Amit Lawanghare

Charitha Chanamolu

Cse-660

Lab 7

Android Distributed Application

1. Simple Remote Calculator

Work Done:

a. Created a graphical interface of a simple calculator which handles integer arithmetics, including addition, subtraction, multiplication and division.

b. Android Calculator is the client here and the sever is on linux machine. The User clicks on User interface to input two integers and an operation. Clients sends these integers and operation to the Server. The Server in-turn computes and sends the results back to the Client which is Android Calculator.

c. After connecting to the client, the server receives the request and performs Arthematic Operations and sends the output back to Client.

Given below is the code for both server and client…

Code

Server side – Server.java

```
public static double calcutator(JSONObject jsonObject) {
	double result = 0;
	String calculatorOperation = (String) jsonObject.get("calculatorOperation");
	double firstOperand = Double.parseDouble((String) jsonObject.get("firstOperand"));
	double secondOperand = Double.parseDouble((String) jsonObject.get("secondOperand"));
	if (calculatorOperation.equalsIgnoreCase("+")) {
		System.out.printf("Calculating: %f + %f\n", firstOperand, secondOperand);
		result = firstOperand + secondOperand;
	} else if (calculatorOperation.equalsIgnoreCase("-")) {
		System.out.printf("Calculating: %f - %f\n", firstOperand, secondOperand);
		result = firstOperand - secondOperand;
	} else if (calculatorOperation.equalsIgnoreCase("*")) {
		System.out.printf("Calculating: %f x %f\n", firstOperand, secondOperand);
		result = firstOperand * secondOperand;
	} else {
		System.out.printf("Calculating: %f / %f\n", firstOperand, secondOperand);
		result = firstOperand / secondOperand;
	}
```

```
        System.out.printf("Result: %f\n", result);
        return result;
}
```

Client Side –

MainActivity.java

```
public class MainActivity extends Activity implements View.OnClickListener {
    public static Socket socket;
    public static double result = 0;
    public static double num1 = 0;
    public static double num2 = 0;
    public static String oper = "";
    EditText t1;
    EditText t2;
    ImageButton plusImageButton;
    ImageButton minusImageButton;
    ImageButton multiplyImageButton;
    ImageButton didvideImageButton;
    TextView displayResult;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
// find the EditText elements (defined in res/layout/activity_main.xml
        t1 = (EditText) findViewById(R.id.t1);
        t2 = (EditText) findViewById(R.id.t2);
        plusImageButton = (ImageButton) findViewById(R.id.plusImageButton);
        minusImageButton = (ImageButton) findViewById(R.id.minusImageButton);
        multiplyImageButton = (ImageButton)
                findViewById(R.id.multiplyImageButton);
        didvideImageButton = (ImageButton) findViewById(R.id.divideImageButton);
        displayResult = (TextView) findViewById(R.id.displayResult);
// set listeners
        plusImageButton.setOnClickListener(this);
        minusImageButton.setOnClickListener(this);
        multiplyImageButton.setOnClickListener(this);
        didvideImageButton.setOnClickListener(this);
    }// @Override
    public void onClick(View view) {
// check if the fields are empty
        if (TextUtils.isEmpty(t1.getText().toString())
                || TextUtils.isEmpty(t2.getText().toString())) {
            return;
        }
// read EditText and fill variables with numbers
        num1 = Float.parseFloat(t1.getText().toString());
        num2 = Float.parseFloat(t2.getText().toString());
        Thread sendSocketThread = new Thread(new Client());
        switch (view.getId()) {
            case R.id.plusImageButton:
                sendSocketThread.start();
                oper = "+";
                break;
            case R.id.minusImageButton:
                oper = "-";
                sendSocketThread.start();
                break;
```

```
            case R.id.multiplyImageButton:
                oper = "*";
                sendSocketThread.start();
                break;
            case R.id.divideImageButton:
                oper = "/";
                sendSocketThread.start();
                break;
            default:
                break;
        }
        try {
            sendSocketThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
// form the output line
        displayResult.setText(num1 + " " + oper + " " + num2 + " = " + result);
    }
```

Client.java

```
public class Client implements Runnable {
    // public static double result = 0;
    @Override
    public void run() {
// TODO Auto-generated method stub
        String serverAddr = "192.168.2.100";
        int portNumber = 30021;
        try {
            MainActivity.socket = Client.initiateContact(serverAddr, portNumber);
            Client.remoteCalculation(MainActivity.socket, "multiplication",
                    MainActivity.num1, MainActivity.num2);
        } catch (UnknownHostException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
    final static Lock mutex = new ReentrantLock();
    public static void remoteCalculation(Socket socket,
                                         String calculatorOperation, double firstOperand,
                                         double secondOperand) throws IOException {
/* initiate the connection */
// Socket socket = initiateContact(serverAddr, portNumber);
        System.out.printf("connecting\n");
        String serverMessage;
        BufferedReader stdIn = new BufferedReader(new InputStreamReader(
                System.in));
/* sending and receiving with the server */
        String serverOperation = "calculator";
        String firstOperandString = String.valueOf(firstOperand);
        String secondOperandString = String.valueOf(secondOperand);
// String calculatorOperation = "multiplication";String calculatorDataString = "{" +
"\"serverOperation\":\""
        + serverOperation + "\"," + "\"calculatorOperation\":\""
                + MainActivity.oper + "\"," + "\"firstOperand\":\""
                + firstOperandString + "\"," + "\"secondOperand\":\""
                + secondOperandString + "\"}";
        System.out.println(calculatorDataString);
/* sending and receving with the server */
```

```
        sendMessage(MainActivity.socket, calculatorDataString);
// sendMessage(MainActivity.socket, calculatorDataString);
        serverMessage = receiveMessage(MainActivity.socket);
        System.out.println("Result: " + serverMessage);
        MainActivity.result = Double.parseDouble((String) serverMessage);
        stdIn.close();
        MainActivity.socket.close();
    }
    /*
    * initiate the connection with the server input parameter: server address
    * and port number. output: connection socket
    */
    public static Socket initiateContact(String serverAddr, int portNumber)
            throws UnknownHostException, IOException {
        Socket socket = null;
        socket = new Socket(serverAddr, portNumber);// ("127.0.0.1", 30021);//
// ("128.193.37.168"
        return socket;
    }/*
* receive message from a socket input parameter: socket. output: string
*/
    public static String receiveMessage(Socket socket) throws IOException {
        String inputLine = null;
        BufferedReader inputBuffer = null;
        inputBuffer = new BufferedReader(new InputStreamReader(
                socket.getInputStream()));
        inputLine = inputBuffer.readLine();
        return inputLine;
    }
    /*
    * send message to socket input: socket, string message output: void
    */
    public static void sendMessage(Socket socket, String outputLine)
            throws IOException {
        PrintWriter outputWriter = null;
        outputWriter = new PrintWriter(socket.getOutputStream(), true);
        outputWriter.println(outputLine);
    }
```

MainActivity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/linearLayout1"
        android:layout_marginLeft="12pt"
        android:layout_marginRight="12pt"
        android:layout_marginTop="4pt">
        <EditText
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:layout_marginRight="6pt"
            android:id="@+id/t1"
            android:layout_width="match_parent"
            android:inputType="numberDecimal">
```

```
    </EditText>
    <EditText
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_marginLeft="6pt"
        android:id="@+id/t2"
        android:layout_width="match_parent"
        android:inputType="numberDecimal">
    </EditText>
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout2"
    android:layout_marginTop="4pt"
    android:layout_marginLeft="6pt"
    android:layout_marginRight="6pt">
    <ImageButton
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:src="@drawable/plus"
        android:layout_weight="1"
        android:textSize="10pt"
        android:id="@+id/plus">
    </ImageButton>
    <ImageButton
        android:layout_height="wrap_content"
        android:src="@drawable/minus"
        android:layout_width="match_parent"
        android:layout_weight="1"
        android:textSize="8pt"
        android:id="@+id/minus">
    </ImageButton>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout3"
    android:layout_marginTop="4pt"
    android:layout_marginLeft="6pt"
    android:layout_marginRight="6pt">

    <ImageButton
        android:layout_height="wrap_content"
        android:src="@drawable/munus"
        android:layout_width="match_parent"
        android:layout_weight="1"
        android:textSize="10pt"
        android:id="@+id/multiply">
    </ImageButton>
    <ImageButton
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:layout_weight="1"
        android:src="@drawable/divide"
        android:textSize="10pt"
        android:id="@+id/divide">
    </ImageButton>
</LinearLayout>
<TextView
```

```
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:layout_marginLeft="6pt"
        android:layout_marginRight="6pt"
        android:textSize="12pt"
        android:layout_marginTop="4pt"
        android:id="@+id/displayResult"
        android:gravity="center_horizontal">
    </TextView>
</LinearLayout>
```

Output –

Client

Server

```
amitlinx@dellins:~660/sv$ clear
amitlinx@dellins:~660/sv$ java -cp .:json-simple-1.1.1.jar Server
Waiting for client ..
Client connected
{"ServerOperation":"calculator","calculatorOperation":"/","firstOperand":"1024.0","secondOperand":
"32.0"}
Received request
Calculating: 1021.0000/32.0000
Result: 32.0000
Done
Waiting for client ..
```

2. **Random Number Generator**

   Work Done:

a. Android device presented an User Interface to let a user enter the number of random numbers they want, and the lower bound and upper bound of the numbers.

b. The device sends the parameters to a remote server, which returns the random numbers to be displayed by the Android device.

c. After connecting to the client server receives the request and generates random numbers.

Given Below is the code for server and client to generate random numbers…

Server.java

```
public static String nRandGenWithRange(JSONObject jsonObject) {
        int n = Integer.parseInt((String) jsonObject.get("numberOfRandom"));
        int min = Integer.parseInt((String) jsonObject.get("min"));
        int max = Integer.parseInt((String) jsonObject.get("max"));
// generate zeros if max < min
        if (max < min){
        max = 0;
        min = 0;
        }
        int[] randArray = new int[n];
        Random randomGen = new Random();
        System.out.printf("Random Generating...\nNumber of random: %d Min: %d Max: %d\n", n,
        min, max);
        for (int i = 0; i < n; i++) {
        randArray[i] = min + (int) randomGen.nextInt((max - min) + 1);
        System.out.printf("%d\n", randArray[i]);
        }
```

MainActivity

```
public class MainActivity extends Activity implements View.OnClickListener {
    public static Socket socket;
    public static int NUMBER_OF_RANDOM = 0;
    public static int MIN_VALUE = 0;
    public static int MAX_VALUE = 0;
    public static String RESULT_RAMDOM = "";
    EditText number_of_random_text_field;
    EditText min_text_field;
    EditText max_text_field;
    Button generateButton;
    TextView displayResult;
    /** Called when the activity is first created. */@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
// find the EditText elements (defined in res/layout/activity_main.xml
        number_of_random_text_field = (EditText)
                findViewById(R.id.number_of_random_text_field);
        min_text_field = (EditText) findViewById(R.id.min_text_field);
```

```
        max_text_field = (EditText) findViewById(R.id.max_text_field);
        generateButton = (Button) findViewById(R.id.generate_button);
        displayResult = (TextView) findViewById(R.id.displayResult);
        displayResult.setMovementMethod(new ScrollingMovementMethod());
// set listeners
        generateButton.setOnClickListener(this);
    }
    // @Override
    @Override
    public void onClick(View view) {
// check if the fields are empty
        if (TextUtils.isEmpty(number_of_random_text_field.getText().toString())
                || TextUtils.isEmpty(min_text_field.getText().toString())
                || TextUtils.isEmpty(max_text_field.getText().toString())) {
            return;
        }
// read EditText and fill variables with numbers
        NUMBER_OF_RANDOM = Integer.parseInt(number_of_random_text_field
                .getText().toString());
        MIN_VALUE = Integer.parseInt(min_text_field.getText().toString());
        MAX_VALUE = Integer.parseInt(max_text_field.getText().toString());
        Log.d("n, min, max", NUMBER_OF_RANDOM + " " + MIN_VALUE + " "
                + MAX_VALUE);
        Thread sendSocketThread = new Thread(new Client());
        sendSocketThread.start();
        try {
            sendSocketThread.join();
        } catch (InterruptedException e) {e.printStackTrace();
        }
// form the output line
        Log.d("return", MainActivity.RESULT_RAMDOM);
        displayResult.setText(RESULT_RAMDOM);
    }
```

Client.java

```
public class Client implements Runnable {
    // public static double result = 0;
    @Override
    public void run() {
// TODO Auto-generated method stub
        String serverAddr = "192.168.2.100";
        int portNumber = 30021;
        try {
            MainActivity.socket = Client
                    .initiateContact(serverAddr, portNumber);
            remoteRandomGenerator(MainActivity.socket,
                    MainActivity.NUMBER_OF_RANDOM,
                    MainActivity.MIN_VALUE,
                    MainActivity.MAX_VALUE);
        } catch (UnknownHostException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
    final static Lock mutex = new ReentrantLock();
    public static void remoteRandomGenerator(Socket socket, int numberOfRandom,int minValue, int
maxValue) throws IOException {
/* initiate the connection */
```

```
// Socket socket = initiateContact(serverAddr, portNumber);
        System.out.printf("connecting\n");
        String serverMessage;
        BufferedReader stdIn = new BufferedReader(new InputStreamReader(
                System.in));
/* sending and receiving with the server */
        String serverOperation = "randomGenerator";
        String min = String.valueOf(minValue);
        String max = String.valueOf(maxValue);
        String n = String.valueOf(numberOfRandom);
        String randomGeneratorDataString = "{" + "\"serverOperation\":\""
                + serverOperation + "\"," + "\"numberOfRandom\":\"" + n + "\","
                + "\"min\":\"" + min + "\"," + "\"max\":\"" + max + "\"}";
        System.out.println(randomGeneratorDataString);
        System.out.println(randomGeneratorDataString);
        sendMessage(MainActivity.socket, randomGeneratorDataString);
// sendMessage(MainActivity.socket, calculatorDataString);
        serverMessage = receiveMessage(MainActivity.socket);
        MainActivity.RESULT_RAMDOM = serverMessage;//
        Double.parseDouble((String)
// serverMessage);
                stdIn.close();
        MainActivity.socket.close();
    }
    /*
    * initiate the connection with the server input parameter: server address
    * and port number. output: connection socket
    */
    public static Socket initiateContact(String serverAddr, int portNumber)
            throws UnknownHostException, IOException {
        Socket socket = null;
        socket = new Socket(serverAddr, portNumber);// ("127.0.0.1", 30021);//
// ("128.193.37.168"return socket;
    }
    /*
    * receive message from a socket input parameter: socket. output: string
    */
    public static String receiveMessage(Socket socket) throws IOException {
        String inputLine = null;
        BufferedReader inputBuffer = null;
        inputBuffer = new BufferedReader(new InputStreamReader(
                socket.getInputStream()));
        inputLine = inputBuffer.readLine();
        return inputLine;
    }
    /*
    * send message to socket input: socket, string message output: void
    */
    public static void sendMessage(Socket socket, String outputLine)
            throws IOException {
        PrintWriter outputWriter = null;
        outputWriter = new PrintWriter(socket.getOutputStream(), true);
        outputWriter.println(outputLine);
    }
```

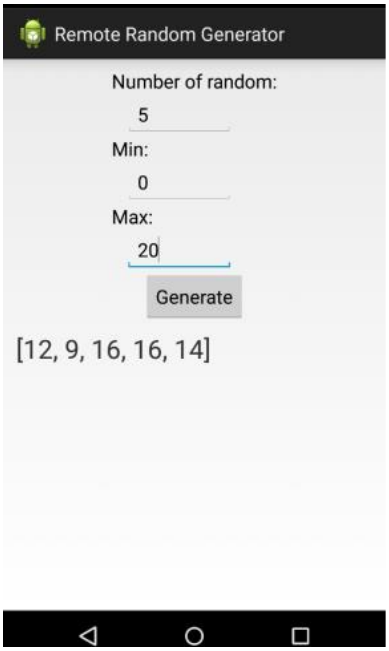
activity_main.xml

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Number of random:"
    android:textAppearance="?android:attr/textAppearanceMedium" />
<EditText
android:id="@+id/number_of_random_text_field"
android:layout_width="50pt"
android:layout_height="wrap_content"
android:layout_marginLeft="6pt"
android:layout_weight="1"android:inputType="numberDecimal" >
</EditText>
<TextView
android:id="@+id/textView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="Min:"
android:textAppearance="?android:attr/textAppearanceMedium" />
```

Output –

Client – 5 Random numbers from 0 to 20



Server –

```
amitlinx@dellins:~660/sv$ java -cp .:json-simple-1.1.1.jar Server
Waiting for client ..
Client connected
{“ServerOperation”:”randomGenerator”,”numberofRandom”:”5”,”min”:”0”,”max”:”20”}
Received request
Number Of random: 5 Min:0 Max:20
12
9
16
16
14
Done
```