**Lab 32: gRPC using Java**

In this Experiment we will learn about gRPC

**Let’s get Started**

In this Exercise, we would be discussing one of the basic ways of communication between a program on a PC and an Android device. Here we will use the concept of Socket programming. We know communication takes place between a sender and a receiver, Socket programming involves a Client-Server setup, where a client connects to the server, send the message, and the server on the other end receives it. Now, this can be unidirectional or bidirectional based on your code.

Socket programming is a method of communicating between two devices connected to the same network. Two sockets, one on the client and one on the server, interact. An IP address plus a port make up a socket’s address. Over the specified port, the server application begins to listen to clients. The client connects to the server using the server’s IP address and the port it opens. After that, bidirectional communication is possible. Refer to this for a depth concept:

Diagram

Description automatically generated

Firstly, let us build the program that is to be executed on the server socket. We will make PC as a server and Android device as a client.

**Step 1:**Create a new project in Eclipse.

**Step 2:**Create a class Server.

|  |
| --- |
| **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.net.ServerSocket;  **import** java.net.Socket;    **public** **class** Server {    // declaring required variables  **private** **static** ServerSocket serverSocket;  **private** **static** Socket clientSocket;  **private** **static** InputStreamReader inputStreamReader;  **private** **static** BufferedReader bufferedReader;  **private** **static** String message="";    **public** **static** **void** main(String[] args) {    **try** {          // creating a new ServerSocket at port 4444          serverSocket = **new** ServerSocket(4444);        } **catch** (IOException e) {          System.out.println("Could not listen on port: 4444");      }        System.out.println("Server started. Listening to the port 4444");        // we keep listening to the socket's        // input stream until the message      // "over" is encountered  **while** (!message.equalsIgnoreCase("over")) {  **try** {                // the accept method waits for a new client connection              // and and returns a individual socket for that connection              clientSocket = serverSocket.accept();                // get the inputstream from socket, which will have                // the message from the clients              inputStreamReader = **new** InputStreamReader(clientSocket.getInputStream());              bufferedReader = **new** BufferedReader(inputStreamReader);                  // reading the message              message = bufferedReader.readLine();                // printing the message              System.out.println(message);                // finally it is very important                // that you close the sockets              inputStreamReader.close();              clientSocket.close();            } **catch** (IOException ex) {              System.out.println("Problem in message reading");          }       }    }  } |

This program when executed creates a ServerSocket on a specific port which is 4444. Now our server starts listening to the connections made by the clients that are android devices in this case.

**Create android app for clients.**

**Step by Step Implementation**

**Step 1: Create a New Project**

To create a new project in Android Studio.

Note that select **Java** as the programming language.

**Step 2:** **Working with the AndroidManifest.xml file**

|  |
| --- |
| <?**xml** version="1.0" encoding="utf-8"?>  <**manifest** xmlns:android="<http://schemas.android.com/apk/res/android>"      package="com.raghav.clientsocketapp">        <!--We require internet permission to perform networking tasks-->      <**uses-permission** android:name="android.permission.INTERNET"/>      <**application**          android:allowBackup="true"          android:icon="@mipmap/ic\_launcher"          android:label="@string/app\_name"          android:roundIcon="@mipmap/ic\_launcher\_round"          android:supportsRtl="true"          android:theme="@style/Theme.ClientSocketApp">          <**activity** android:name=".MainActivity">              <**intent-filter**>                  <**action** android:name="android.intent.action.MAIN" />                    <**category** android:name="android.intent.category.LAUNCHER" />              </**intent-filter**>          </**activity**>      </**application**>    </**manifest**> |

**Step 3: Working with the activity\_main.xml file**

Navigate to the **app > res > layout > activity\_main.xml**and add the below code to that file. Below is the code for the **activity\_main.xml** file.

|  |
| --- |
| <?**xml** version="1.0" encoding="utf-8"?>  <**LinearLayout**      xmlns:android="<http://schemas.android.com/apk/res/android>"      xmlns:app="<http://schemas.android.com/apk/res-auto>"      xmlns:tools="<http://schemas.android.com/tools>"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      android:orientation="vertical"      tools:context=".MainActivity">        <**EditText**          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:id="@+id/editText1"/>      <**Button**          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Send"          android:id="@+id/button1"/>    </**LinearLayout**> |

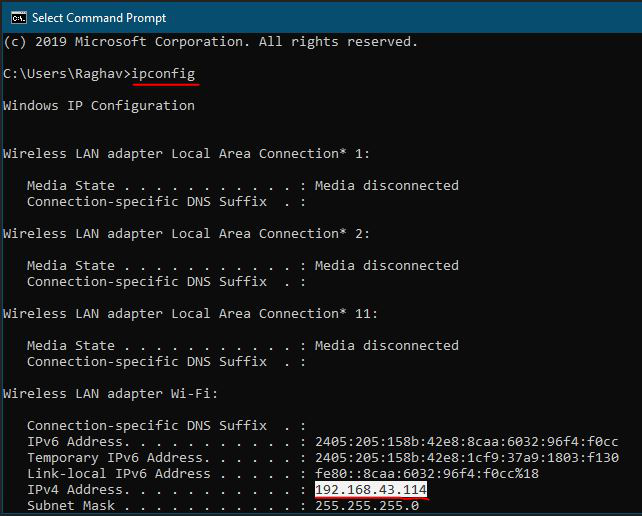
**Step 4: Working with the** **MainActivity.java file**

Go to the **MainActivity.java** file and refer to the following code. Below is the code for the **MainActivity.java** file. Comments are added inside the code to understand the code in more detail.

|  |
| --- |
| **import** androidx.appcompat.app.AppCompatActivity;    **import** android.os.Bundle;  **import** android.view.View;  **import** android.widget.Button;  **import** android.widget.EditText;  **import** java.io.IOException;  **import** java.io.PrintWriter;  **import** java.net.Socket;    **public** **class** MainActivity **extends** AppCompatActivity {        // declaring required variables  **private** Socket client;  **private** PrintWriter printwriter;  **private** EditText textField;  **private** Button button;  **private** String message;      @Override  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);              // reference to the text field          textField = (EditText) findViewById(R.id.editText1);               // reference to the send button          button = (Button) findViewById(R.id.button1);            // Button press event listener          button.setOnClickListener(**new** View.OnClickListener() {    **public** **void** onClick(View v) {                      // get the text message on the text field                  message = textField.getText().toString();                    // start the Thread to connect to server  **new** Thread(**new** ClientThread(message)).start();                }          });      }        // the ClientThread class performs       // the networking operations  **class** ClientThread **implements** Runnable {  **private** **final** String message;            ClientThread(String message) {  **this**.message = message;          }          @Override  **public** **void** run() {  **try** {                  // the IP and port should be correct to have a connection established                  // Creates a stream socket and connects it to the specified port number on the named host.                  client = **new** Socket("192.168.43.114", 4444);  // connect to server                  printwriter = **new** PrintWriter(client.getOutputStream(),**true**);                  printwriter.write(message);  // write the message to output stream                    printwriter.flush();                  printwriter.close();                      // closing the connection                  client.close();                } **catch** (IOException e) {                  e.printStackTrace();              }                // updating the UI              runOnUiThread(**new** Runnable() {                  @Override  **public** **void** run() {                      textField.setText("");                  }              });          }      }  } |

When running on devices with api11 or higher, we would get a NetworkOnMainThreadException if we do the socket programming on the main thread. To resolve this, we can either use AsyncTask class or create a new thread. Since AsyncTask is no more supported in Android R we create a simple Thread that performs the networking part.

**Getting the correct IP Address**



**Step 1:** Enable your device’s hotspot and connect your PC to this(hotspot) network.

**Step 2:**Open the command prompt and write the command “ipconfig”

**Step 3:**Copy the IPv4 Address

**Output:**

