Android Application Development

A Internship report submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

Submitted by

FOCUS ON EXCELLENCE

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)
ANGAMALY-683577

Affiliated to

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
Kerala -695016

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)

Mookkannoor(P.O), Angamaly-683577

FISAT

FOCUS ON EXCELLENCE

CERTIFICATE

This is to certify the Internship report entitled "**Android Application Development**" is a bonafide report of the Internship presented during IVth semester by **Yadhu Krishnan PC**, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B.Tech) in Computer Science & Engineering.

Dr. Prasad J C

Staff in Charge | Head of the Department

Place:

Date:

ABSTRACT

CAMINO INFOTECH is known for its full-stack development package. With focus teams for every layer of the applicationfrom fluidly responsive screens to powerful data crunching engines and analytical subsystemswe deliver a rock-solid solution.

Our front-end engineers work with the latest JS frameworks and libraries to bring mockups from development to delivery. To build the application brain, skilled backend developers team up with database experts and experienced data scientists. On the cloud or otherwise, we build you resilient systems meshing the various components together with suitable middleware.

For businesses that prefer everything under one roof, we offer design, usability, and other complementary services along with custom application development.

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to my teachers Reshmi R and Jestin Joy as well as our principal Dr. George Issac who gave me the permission to do this wonderful internship on the topic Android Application Development,and Camino Infotech,They provide me with this program, which also helped me in doing a lot of Research and I came to know about so many new things,I am really thankful to them. Secondly I would also like to thank my parents and friends who helped me a lot in completing this project within the limited time frame.

Yadhu Krishnan PC

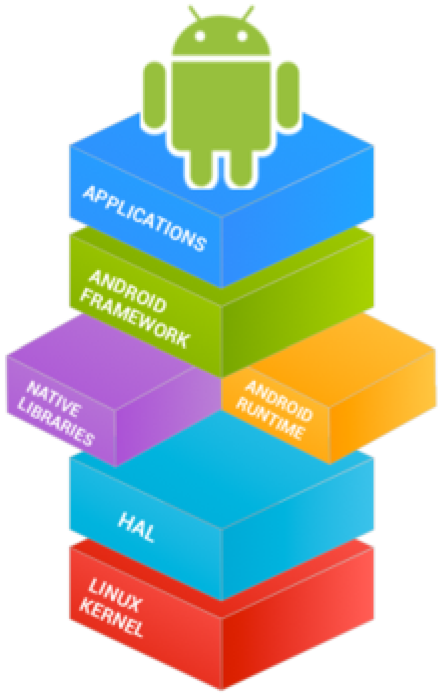
Contents

Chapter 1

INTRODUCTION

1.1 Overview

Android software development is the process by which new applications are created for devices running the Android operating system. Google states that, ”Android apps can be written using Kotlin, Java, and C++ languages” using the Android software development kit (SDK), while using other languages is also possible. All non-JVM languages, such as Go (JavaScript, C, C++ or assembly), need the help of JVM language code, that may be supplied by tools, likely with restricted API support. Some languages/programming tools allow cross-platform app support, i.e. for both Android and iOS. Third party tools, development environments and language support have also continued to evolve and expand since the initial SDK was released in 2008 The Android software development kit (SDK) includes a comprehensive set of development tools.[5] These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7 or later. As of March 2015, the SDK is not available on Android itself, but software development is possible by using specialized Android applications.



Stack.png ANDROID STACK

Until around the end of 2014, the officially supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box,[9] and NetBeans IDE also supports Android development via a plugin.[10] As of 2015, Android Studio,[11] made by Google and powered by IntelliJ, is the official IDE; however, developers are free to use others, but Google made it clear that ADT was officially deprecated since the end of 2015 to focus on Android Studio as the official Android IDE. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).[12][6]

Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.[13]

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains .dex files[14] (compiled byte code files called Dalvik executables), resource files, etc.

Chapter 2

ANDROID OPERATING SYSTEM

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The operating system has since gone through multiple major releases, with the current version being 8.1 "Oreo", released in December 2017. The core Android source code is known as Android Open Source Project (AOSP), and is primarily licensed under the Apache License.

Android is also associated with a suite of proprietary software developed by Google, including core apps for services such as Gmail and Google Search, as well as the application store and digital distribution platform Google Play, and associated development platform. These apps are licensed by manufacturers of Android devices certified under standards imposed by Google, but AOSP has been used as the basis of competing Android ecosystems, such as Amazon.com's Fire OS, which utilize their own equivalents to the Google Mobile Services.

Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system, and as of June 2018, the Google Play store features over 3.3 million apps.

Chapter 3

HISTORY AND MARKETSHARE

Android was created by the Open Handset Alliance, which is led by Google. The early feedback on developing applications for the Android platform was mixed.[73] Issues cited include bugs, lack of documentation, inadequate QA infrastructure, and no public issue-tracking system. (Google announced an issue tracker on January 18, 2008.)[74] In December 2007, MergeLab mobile startup founder Adam MacBeth stated, "Functionality is not there, is poorly documented or just doesn't work... It's clearly not ready for prime time."[75] Despite this, Android-targeted applications began to appear the week after the platform was announced. The first publicly available application was the Snake game.[76][77]

A preview release of the Android SDK was released on November 12, 2007. On July 15, 2008, the Android Developer Challenge Team accidentally sent an email to all entrants in the Android Developer Challenge announcing that a new release of the SDK was available in a "private" download area. The email was intended for winners of the first round of the Android Developer Challenge. The revelation that Google was supplying new SDK releases to some developers and not others (and keeping this arrangement private) led to widely reported frustration within the Android developer community at the time.[78]

On August 18, 2008, the Android 0.9 SDK beta was released. This release provided an updated and extended API, improved development tools and an updated design for the home screen. Detailed instructions for upgrading are available to those already working with an earlier release.[79] On September 23, 2008, the Android 1.0 SDK (Release 1) was released.[80] According to the release notes, it included "mainly bug fixes, although some smaller features were added." It also included several API changes from the 0.9 version. Multiple versions have been

released since it was developed . app.png

On December 5, 2008, Google announced the first Android Dev Phone, a SIM-unlocked and hardware-unlocked device that is designed for advanced developers. It was a modified version of HTC's Dream phone. While developers can use regular consumer devices to test and use their applications, some developers may choose a dedicated unlocked or no-contract device.

As of July 2013, more than one million applications have been developed for

Android,[82] with over 25 billion downloads.[83][84] A June 2011 research indicated that over 67

Chapter 4

DESIGN

package com.example.helloworld;

import android.support.v7.app.AppCompatActivity; import android.os.Bundle;

public class MainActivity extends AppCompatActivity @Override protected void onCreate(Bundle savedInstanceState) super.onCreate(savedInstanceState); setContentView(R.layout.activity$_m$*ain*); *ThisisasimpleAndroidhelloworldprogram.Themainactivi*

4.1 Android Libraries

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows

android.app Provides access to the application model and is the cornerstone of all Android applications.

android.content Facilitates content access, publishing and messaging between applications and application components.

android.database Used to access data published by content providers and includes SQLite database management classes.

android.opengl A Java interface to the OpenGL ES 3D graphics rendering API.

android.os Provides applications with access to standard operating system services including messages, system services and inter-process communication.

android.text Used to render and manipulate text on a device display.

android.view The fundamental building blocks of application user interfaces.

android.widget A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.

android.webkit A set of classes intended to allow web-browsing capabilities to be built into applications.

Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

4.2 Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual

Machine which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

4.3 Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

4.4 Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services

Activity Manager Controls all aspects of the application lifecycle and activity stack.

Content Providers Allows applications to publish and share data with other applications.

Resource Manager Provides access to non-code embedded resources such as strings, color settings and user interface layouts.

Notifications Manager Allows applications to display alerts and notifications to the user.

View System An extensible set of views used to create application user interfaces.

4.5 Applications

We will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

4.6 UI controls

Input controls are the interactive components in your app's user interface. Android provides a wide variety of controls you can use in your UI, such as buttons, text fields, seek bars, check box, zoom buttons, toggle buttons, and many more. A View is an object that draws something on the screen that the user can interact with and a ViewGroup is an object that holds other View (and ViewGroup) objects in order to define the layout of the user interface.

You define your layout in an XML file which offers a human-readable structure for the layout, similar to HTML. For example, a simple vertical layout with a text view and a button looks like this

¡?xml version=”1.0” encoding=”utf-8”?¿ ¡LinearLayout xmlns:android=”http://schemas.androic
android:layout$_w$*idth = ”fill*$_p$*arent”android : layout*$_h$*eight = ”fill*$_p$*arent”android : orientation = ”vertical” >*

¡TextView android:id=”@+id/text” android:layout$_w$*idth = ”wrap*$_c$*ontent”android : layout*$_h$*eight = ”wrap*$_c$*ontent”android : text = ”IamaTextView” / >*

¡Button android:id=”@+id/button” android:layout$_w$*idth = ”wrap*$_c$*ontent”android : layout*$_h$*eight = ”wrap*$_c$*ontent”android : text = ”IamaButton” / >< /LinearLayout >*

4.7 Event Handling

Events are a useful way to collect data about a user's interaction with interactive components of Applications. Like button presses or screen touch etc. The Android framework maintains an event queue as first-in, first-out (FIFO) basis. You can capture these events in your program and take appropriate action as per requirements.

There are following three concepts related to Android Event Management

Event Listeners An event listener is an interface in the View class that contains a single callback method. These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.

Event Listeners Registration Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.

Event Handlers When an event happens and we have registered an event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

Event Listeners Event Handlers Event Handler Event Listener Description onClick() OnClickListener()

This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick() event handler to handle such event.

onLongClick() OnLongClickListener()

This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use onLongClick() event handler to handle such event.

onFocusChange() OnFocusChangeListener()

This is called when the widget looses its focus ie. user goes away from the view item. You will use onFocusChange() event handler to handle such event.

onKey() OnFocusChangeListener()

This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such event.

onTouch() OnTouchListener()

This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such event.

onMenuItemClick() OnMenuItemClickListener()

This is called when the user selects a menu item. You will use onMenuItem-Click() event handler to handle such event.

onCreateContextMenu() onCreateContextMenuItemListener()

This is called when the context menu is being built(as the result of a sustained "long click)

There are many more event listeners available as a part of View class like On-HoverListener, OnDragListener etc which may be needed for your application. So I recommend to refer official documentation for Android application development in case you are going to develop a sophisticated apps.

Event Listeners Registration Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event. Though there are several tricky ways to register your event listener for any event, but I'm going to list down only top 3 ways, out of which you can use any of them based on the situation.

Using an Anonymous Inner Class

Activity class implements the Listener interface.

Using Layout file activity$_main.xmltospecifyeventhandlerdirectly.$

Below section will provide you detailed examples on all the three scenarios

Touch Mode Users can interact with their devices by using hardware keys or buttons or touching the screen.Touching the screen puts the device into touch mode. The user can then interact with it by touching the on-screen virtual buttons, images, etc.You can check if the device is in touch mode by calling the View classs isInTouchMode() method.

Focus A view or widget is usually highlighted or displays a flashing cursor when its in focus. This indicates that its ready to accept input from the user.

isFocusable() it returns true or false

isFocusableInTouchMode() checks to see if the view is focusable in touch mode. (A view may be focusable when using a hardware key but not when the device is in touch mode)

android:foucsUp="@=id/button$_l$"$onTouchEvent()publicbooleanonTouchEvent(motionEvente$

case TOUCH$_UP : Toast.makeText(this, "youhaveclickeduptouchbutton", Toast.LENTH_LON$

case TOUCH$_MOVE : Toast.makeText(this, "youhaveclickedmovetouchbutton" Toast.LENTH$

But if you applied the handler to more than one control, you would have to cut and paste the code for the handler and if the code for the handler is long, it makes the code harder to maintain.

Following are the simple steps to show how we will make use of separate Listener class to register and capture click event. Similar way you can implement your listener for any other required event type.

Step Description 1 You will use Android studio IDE to create an Android application and name it as myapplication under a package com.example.myapplication as explained in the Hello World Example chapter. 2 Modify src/MainActivity.java file to add click event listeners and handlers for the two buttons defined. 3 Modify the detault content of res/layout/activity$_m$*ain.xmlfiletoincludeAndroidUIcontrols.4Noneedtodecl*

package com.example.myapplication;

import android.app.ProgressDialog; import android.os.Bundle; import android.support.v7.app.A
import android.view.View; import android.widget.Button; import android.widget.TextView;

public class MainActivity extends ActionBarActivity private ProgressDialog progress; Button b1,b2;

@Override protected void onCreate(Bundle savedInstanceState) super.onCreate(savedInstanceSt
setContentView(R.layout.activity$_m$*ain*); *progress* = *newProgressDialog*(*this*);

b1=(Button)findViewById(R.id.button); b2=(Button)findViewById(R.id.button2); b1.setOnClickListener(new View.OnClickListener()

@Override public void onClick(View v) TextView txtView = (TextView) findViewById(R.id.textView); txtView.setTextSize(25););

b2.setOnClickListener(new View.OnClickListener()

@Override public void onClick(View v) TextView txtView = (TextView) findViewById(R.id.textView); txtView.setTextSize(55);); Following will be the content of res/layout/activity$_m$*ain.xmlfile*

Here abc indicates about tutorialspoint logo ¡?xml version="1.0" encoding="utf-8"?¿ ¡RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools" android:layout$_w$*idth* = "*match*$_p$*arent*" *android* : *layout*$_h$*eight* = "*match*$_p$*arent*" *android* : *paddingBottom* = "@*dimen*/*activity*$_v$*ertical*$_m$*argin*" *andro*
paddingLeft = "@*dimen*/*activity*$_h$*orizontal*$_m$*argin*" *android* : *paddingRight* = "@*dimen*/*activity*$_h$*orizontal*$_m$*argin*" *android* : *paddingTop* = "@*dimen*/*activity*$_v$*ertical*$_m$*argin*" *too*
context = ".*MainActivity*" >

¡TextView android:id="@+id/textView1" android:layout$_w$*idth* = "*wrap*$_c$*ontent*" *android* : *layout*$_h$*eight* = "*wrap*$_c$*ontent*" *android* : *text* = "*EventHandling*" *android* : *layout*$_a$*lignParentTop*
"*true*" *android* : *layout*$_c$*enterHorizontal* = "*true*" *android* : *textSize* = "30*dp*" / >

¡TextView android:id="@+id/textView2" android:layout$_w$*idth* = "*wrap*$_c$*ontent*" *android* : *layout*$_h$*eight* = "*wrap*$_c$*ontent*" *android* : *text* = "*Tutorialspoint*" *android* : *textColor* = "*ff*87*ff*09" *android* : *textSize* = "30*dp*" *android* : *layout*$_a$*bove* = "@+*id*/*imageButton*" *android* : *layout*$_c$*enterHorizontal* = "*true*" *android* : *layout*$_m$*arginBottom* = "40*dp*" / >

¡ImageButton android:layout$_w$*idth* = "*wrap*$_c$*ontent*" *android* : *layout*$_h$*eight* = "*wrap*$_c$*ontent*" *android* : *id* = "@+*id*/*imageButton*" *android* : *src* = "@*drawable*/*abc*" *android* : *layout*$_c$*enterVertical* = "*true*" *android* : *layout*$_c$*enterHorizontal* = "*true*" / >

¡Button android:layout$_w$*idth* = "*wrap*$_c$*ontent*" *android* : *layout*$_h$*eight* = "*wrap*$_c$*ontent*" *android* :
text = "*Smallfont*" *android* : *id* = "@ + *id*/*button*" *android* : *layout*$_b$*elow* = "@ + *id*/*imageButton*" *android* : *layout*$_c$*enterHorizontal* = "*true*" / >

¡Button android:layout$_w$*idth* = "*wrap*$_c$*ontent*" *android* : *layout*$_h$*eight* = "*wrap*$_c$*ontent*" *android* :
text = "*LargeFont*" *android* : *id* = "@+*id*/*button*2" *android* : *layout*$_b$*elow* = "@+ *id*/*button*" *android* : *layout*$_a$*lignRight* = "@+*id*/*button*" *android* : *layout*$_a$*lignEnd* = "@ + *id*/*button*" / >

¡TextView android:layout$_w$*idth* = "*wrap*$_c$*ontent*" *android* : *layout*$_h$*eight* = "*wrap*$_c$*ontent*" *android* : *text* = "*HelloWorld*!" *android* : *id* = "@+*id*/*textView*" *android* : *layout*$_b$*elow* = "@+*id*/*button*2" *android* : *layout*$_c$*enterHorizontal* = "*true*" *android* : *textSize* = "25*dp*" / >

¡/RelativeLayout¿ Following will be the content of res/values/strings.xml to define two new constants

¡?xml version="1.0" encoding="utf-8"?¿ ¡resources¿ ¡string name="app$_n$*ame*" >
myapplication < /*string* >< /*resources* > *FollowingisthedefaultcontentofAndroidManifest.xm*

¡?xml version="1.0" encoding="utf-8"?¿ ¡manifest xmlns:android="http://schemas.android.com
package="com.example.myapplication" ¿

¡application android:allowBackup="true" android:icon="@drawable/ic$_launcher" android : label = "@string/app_name" android : theme = "@style/AppTheme" >$

¡activity android:name="com.example.myapplication.MainActivity" android:label="@string/ap

¡intent-filter¿ ¡action android:name="android.intent.action.MAIN" /¿ ¡category android:name="android.intent.category.LAUNCHER" /¿ ¡/intent-filter¿

¡/activity¿

¡/application¿ ¡/manifest¿ Let's try to run your myapplication application. I assume you had created your AVD while doing environment setup. To run the app from Android Studio, open one of your project's activity files and click Run Eclipse Run Icon icon from the toolbar. Android Studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window Now you try to click on two buttons, one by one and you will see that font of the Hello World text will change, which happens because registered click event handler method is being called against each click event.

Exercise I will recommend to try writing different event handlers for different event types and understand exact difference in different event types and their handling. Events related to menu, spinner, pickers widgets are little different but they are also based on the same concepts as explained above.

4.8 Calculator Program

This is simple calculator program using android. package com.hackpundit.www.assignment1;

import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.view.Menu; import android.view.MenuItem; import android.view.View; import android.widget.Button; import android.widget.EditText;

public class MainActivity extends AppCompatActivity

Button button0 , button1 , button2 , button3 , button4 , button5 , button6 , button7 , button8 , button9 , buttonAdd , buttonSub , buttonDivision , buttonMul , button10 , buttonC , buttonEqual ;

EditText edt1 ;

float mValueOne , mValueTwo ;

boolean mAddition , mSubtract ,mMultiplication ,mDivision ;

@Override protected void onCreate(Bundle savedInstanceState) super.onCreate(savedInstanceSt
setContentView(R.layout.activity$_main$);

button0 = (Button) findViewById(R.id.button0); button1 = (Button) findViewById(R.id.button1); button2 = (Button) findViewById(R.id.button2); button3 = (Button) findViewById(R.id.button3); button4 = (Button) findViewById(R.id.button4); button5 = (Button) findViewById(R.id.button5); button6 = (Button) findViewById(R.id.button6); button7 = (Button) findViewById(R.id.button7); button8 = (Button) findViewById(R.id.button8); button9 = (Button) findViewById(R.id.button9); button10 = (Button) findViewById(R.id.button10); buttonAdd = (Button) findViewById(R.id.buttonadd); buttonSub = (Button) findViewById(R.id.buttonsub); buttonMul = (Button) findViewById(R.id.buttonmul); buttonDivision = (Button) findViewById(R.id.buttondiv); buttonC = (Button) findViewById(R.id.buttonC); buttonEqual = (Button) findViewById(R.id.buttoneql); edt1 = (EditText) findViewById(R.id.edt1);

button1.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(edt1.getText()+"1"););
button2.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(edt1.getText()+"2"););
button3.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(edt1.getText()+"3"););
button4.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(edt1.getText()+"4"););
button5.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(edt1.getText()+"5"););
button6.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(edt1.getText()+"6"););
button7.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(edt1.getText()+"7"););
button8.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(edt1.getText()+"8"););
button9.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(edt1.getText()+"9"););
button0.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(edt1.getText()+"0"););
buttonAdd.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v)
if (edt1 == null) edt1.setText(""); else mValueOne = Float.parseFloat(edt1.getText() + ""); mAddition = true; edt1.setText(null););
buttonSub.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) mValueOne = Float.parseFloat(edt1.getText() + ""); mSubtract = true ; edt1.setText(null););
buttonMul.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) mValueOne = Float.parseFloat(edt1.getText() + ""); mMultiplication = true ; edt1.setText(null););
buttonDivision.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) mValueOne = Float.parseFloat(edt1.getText()+""); mDivision = true ; edt1.setText(null););
buttonEqual.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) mValueTwo = Float.parseFloat(edt1.getText() + "");
if (mAddition == true)
edt1.setText(mValueOne + mValueTwo +""); mAddition=false;
if (mSubtract == true) edt1.setText(mValueOne - mValueTwo+""); mSubtract=false;
if (mMultiplication == true) edt1.setText(mValueOne * mValueTwo+""); mMultiplication=false;
if (mDivision == true) edt1.setText(mValueOne / mValueTwo+""); mDivision=false;);
buttonC.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(""););
button10.setOnClickListener(new View.OnClickListener() @Override public void onClick(View v) edt1.setText(edt1.getText()+"."););

Chapter 5

Testing

5.1 Run on a real device

Set up your device as follows:

Connect your device to your development machine with a USB cable. If you're developing on Windows, you might need to install the appropriate USB driver for your device. Enable USB debugging in the Developer options as follows. First, you must enable the developer options:

Open the Settings app. (Only on Android 8.0 or higher) Select System. Scroll to the bottom and select About phone. Scroll to the bottom and tap Build number 7 times. Return to the previous screen to find Developer options near the bottom. Open Developer options, and then scroll down to find and enable USB debugging.

Run the app on your device as follows:

In Android Studio, click the app module in the Project window and then select Run ¿ Run (or click Run in the toolbar). In the Select Deployment Target window, select your device, and click OK.

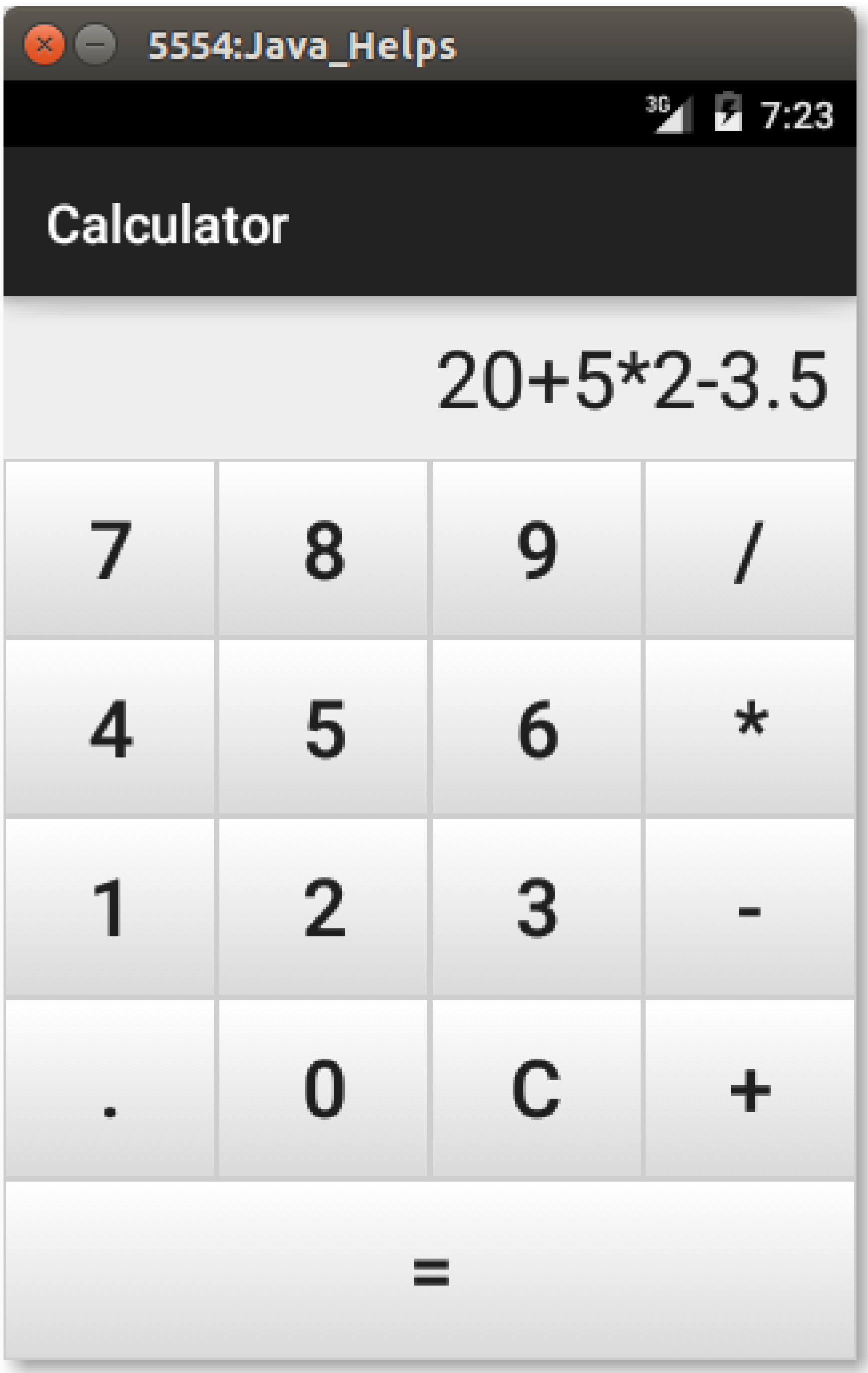
5.2 Run on an emulator

Run the app on an emulator as follows:

1.In Android Studio, click the app module in the Project window and then select Run ¿ Run (or click Run in the toolbar). 2.In the Select Deployment Target window, click Create New Virtual Device. 3.In the Select Hardware screen, select a phone device, such as Pixel, and then click Next. 4.In the System Image screen, select the version with the highest API level. If you don't have that version installed, a Download link is shown, so click that and complete the download. 5.Click Next. 6.On the Android Virtual Device (AVD) screen, leave all the settings alone and click Finish. 7.Back in the Select Deployment Target dialog, select the device you just created and click OK. 8.Android Studio installs the app on the emulator and starts it. You should now see "Hello World!" displayed in the app running on the emulator.

Chapter 6

RESULTS



calculator using Android.png

Chapter 7

CONCLUSION

Have you got any ideas for mobile application yet? Yes, you probably have several in thoughts by now! These 5 motorists can be used over and over again in your pursuit for growing, maintainable earnings. Make sure to look with a balanced view. Go strong into your procedures.

The technique may be as easy as enhancing a current create, TV, poster or on the internet technique, or be entirely, exclusively mobile. This is new scenery enjoy being an innovator, safe in the knowledge that the resources are well proven, affordable and are the most commonly used technology around.

Chapter 8

What's Next

I still want to write about handling data over the wire and Commands in Android. Ill continue writing articles in the Android Architecture series, but Ill also be throwing in some other post here and there as I come up with things. I think the next post may deal with styling Android in the appropriate way.