$ ls
Makefile.rand  rand_client.o  rand.h         rand_server.o  rand.x
rand_client    rand_clnt.c    rand_server    rand_svc.c
rand_client.c  rand_clnt.o    rand_server.c  rand_svc.o
erwin@ubuntu:~/spring2011/cse660/lab/lab3$
```

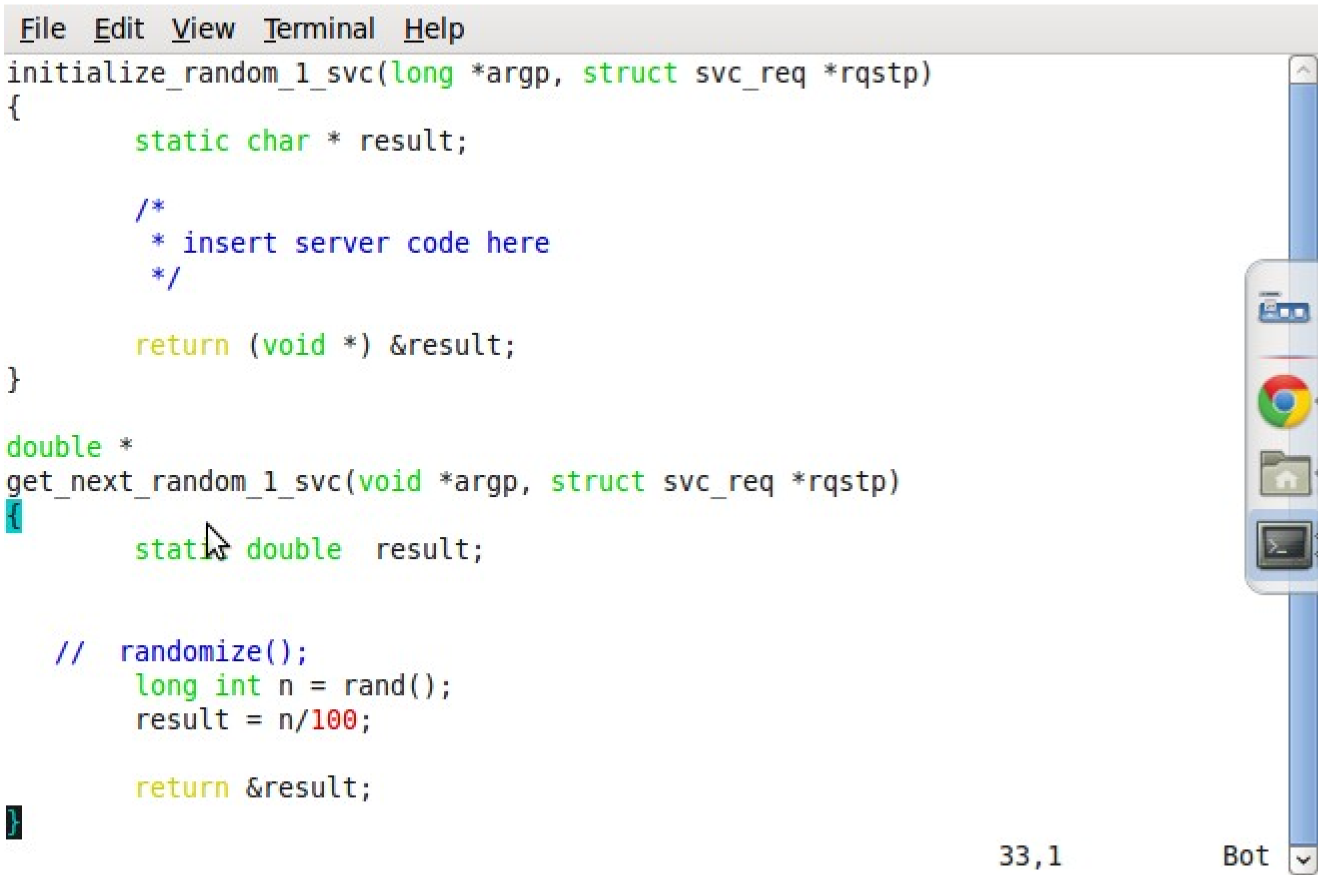
- After studying the generated files, we can edit that so that I would generate our own random numbers as assigned for this lab. And here the code from each file.

Client Code



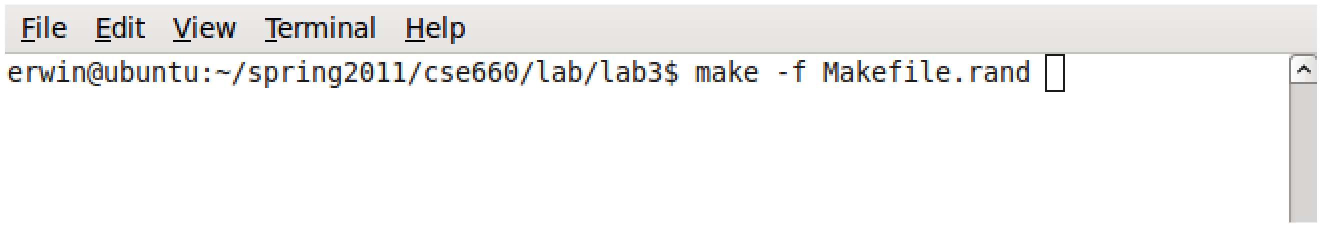
```
int
main (int argc, char *argv[])
{
        char *host;

        if (argc < 2) {
                printf ("usage: %s server_host\n", argv[0]);
                exit (1);
        }
        host = argv[1];
        radn_prog_1 (host);

double x;
        int i;
        printf("\n twenty random numbers ");
        for ( i = 0; i < 20; ++i ){
                x = radn_prog_1 (host);
                printf(" %f, ", x );
        }


exit (0);
}
                                                                  64,1          Bot
```

Server code



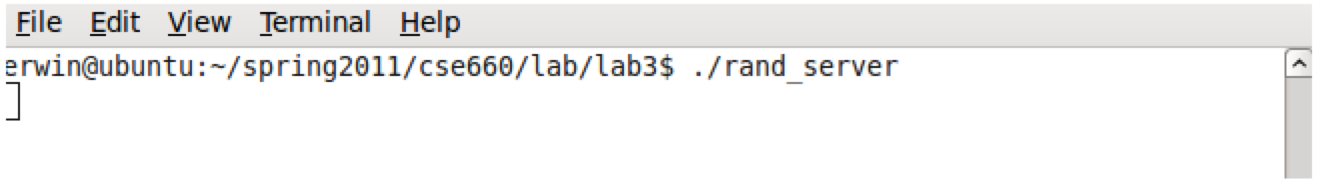
```
initialize_random_1_svc(long *argp, struct svc_req *rqstp)
{
        static char * result;

        /*
         * insert server code here
         */

        return (void *) &result;
}

double *
get_next_random_1_svc(void *argp, struct svc_req *rqstp)
{
        static double  result;


   //  randomize();
        long int n = rand();
        result = n/100;

        return &result;
}
                                                              33,1          Bot
```

– After that, we can compile it with $**make -f Makefile.rand**



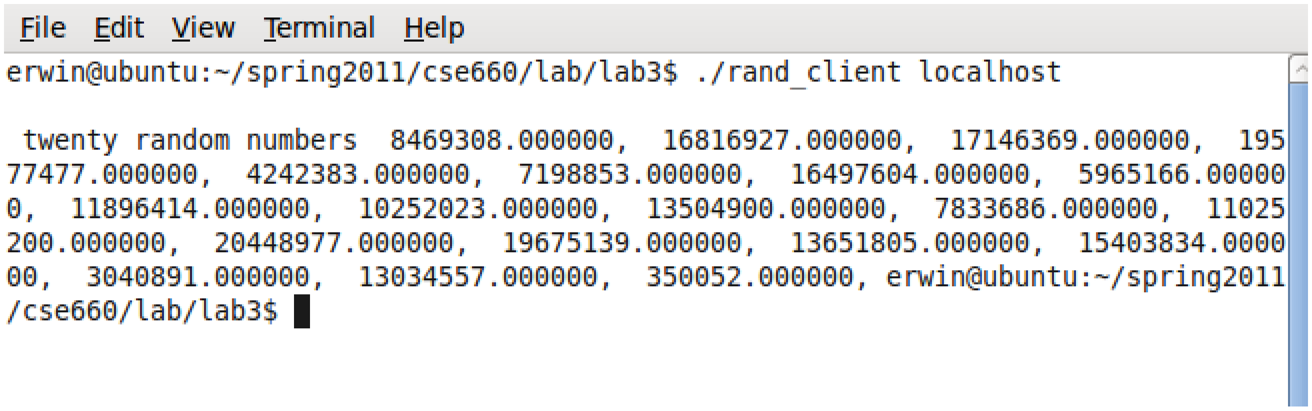
```
erwin@ubuntu:~/spring2011/cse660/lab/lab3$ make -f Makefile.rand
```

– and start the server by $./rand_server



```
erwin@ubuntu:~/spring2011/cse660/lab/lab3$ ./rand_server
```

- Open another terminal or from other computer, start the client by $./rand_client localhost (or server IP address)



```
erwin@ubuntu:~/spring2011/cse660/lab/lab3$ ./rand_client localhost

 twenty random numbers  8469308.000000,  16816927.000000,  17146369.000000,  195
77477.000000,  4242383.000000,  7198853.000000,  16497604.000000,  5965166.00000
0,  11896414.000000,  10252023.000000,  13504900.000000,  7833686.000000,  11025
200.000000,  20448977.000000,  19675139.000000,  13651805.000000,  15403834.0000
00,  3040891.000000,  13034557.000000,  350052.000000, erwin@ubuntu:~/spring2011
/cse660/lab/lab3$
```

Note:

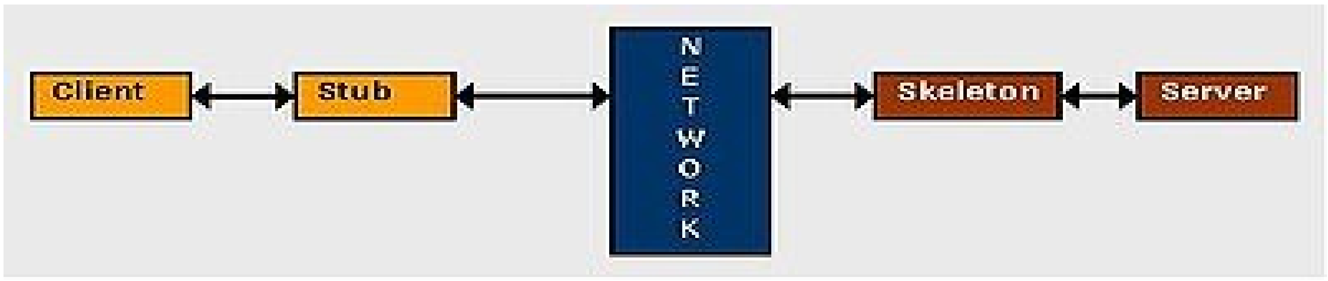
- Make sure you are in the same subnet mask when you do that remotely.
- Make sure to stop other server or connection, it could disturb the connection

Part 2: Java RMI (Remote Method Invocation)

Java RMI is a Java application programming interface that performs the object-oriented equivalent of remote procedure call (RPC).

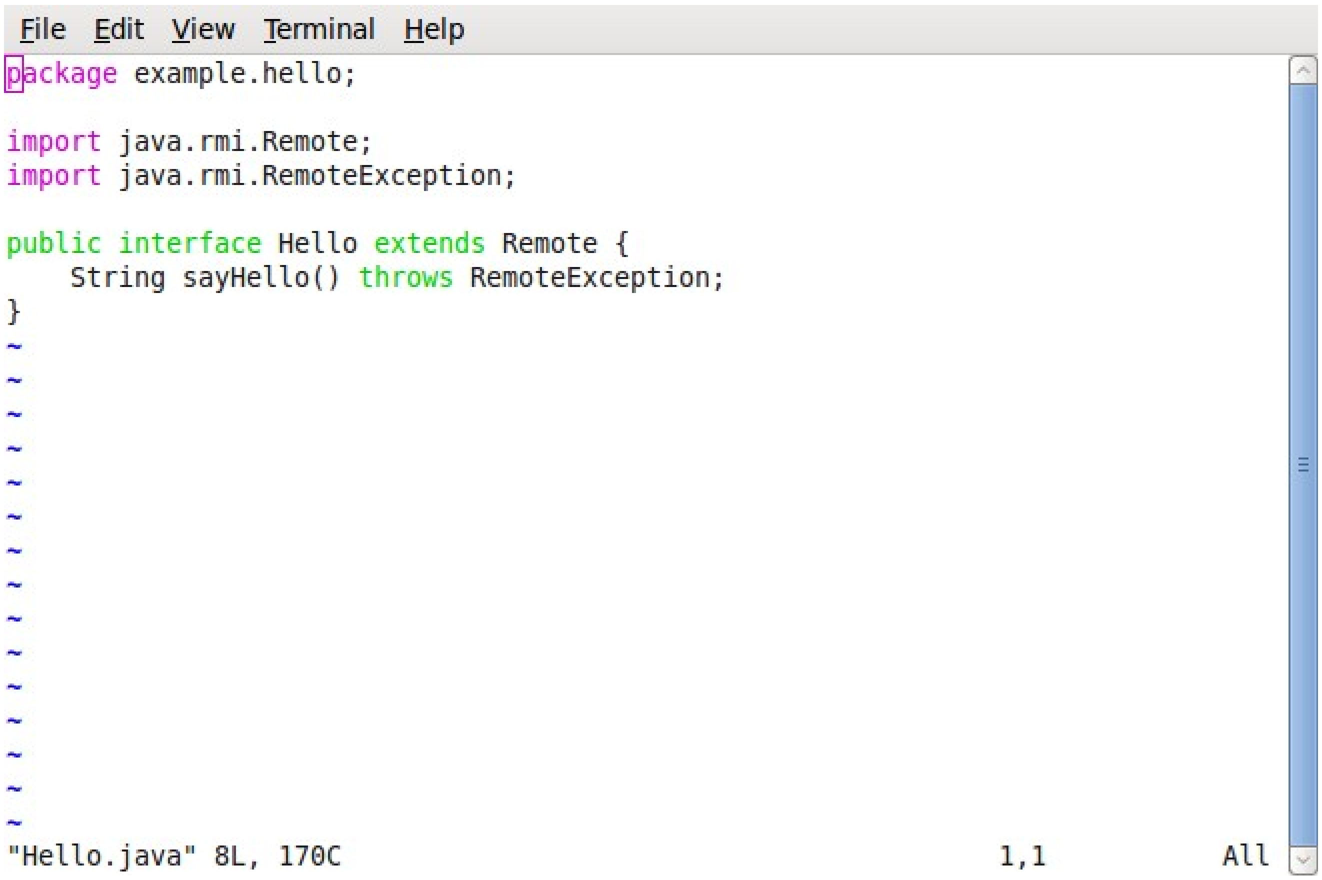
The original implementation depends on Java Virtual Machine (JVM) class representation mechanisms and it thus only supports making calls from one JVM to another. The protocol underlying this Java-only implementation is known as Java Remote Method Protocol (JRMP)

This is the diagram how Java RMI works.

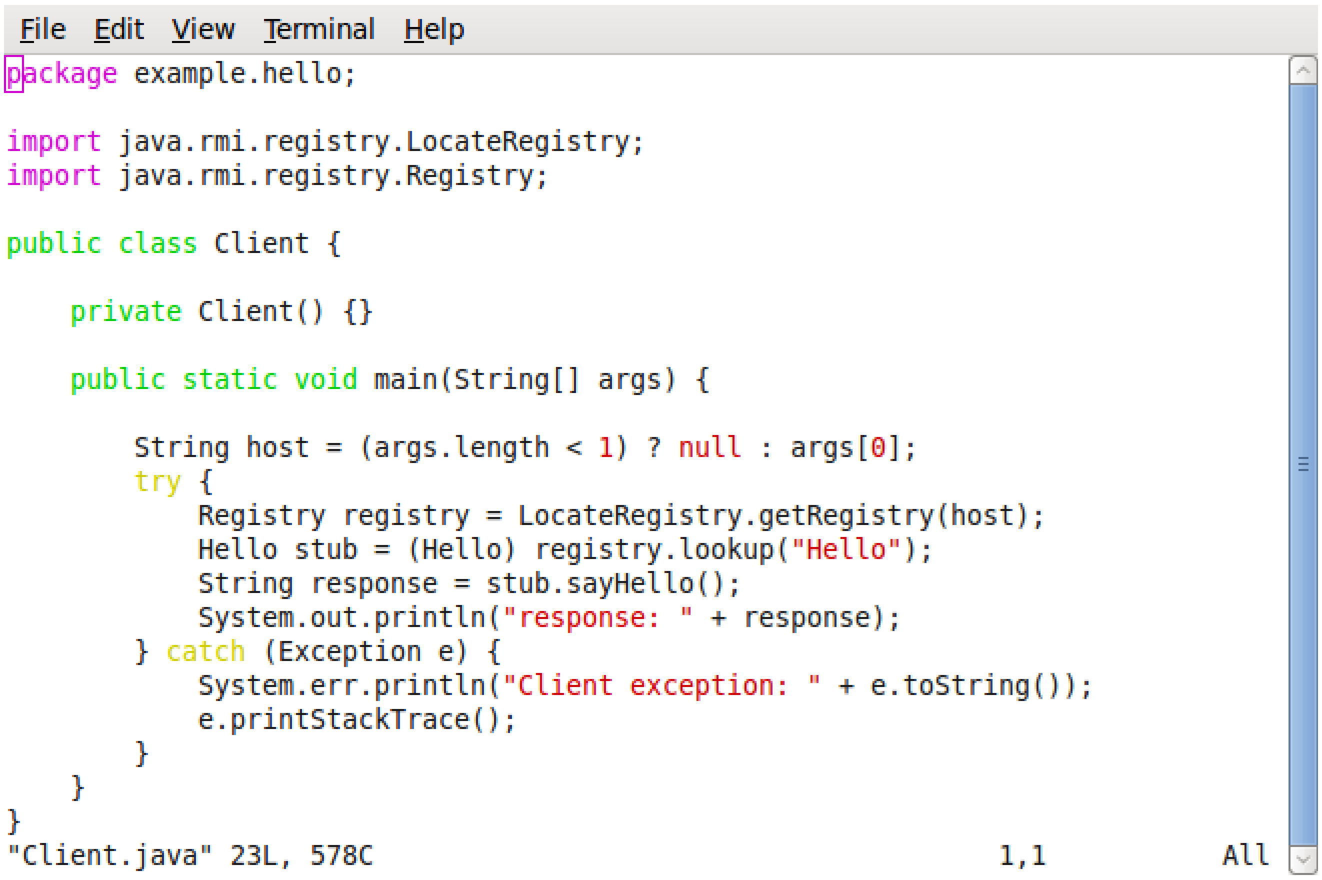


Instruction:

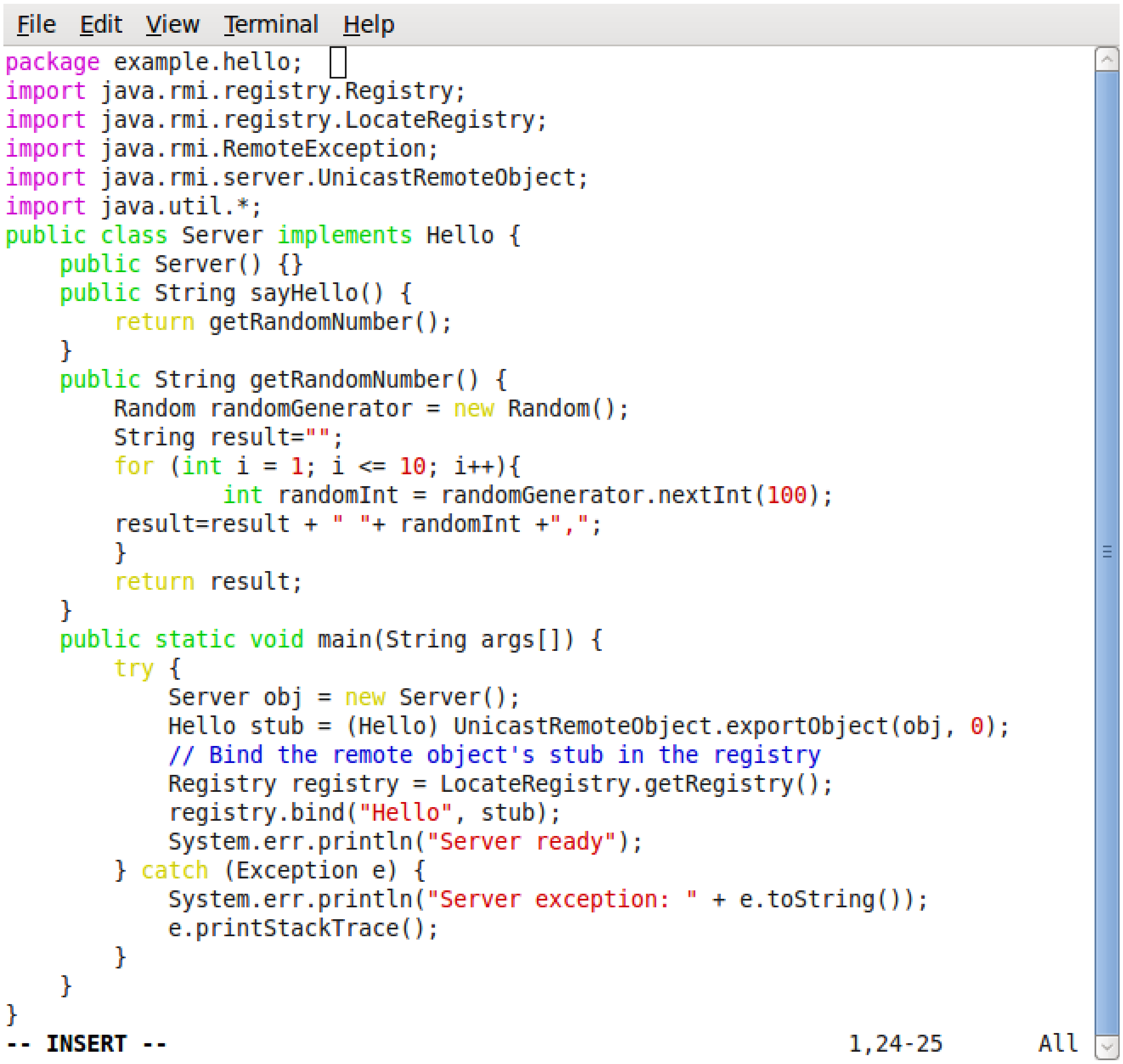
- Of course we need to have Java installed in our computer, we recommend to get the latest version. (As of today, June 2011, it's version 6.26)
- First, we need to create these three file: Hello.java , Server.java, and Client.java



```
package example.hello;

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

public interface Hello extends Remote {
    String sayHello() throws RemoteException;
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"Hello.java" 8L, 170C                                       1,1           All
```



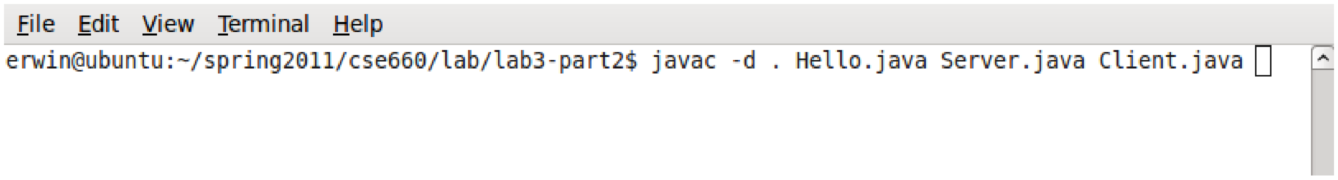
```
package example.hello;

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

public class Client {

    private Client() {}

    public static void main(String[] args) {

        String host = (args.length < 1) ? null : args[0];
        try {
            Registry registry = LocateRegistry.getRegistry(host);
            Hello stub = (Hello) registry.lookup("Hello");
            String response = stub.sayHello();
            System.out.println("response: " + response);
        } catch (Exception e) {
            System.err.println("Client exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
"Client.java" 23L, 578C                                     1,1           All
```



```
package example.hello;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.*;
public class Server implements Hello {
    public Server() {}
    public String sayHello() {
        return getRandomNumber();
    }
    public String getRandomNumber() {
        Random randomGenerator = new Random();
        String result="";
        for (int i = 1; i <= 10; i++){
                int randomInt = randomGenerator.nextInt(100);
        result=result + " "+ randomInt +",";
        }
        return result;
    }
    public static void main(String args[]) {
        try {
            Server obj = new Server();
            Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
            // Bind the remote object's stub in the registry
            Registry registry = LocateRegistry.getRegistry();
            registry.bind("Hello", stub);
            System.err.println("Server ready");
        } catch (Exception e) {
            System.err.println("Server exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
-- INSERT --                                              1,24-25       All
```

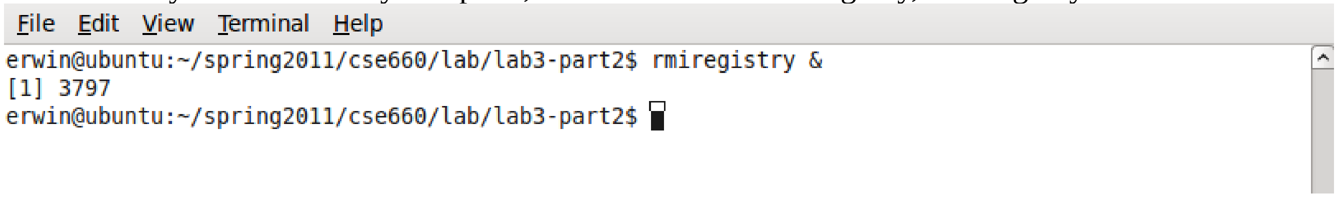
Note : If you notice that we had our custom random generator in the server side and client side.

– Then compile all the file with $javac -d ***destDir*** Hello.java Server.java Client.java



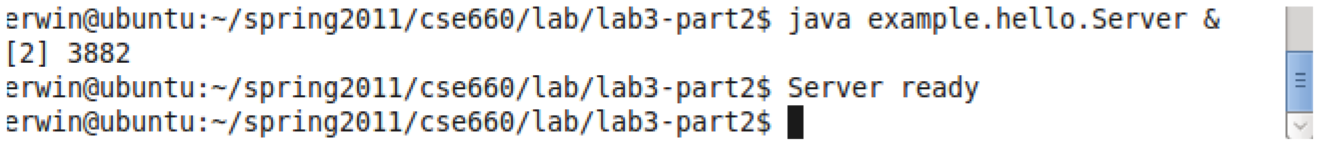
```
erwin@ubuntu:~/spring2011/cse660/lab/lab3-part2$ javac -d . Hello.java Server.java Client.java
```

- If they are successfully compiled, we can start the RMI registry, $rmiregistry &



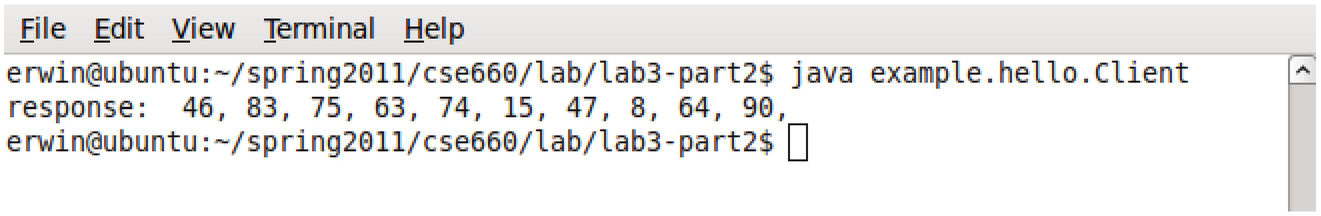
```
erwin@ubuntu:~/spring2011/cse660/lab/lab3-part2$ rmiregistry &
[1] 3797
erwin@ubuntu:~/spring2011/cse660/lab/lab3-part2$
```

- Then start the server, $java example.hello.Server &



```
erwin@ubuntu:~/spring2011/cse660/lab/lab3-part2$ java example.hello.Server &
[2] 3882
erwin@ubuntu:~/spring2011/cse660/lab/lab3-part2$ Server ready
erwin@ubuntu:~/spring2011/cse660/lab/lab3-part2$
```

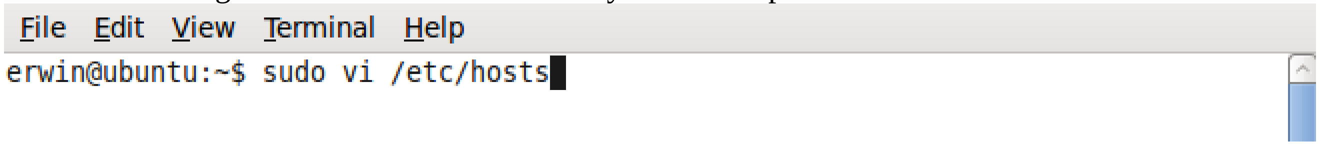
- Open another terminal or from other computer . And start the client, $ java example.hello.Client Ipaddress or empty for local host.



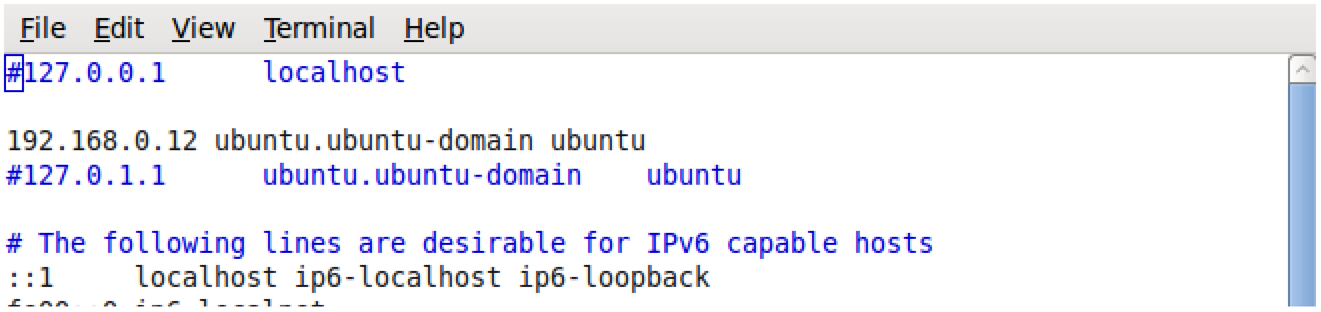
```
erwin@ubuntu:~/spring2011/cse660/lab/lab3-part2$ java example.hello.Client
response:  46, 83, 75, 63, 74, 15, 47, 8, 64, 90,
erwin@ubuntu:~/spring2011/cse660/lab/lab3-part2$
```

Note:

- Make sure you are in same subnet mask
- And port 1099 is open or you need to disable firewall
- And change the host file in /etc/hosts to your server ip address



```
erwin@ubuntu:~$ sudo vi /etc/hosts
```



```
#127.0.0.1      localhost

192.168.0.12 ubuntu.ubuntu-domain ubuntu
#127.0.1.1      ubuntu.ubuntu-domain    ubuntu

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
```

Part 3 Android Application Development

As we all know that open source has come out with the operating system for smart phone and tablet, so called Android. Android used Java as programming language where we need to use Android SDK, until today the latest version of Android is 3.1 using API 12, running in Java 6.

In developing the Application, we need to use the following and make sure you download the right thing.

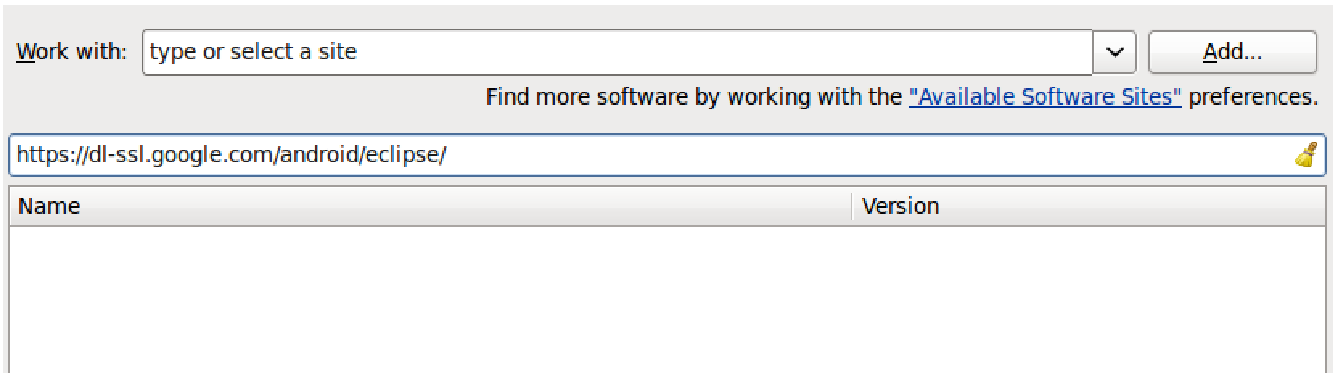
- Eclipse RPC – recommend latest version <http://www.eclipse.org/downloads/>
- Java JDK and JRE (obviously) – latest version <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- Android SDK <http://developer.android.com/sdk/index.html>

Installation guide:

- Install the latest version of java
- unzip Android SDK, put it where it has write and read permission (linux) or not under C: in windows.

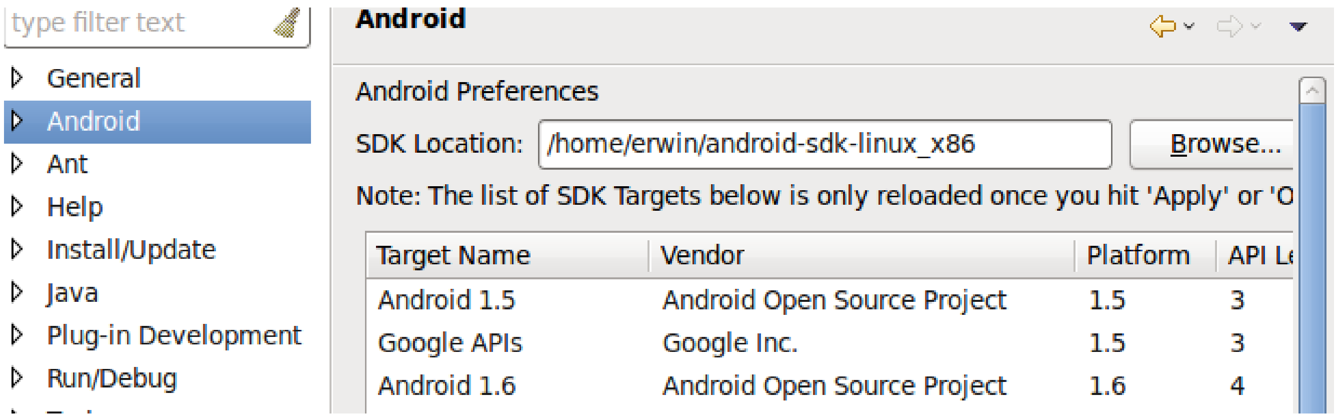
Eclipse Set up

- Start Eclipse, then select Help > Install new software , then this window will come out. And enter "https://dl-ssl.google.com/android/eclipse/" , and click add. This is to install ADT plug-in in eclipse


Work with:
type or select a site
Add...
Find more software by working with the "Available Software Sites" preferences.
Name
Version

https://dl-ssl.google.com/android/eclipse/

- And then we need to configure the ADT plug-in by pointing the to where we save Android SDK files. Go to Window > Preference

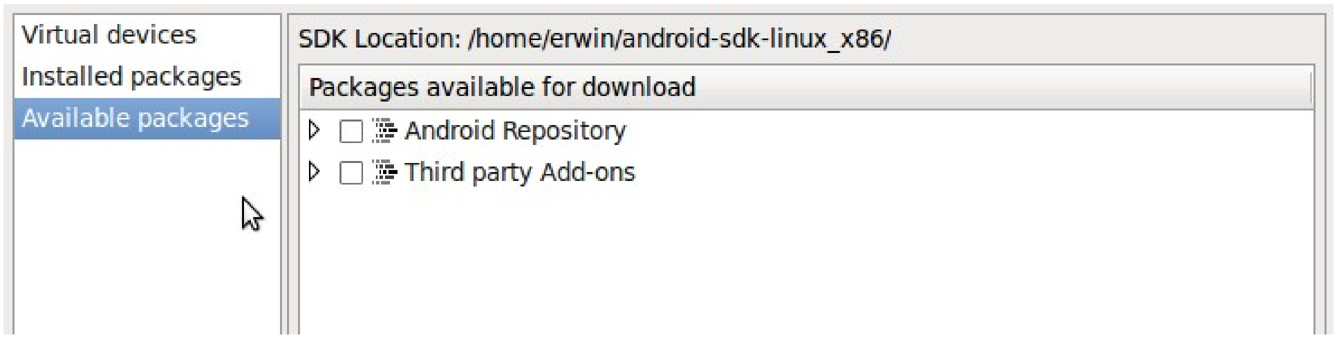


SDK Location: /home/erwin/android-sdk-linux_x86

Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'O

Target Name	Vendor	Platform	API Le
Android 1.5	Android Open Source Project	1.5	3
Google APIs	Google Inc.	1.5	3
Android 1.6	Android Open Source Project	1.6	4

- And then we need to add Android platform and other component. By going to Window > Android SDK and AVD Manager > Available package > Repositories, then we recommend to install everything.



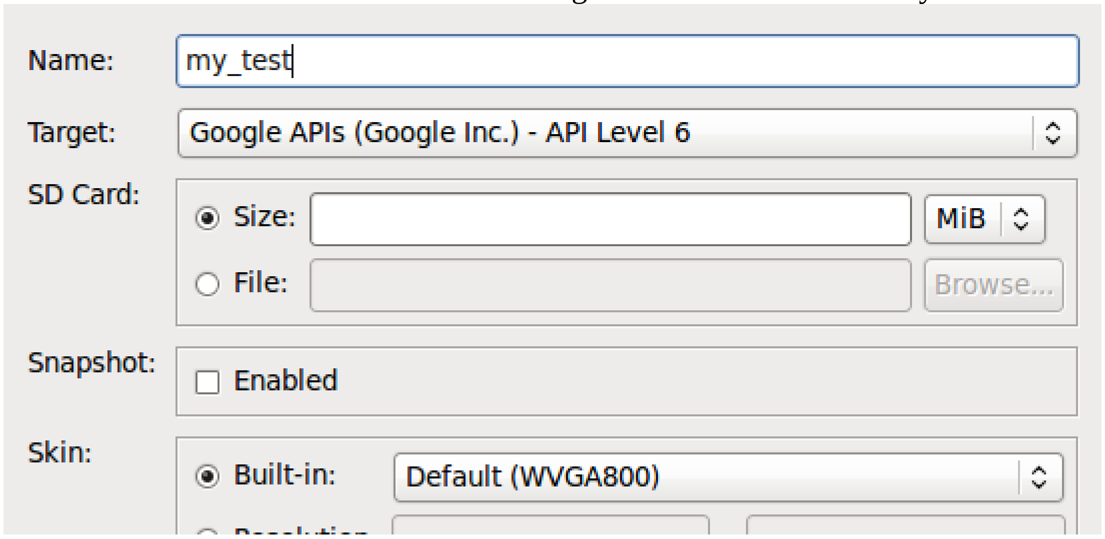
SDK Location: /home/erwin/android-sdk-linux_x86/

Packages available for download
- ☐ Android Repository
- ☐ Third party Add-ons

- Until this step you have successfully installed and configured Android Development tools in your computer

Building test application or Hello World application

- you can follow from this web page too <http://developer.android.com/resources/tutorials/hello-world.html> or follow my tutorial.
- First we need to create new AVD (Android Virtual Device), where the application is going to be displayed or tested on. In Eclipse, go to Window > Android SDK and AVD Manager
- Select Virtual Device in the left panel, then click New, then the New AVD dialog appears.
- Enter the name and select the Android API target and set SD card memory



Name: my_test

Target: Google APIs (Google Inc.) - API Level 6

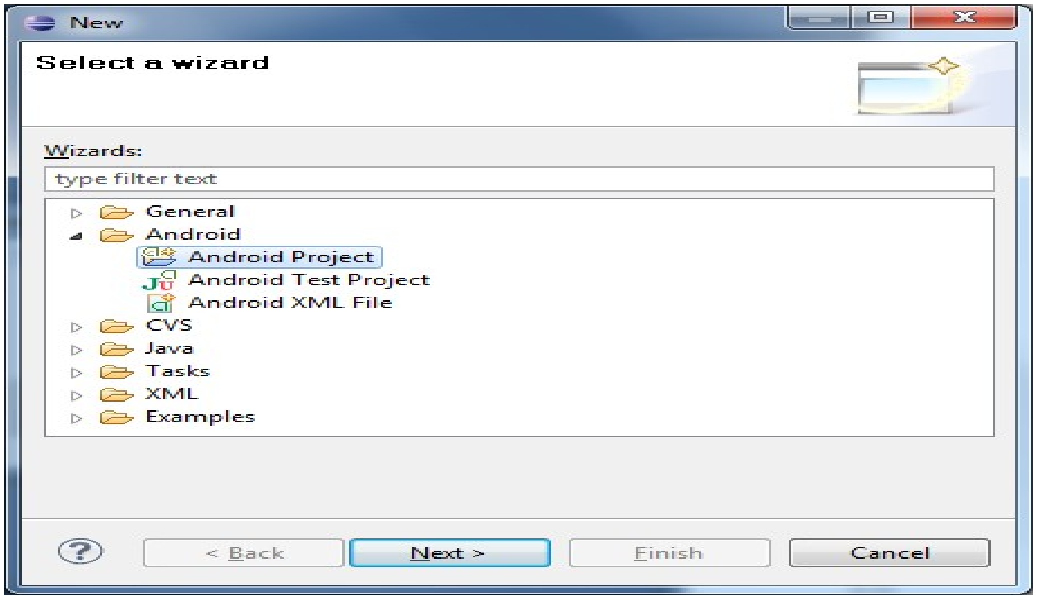
SD Card: ● Size: MiB ○ File:

Snapshot: ☐ Enabled

Skin: ● Built-in: Default (WVGA800)

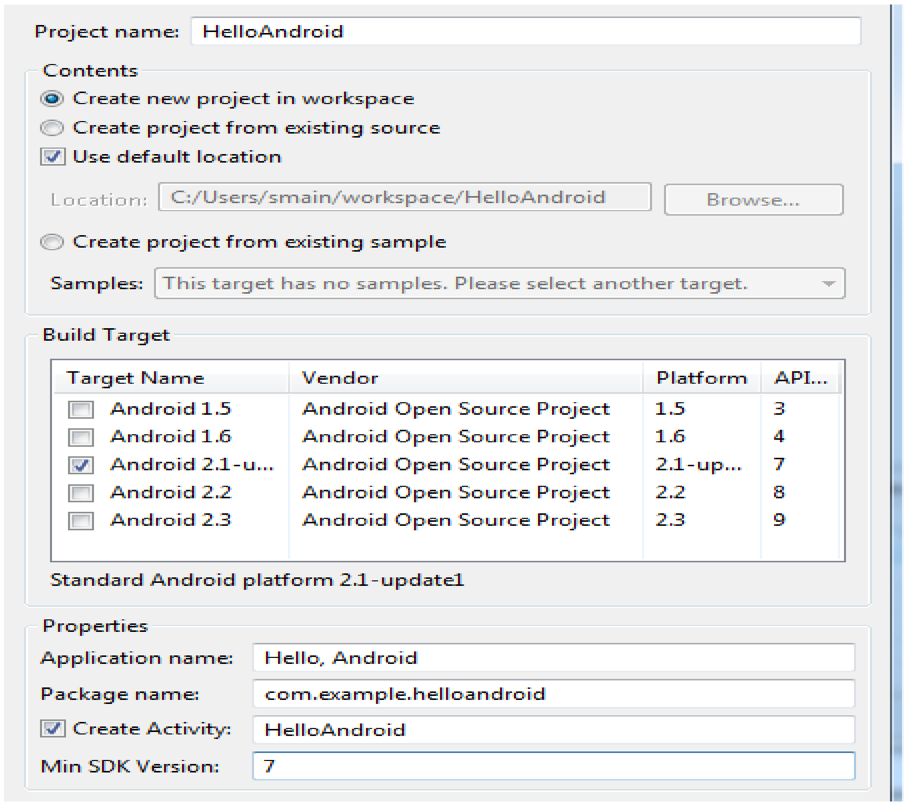
- After this, you already have Android Virtual Device for your android application, you can edit the API based on your need or create new one, depending on the application needs.

- Then we now can create new android project in Eclipse. Go to File > New > Project. Note that if you have successfully installed the ADT Plugin, there will be option to choose Android Project, then choose it.



– Fill in the project details with the following values:

- Project name: HelloAndroid
- Application name: Hello, Android
- Package name: com.example.helloandroid (or your own private namespace)
- Create Activity: HelloAndroid
- Click Finish.



Project name: HelloAndroid

Contents

- [✓ Create new project in workspace]
- Create project from existing source
- [✓] Use default location

Location: C:/Users/smain/workspace/HelloAndroid

- Create project from existing sample

Samples: This target has no samples. Please select another target.

Build Target

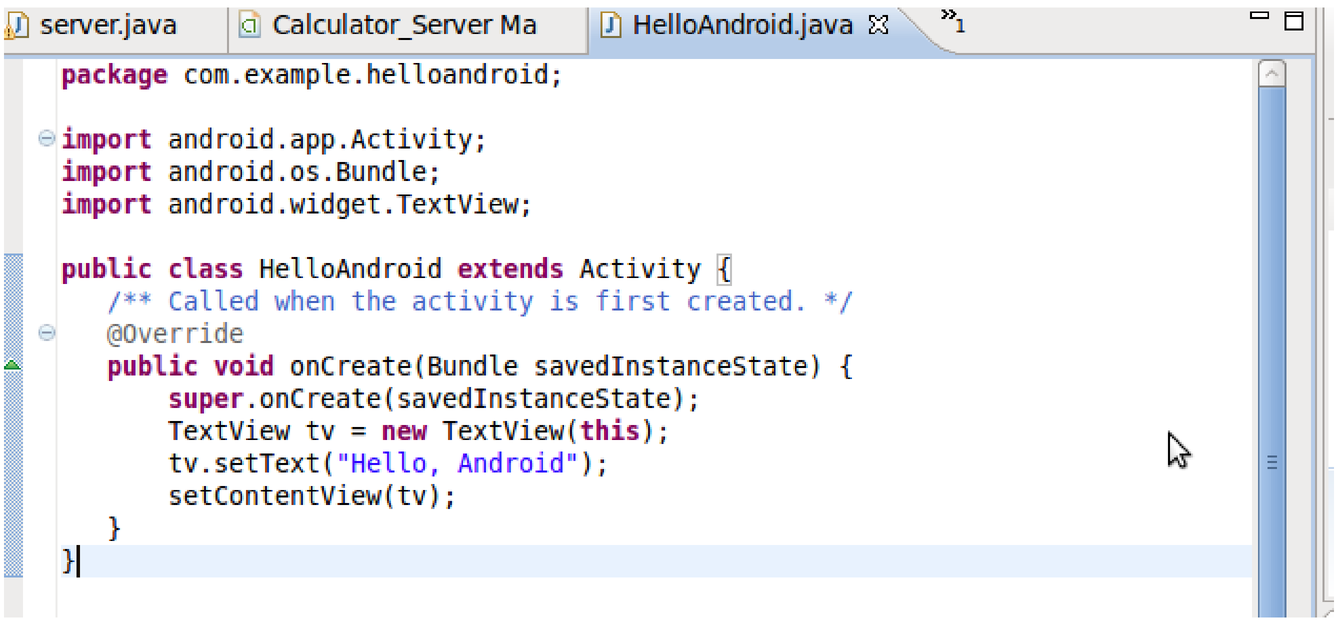
	Target Name	Vendor	Platform	API...
	Android 1.5	Android Open Source Project	1.5	3
	Android 1.6	Android Open Source Project	1.6	4
✓	Android 2.1-u...	Android Open Source Project	2.1-up...	7
	Android 2.2	Android Open Source Project	2.2	8
	Android 2.3	Android Open Source Project	2.3	9

Standard Android platform 2.1-update1

Properties

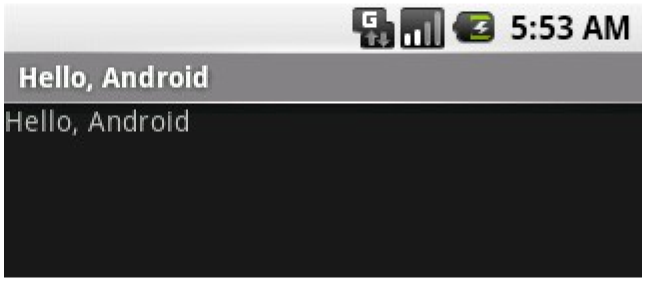
Application name:	Hello, Android
Package name:	com.example.helloandroid
[✓] Create Activity:	HelloAndroid
Min SDK Version:	7

– And then under the project directory, go to HelloAndroid > src > com.example.helloandroid > HelloAndroid.java then change the source code with this code below.



```java
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

– And then save the file and run the program, Run > Run or CRTL + F11. Wait for some time, it might take a while. And you should see something like this.



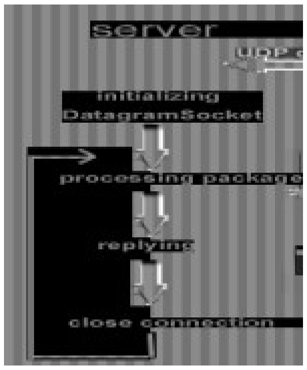
– So Congratulation! Until this point you have successfully created your first android application.

Building Remote Android Calculator.

Basically, in this lab we are tasked to build simple calculator application in Android but it has to be run remotely like client in Android enter the two numbers and operator, then they would be sent to server and being calculated then server will send back the answer.

We think that building the client calculator, would be very easy. But the difficult part is how to pass that from client to server and back to client remotely. We did some study, and there are several options on Android communication, via UDP and TCP.

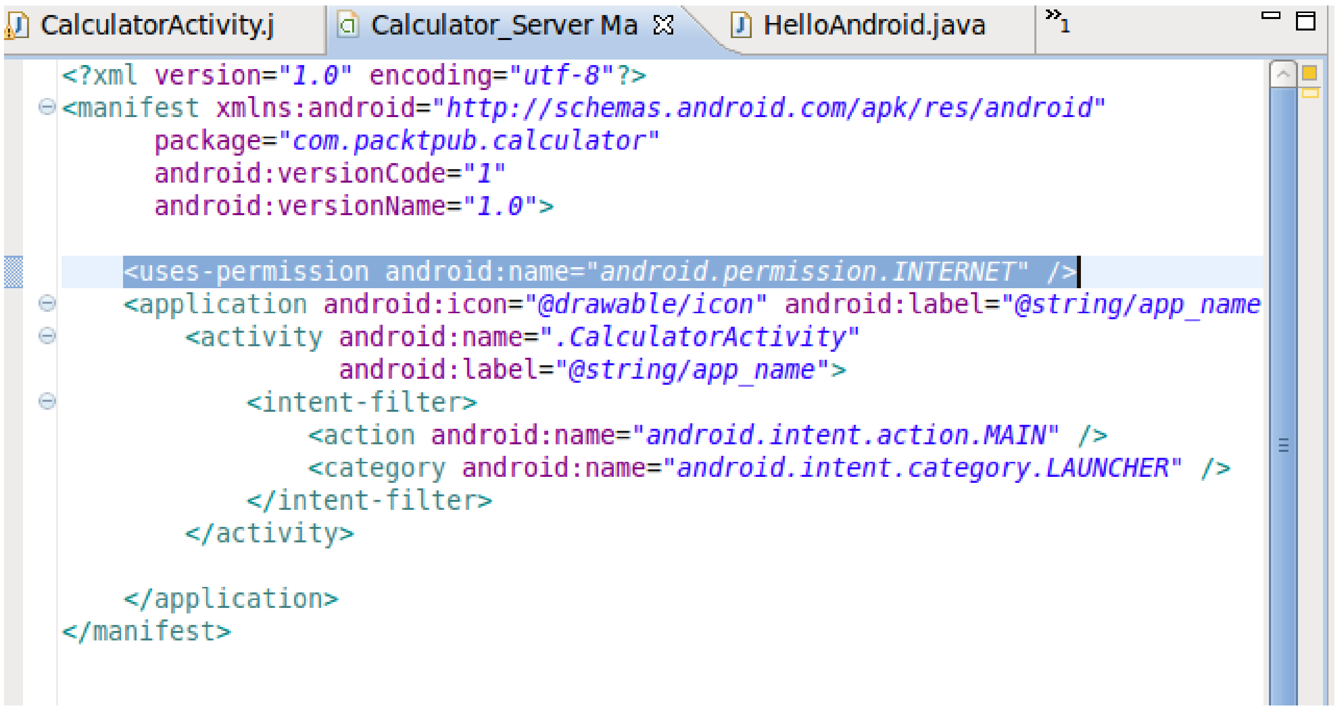
We found that UDP is easier to be implemented, so we chose UDT and this is the brief introduction on how UDT works.



The User Datagram Protocol (UDP) is one of the core members of the Internet Protocol Suite, the set of network protocols used for the Internet. With UDP, computer applications can send messages, in this case referred to as datagrams, to other hosts on an Internet Protocol (IP) network without requiring prior communications to set up special transmission channels or data paths. [4]

And as the side note, to enable the android Internet communication, you need to add this line to the AndroidManifest.xml file.

```
<uses-permission android:name="android.permission.INTERNET" />
```



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.packtpub.calculator"
      android:versionCode="1"
      android:versionName="1.0">

    <uses-permission android:name="android.permission.INTERNET" />
    <application android:icon="@drawable/icon" android:label="@string/app_name
        <activity android:name=".CalculatorActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

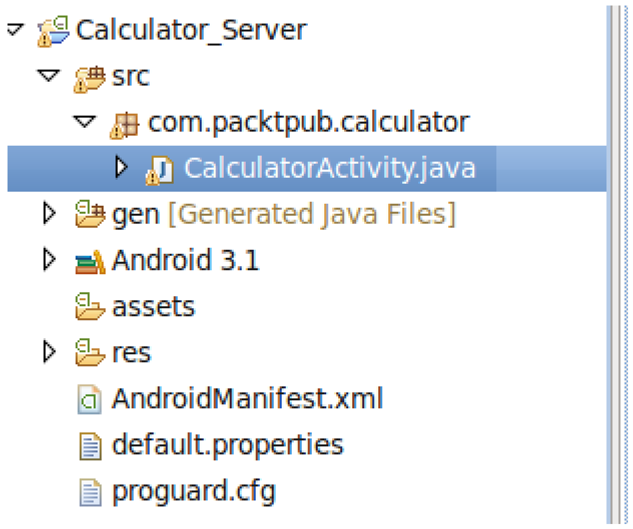
And then this is how we implemented the server side in regular java.

```
//UDPServer.java

import java.net.*;
import java.io.*;


class server
{
   public static void main(String args[]) throws Exception
      {

             byte[] receive_data = new byte[1024];
         byte[] send_data = new byte[1024];

         int recv_port;

         DatagramSocket server_socket = new DatagramSocket(5000);

         System.out.println ("UDPServer Waiting for client on port 5000");


         while(true)
         {

          DatagramPacket receive_packet = new DatagramPacket(receive_data,
                                           receive_data.length);

                  server_socket.receive(receive_packet);

                  String data = new String(receive_packet.getData(),0,0,receive_packet.getLength());

                  InetAddress IPAddress = receive_packet.getAddress();

                  recv_port = receive_packet.getPort();

                  if (data.equals("q") || data.equals("Q"))
                  break;

                  else

                  System.out.println("( " + IPAddress + " , " + recv_port
                  + " ) said :" + data );

                  String input1="";
                  String input2="";
                  char operator='+';
```

```
            //private void handleString(String data){
                if (data.length() > 0 ){
                    for (int i=0;i<data.length()-1;i++){
                        if (data.charAt(i)=='+' || data.charAt(i)=='-' || data.charAt(i)=='*' ||
data.charAt(i)=='/'){
                            input1=(data.substring(0,i));
                            input2=(data.substring(i+1,data.length()));
                            operator=(data.charAt(i));
                        }
                    }
                }else{
                    System.out.println("empty data");
                }
               //  }
                double num1 = Double.parseDouble(input1);
                double num2 = Double.parseDouble(input2);
                double num=0;

                        if (operator=='+') {
                            num = num1+num2;
                        }
                        else if (operator=='-') {
                            num = num1-num2;
                        }
                        else if (operator=='*') {
                            num = num1*num2;
                        }
                        else{
                            num = num1/num2;
                        }

                System.out.println("( " + input1 + Character.toString(operator)+ input2
                    + " ) is : " + num);


                  try {
                    String host1 = "192.168.0.12";
                    int port1 = 5000;
                    byte[] message1 = Double.toString(num).getBytes();

                    // Get the internet address of the specified host
                    InetAddress address1 = InetAddress.getByName(host1);

                    // Initialize a datagram packet with data and address
                    DatagramPacket packet1 = new DatagramPacket(message1,
message1.length,
                        address1, port1);
```

```
                                  // Create a datagram socket, send the packet through it, close it.
                                  DatagramSocket dsocket1 = new DatagramSocket();
                                  dsocket1.send(packet1);
                                  dsocket1.close();
                                } catch (Exception e) {
                                  System.err.println(e);
                                }

          }
       }
}
```

And this is how we implemented the Calculator client.



And this the code for CalculatorActivity.java

```
package com.packtpub.calculator;

import android.app.Activity;
import android.app.AlertDialog;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class CalculatorActivity extends Activity {
        private EditText input1;
        private EditText input2;
        private EditText solution;
        private TextView operator;

        private CalculatorActivity mContext;
        Double operand1;
        Double operand2;
        String Oper1;
        String Oper2;
        String Operate;
        private String ipAddress = "192.168.0.12";
        private int port = 5000;



    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mContext = this;
        setContentView(R.layout.main);
        input1 = (EditText) findViewById(R.id.input1);
         input2 = (EditText) findViewById(R.id.input2);
        solution = (EditText) findViewById(R.id.solution);
        operator = (TextView) findViewById(R.id.operator);
        /* We create anEvent in each button.*/
        Button plusButton = (Button) findViewById(R.id.plus);
        Button minusButton = (Button) findViewById(R.id.min);
        Button prodButton = (Button) findViewById(R.id.mul);
        Button divButton = (Button) findViewById(R.id.div);
        Button equalButton = (Button) findViewById(R.id.equal);
        Button oneButton = (Button) findViewById(R.id.one);
        Button twoButton = (Button) findViewById(R.id.two);
        Button threeButton = (Button) findViewById(R.id.three);
        Button fourButton = (Button) findViewById(R.id.four);
        Button fiveButton = (Button) findViewById(R.id.five);
        Button sixButton = (Button) findViewById(R.id.six);
        Button sevenButton = (Button) findViewById(R.id.seven);
        Button eightButton = (Button) findViewById(R.id.eight);
        Button nineButton = (Button) findViewById(R.id.nine);
        Button zeroButton = (Button) findViewById(R.id.zero);

        oneButton.setOnClickListener(new OnClickListener() {
                public void onClick(View arg0) {
                        input1.append("1");
                        }
                });
    twoButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                    input1.append("2");
                    }
            });
    threeButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                    input1.append("3");
                    }
            });
    fourButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                    input1.append("4");
                    }
            });
    fiveButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                    input1.append("5");
                    }
            });
    sixButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                    input1.append("6");
                    }
            });
    sevenButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                    input1.append("7");
                    }
            });
    eightButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                    input1.append("8");
                    }
            });
    nineButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                    input1.append("9");
                    }
            });
    zeroButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                    input1.append("0");
                    }
            });

  plusButton.setOnClickListener(new OnClickListener() {
    public void onClick(View arg0) {
```

```
                operator.setText("+");
                }
        });
      minusButton.setOnClickListener(new OnClickListener() {
        public void onClick(View arg0) {
                operator.setText("-");
                }
        });
      prodButton.setOnClickListener(new OnClickListener() {
        public void onClick(View arg0) {
                operator.setText("*");
                }
        });
      divButton.setOnClickListener(new OnClickListener() {
        public void onClick(View arg0) {
                operator.setText("/");
                }
        });

      equalButton.setOnClickListener(new OnClickListener() {
        private AlertDialog show;
        public void onClick(View arg0) {
                if ((input1.getText().length() == 0)
                                || (input1.getText().toString() == " ")) {
                        show = new AlertDialog.Builder(mContext).setTitle("Error")
                        .setMessage("Some inputs are empty")
                        .setPositiveButton("OK", null).show();
                }else if (operator.getText().equals("")) {
                        show = new AlertDialog.Builder(mContext).setTitle("Error")
                        .setMessage("Operator is null").setPositiveButton("OK", null).show();
                }else if (operator.getText().equals("+")) {
                        double result = new Double(input1.getText().toString())
                        + new Double(input2.getText().toString());
                        solution.setText(Double.toString(result));
                        operand1= new Double(input1.getText().toString());
                        operand2= new Double(input2.getText().toString());
                        operator.setText(operand1+"+"+operand2);
                        Oper1=operand1.toString()+ "+" + operand2.toString();
                        try{
                                String message=Oper1;
                                int server_port = 5000;
                                byte[] rec= new byte [100];
                                DatagramSocket s = new DatagramSocket();
                                InetAddress local = InetAddress.getByName("192.168.0.12");
                                int msg_length=message.length();
                                byte[] mess = message.getBytes();
                                DatagramPacket p = new DatagramPacket(mess,
msg_length,local,server_port);
```

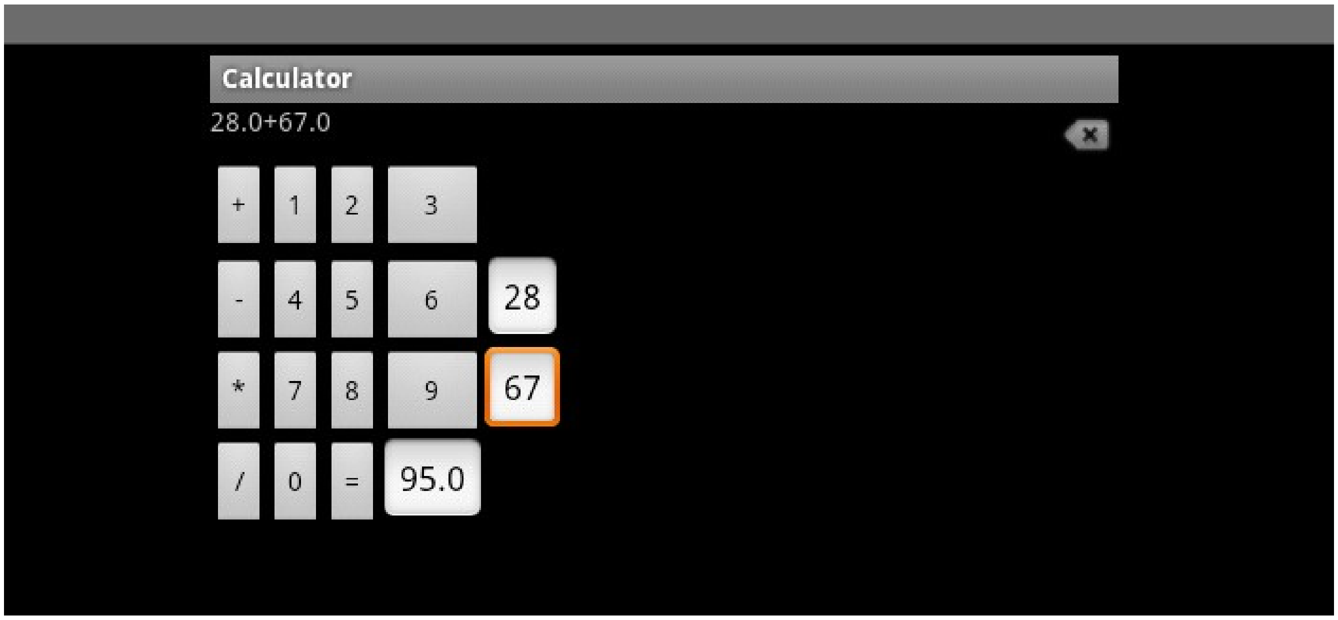
```
                                        s.send(p);
                                        //DatagramPacket p1 = new DatagramPacket(rec, rec.length);
                                        //s.receive(p1);
                                        //String t= new String(rec, 0, p1.getLength());
                                        s.close();
                                        //System.out.println("The Answer"+t);
                                        //operator.setText(t);
                                        //solution.setText(t);

                                        }
                                        catch(Exception e){}


                    } else if (operator.getText().equals("-")) {
                              double result = new Double(input1.getText().toString())
                              - new Double(input2.getText().toString());
                              solution.setText(Double.toString(result));
                              operand1= new Double(input1.getText().toString());
                              operand2= new Double(input2.getText().toString());
                              operator.setText(operand1+"-"+operand2);
                              Oper1=operand1.toString()+ "-" + operand2.toString();
                              try{
                                        String message=Oper1;
                                        int server_port = 5000;
                                        byte[] rec= new byte [100];
                                        DatagramSocket s = new DatagramSocket();
                                        InetAddress local = InetAddress.getByName("192.168.0.12");
                                        int msg_length=message.length();
                                        byte[] mess = message.getBytes();
                                        DatagramPacket p = new DatagramPacket(mess,
msg_length,local,server_port);
                                        s.send(p);
                                        //DatagramPacket p1 = new DatagramPacket(rec, rec.length);
                                        //s.receive(p1);
                                        //String t= new String(rec, 0, p1.getLength());
                                        s.close();
                                        //System.out.println("The Answer"+t);
                                        //operator.setText(t);
                                        //solution.setText(t);

                                        }
                                        catch(Exception e){}

                    } else if (operator.getText().equals("*")) {
                              double result = new Double(input1.getText().toString())
                              * new Double(input2.getText().toString());
                              solution.setText(Double.toString(result));
                              operand1= new Double(input1.getText().toString());
```

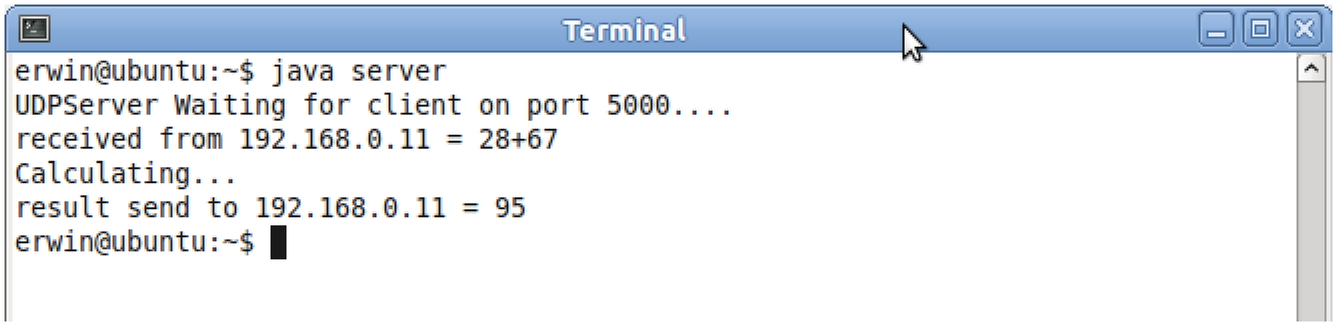
```
                        operand2= new Double(input2.getText().toString());
                        operator.setText(operand1+"*"+operand2);
                        Oper1=operand1.toString()+ "*" + operand2.toString();
                        /*try{



                        byte[] message = new byte[1500];
                        byte[] sen1=new byte[100];
                        sen1=Oper1.getBytes();
                        //InetAddress local = InetAddress.getByName("192.168.1.2");
                        InetAddress IPAdress  = InetAddress.getByName("192.168.0.20");


                        DatagramPacket p = new DatagramPacket(sen1,
sen1.length,IPAdress,port);
                        DatagramSocket s = new DatagramSocket(port);
              s.send(p);

                        s.close();
                        }
                        catch( Exception e )
                    {
                                e.printStackTrace();
                    }
                        try{

                                String text;

                                byte[] message = new byte[1500];
                                byte[] sen1=new byte[100];
                                message=Oper1.getBytes();
                                DatagramPacket p1 = new DatagramPacket(message,
message.length);
                                DatagramSocket s = new DatagramSocket(4444);
                                s.receive(p1);

                                text = new String(message, 0, p1.getLength());
                                operator.setText(text);

                                s.close();
                                }
                                catch( Exception e )
                            {
                                        e.printStackTrace();
                            }   */

                        try{
```

```
                                        String message=Oper1;
                                        int server_port = 5000;
                                        byte[] rec= new byte [100];
                                        DatagramSocket s = new DatagramSocket();
                                        InetAddress local = InetAddress.getByName("192.168.0.12");
                                        int msg_length=message.length();
                                        byte[] mess = message.getBytes();
                                        DatagramPacket p = new DatagramPacket(mess,
msg_length,local,server_port);
                                        s.send(p);
                                        //DatagramPacket p1 = new DatagramPacket(rec, rec.length);
                                        //s.receive(p1);
                                        //String t= new String(rec, 0, p1.getLength());
                                        s.close();
                                        //System.out.println("The Answer"+t);
                                        //operator.setText(t);
                                        //solution.setText(t);

                                        }
                                        catch(Exception e){}
                    }

                     else if (operator.getText().equals("/")) {
                              double result = new Double(input1.getText().toString())
                              / new Double(input2.getText().toString());
                              solution.setText(Double.toString(result));
                              operand1= new Double(input1.getText().toString());
                              operand2= new Double(input2.getText().toString());
                              operator.setText(operand1 + "/" + operand2);
                              Oper1=operand1.toString()+ "/" + operand2.toString();
                              try{
                                        String message=Oper1;
                                        int server_port = 5000;
                                        byte[] rec= new byte [100];
                                        DatagramSocket s = new DatagramSocket();
                                        InetAddress local = InetAddress.getByName("192.168.0.12");
                                        int msg_length=message.length();
                                        byte[] mess = message.getBytes();
                                        DatagramPacket p = new DatagramPacket(mess,
msg_length,local,server_port);
                                        s.send(p);
                                        //DatagramPacket p1 = new DatagramPacket(rec, rec.length);
                                        //s.receive(p1);
                                        //String t= new String(rec, 0, p1.getLength());
                                        s.close();
                                        //System.out.println("The Answer"+t);
                                        //operator.setText(t);
                                        //solution.setText(t);
```

```
                                    }
                                    catch(Exception e){}

                        }
                }
            });

        }
}
```

And this is the UI of the android application





```
erwin@ubuntu:~$ java server
UDPServer Waiting for client on port 5000....
received from 192.168.0.11 = 28+67
Calculating...
result send to 192.168.0.11 = 95
erwin@ubuntu:~$
```

Note:

- Don't forget to add the line in AndroidManifest.xml to enable Android Internet connection
- This UDP version works best with Android 3.1 API 12.
- Change the destination IP adress in the code, and make sure to use the same port to send and receive.
- Make sure that the ports are open.

References:

[1] XML-RPC, <http://www.xmlrpc.com/>

[2] Oracle – Java RMI <http://download.oracle.com/javase/1.5.0/docs/guide/rmi/hello/hello-world.html>

[3] Android Developer Guide <http://developer.android.com/guide/index.html>

[4] Simple UDT Android <http://www.helloandroid.com/tutorials/simple-udp-communication-example>