Maharashtra Education Society's

Abasaheb Garware College, Pune

(Autonomous)

(Affiliated to Savitribai Phule Pune University)

As per NEP Structure

T.Y.B.C.A.

Semester -VI

CA-354-MJP Practical based on Mobile Application Development

Work Book

Name: ____

Roll No:__________________Seat No__________________

Academic Year: 20_______- 20___

Introduction

1. About the Workbook:

This workbook is intended to be used by TYBCA students for the Mobile Application Development Assignments in Semester–VI. This workbook is designed by considering all the practical concepts / topics mentioned in syllabus.

2. The objectives of this Workbook are:

- Defining the scope of the course.
- To bring the uniformity in the practical conduction and implementation in all colleges affiliated to SPPU.
- To have continuous assessment of the course and students.
- Providing ready reference for the students during practical implementation.
- Provide more options to students so that they can have good practice before facingthe examination.
- Catering to the demand of slow and fast learners and accordingly providing the practice assignments to them.

3. Instructions to the students:

Please read the following instructions carefully and follow them.

- Students are expected to carry this workbook every time they come to the lab for practical.
- Students should prepare for the assignment by reading the relevant material which is mentioned in ready reference.
- Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However, student should spend additional hours in Lab and at home to cover all workbook assignments if needed.
- Students will be assessed for each assignment on a scale from 0 to 5

Not done	0
Incomplete	1
Late Complete	2
Needs improvement	3
Complete	4
Well Done	5

4. **Instruction to the Instructors:**

- Make sure that students should follow above instructions.
- Explain the assignment and related concepts using white board if required orby demonstrating the software.
- Give specific input to fill the blanks in queries which can vary from studentto student.
- Evaluate each assignment carried out by a student on a scale of 5 as specifiedabove by ticking appropriate box.
- The value should also be entered on assignment completion page of therespective Lab course.

5. **Instructions to the Lab administrator:**

   You have to ensure appropriate hardware and software is made available to each student. The operating system and software requirements on server side and also client-side areas given below:

   - Server and Client Side- (Operating System) Linux/Windows
   - RAM: Minimum 8 GB
   - JDK
   - Android Studio

Assignment Completion Sheet

Sr.No.	Assignment Name	Marks (Out of 5)	Signature
1	Introduction to Android		
2	Activities and Intents		
3	Android User Interface		
4	Designing User Interface with Views and Menus		
5	SQLite Database		
	Total Marks (Out of 25)		
	Total Marks (Out of 15)		

This is to certify that Mr./Ms. ____ has successfully completed the CA-354-MJP Practical based on Mobile Application Development course work and has scored ________marks out of 15.

Instructor **Head of the Department**

Internal Examiner **External Examiner**

Assignment No: 1 Introduction to Android

Objectives

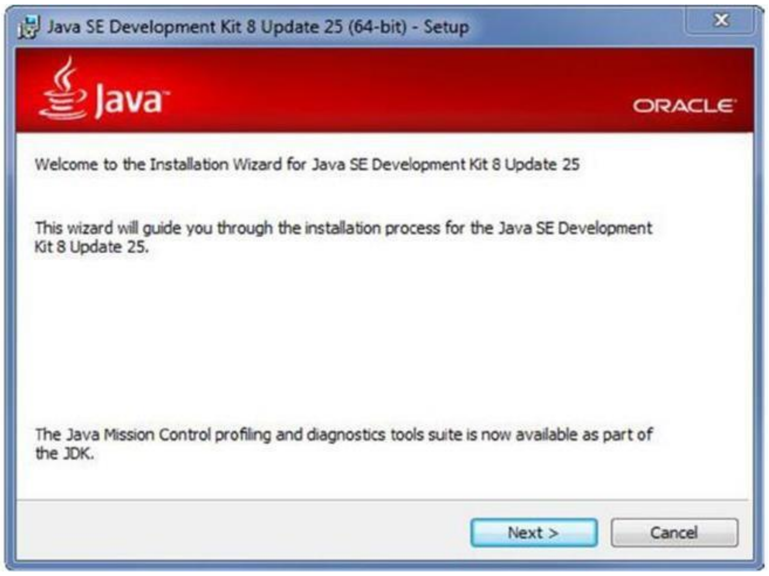
- Study Android Studio installation.
- Create basic android application.

Reading

- You should read the following topics before starting this exercise:
  1. Java Installation
  2. Android Studio
  3. Android Directory Structure

Ready Reference

- Android Studio uses the Java tool chain to build, so you need to make sure that you have the Java Development Kit (JDK) installed on your computer before you start using Android Studio.
- It's quite possible that you already have the JDK installed on your computer, particularly if you're a seasoned Android or Java developer.
- If you already have the JDK installed on your computer, and you're running JDK version 1.6 or higher, then you can skip this section. However, you may want to download, install, and configure the latest JDK anyway.
- You can download the JDK from the following Oracle site:
  www.oracle.com/technetwork/java/javase/downloads/index.html



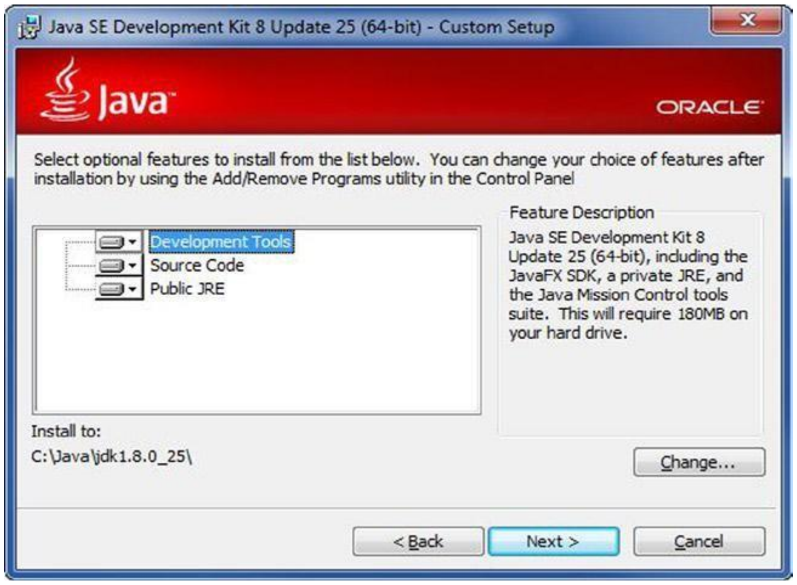
Welcome to the Installation Wizard for Java SE Development Kit 8 Update 25

This wizard will guide you through the installation process for the Java SE Development Kit 8 Update 25.

The Java Mission Control profiling and diagnostics tools suite is now available as part of the JDK.

Figure 1.1 : Installation Wizard for the JDK on Windows

Make a note of where you are installing your JDK. Follow the prompts until the installation is complete. If prompted to install the Java Runtime Edition (JRE), choose the same directory where you installed the JDK.


Java SE Development Kit 8 Update 25 (64-bit) - Custom Setup

Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel

Development Tools
Source Code
Public JRE

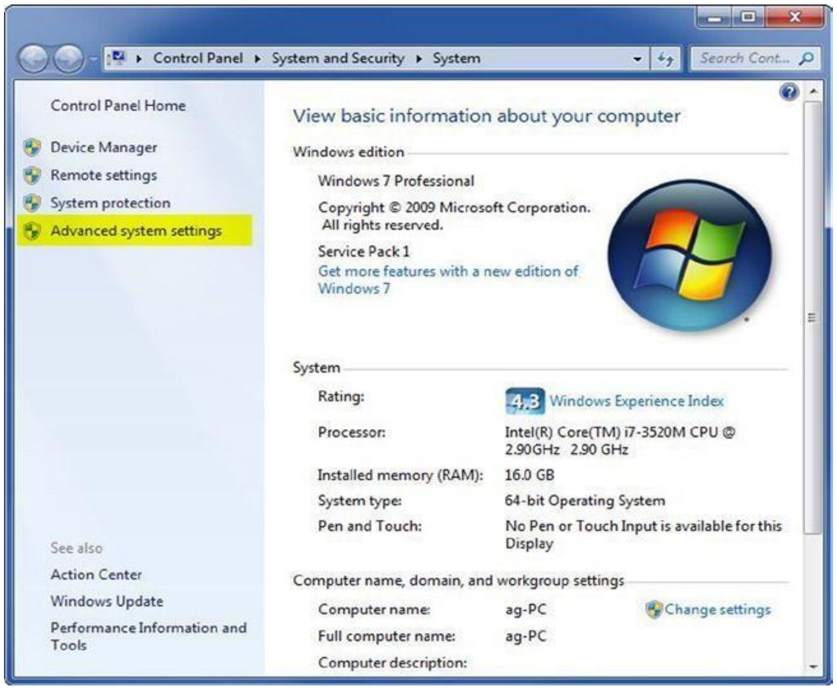
Feature Description
Java SE Development Kit 8 Update 25 (64-bit), including the JavaFX SDK, a private JRE, and the Java Mission Control tools suite. This will require 180MB on your hard drive.

Install to:
C:\Java\jdk1.8.0_25\

Figure 1.2: Select the JDK installation directory

Configuring Environmental Variables on Windows

This section shows you how to configure Windows so that the JDK is found by Android Studio. On a computer running Windows, hold down the Windows key and press the Pause key to open the System window. Click the Advanced System Settings option, shown in Figure -.



View basic information about your computer

Windows edition
Windows 7 Professional
Copyright © 2009 Microsoft Corporation. All rights reserved.
Service Pack 1
Get more features with a new edition of Windows 7

System
Rating: 4.3 Windows Experience Index
Processor: Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz 2.90 GHz
Installed memory (RAM): 16.0 GB
System type: 64-bit Operating System
Pen and Touch: No Pen or Touch Input is available for this Display

Computer name, domain, and workgroup settings
Computer name: ag-PC Change settings
Full computer name: ag-PC
Computer description:

Figure 1.3: Windows System window

- Click the Environmental Variables button, In the System Variables list along the bottom, shown in Figure 1-7, navigate to the JAVA_HOME item. If the JAVA_HOME item does not exist, click New to create it. Otherwise,click Edit.
- Clicking either New or Edit displays a dialog box. Be sure to type JAVA_HOME in the Variable Name field. Inthe Variable Value field, type the location where you installed the JDK earlier (less any trailing slashes). Now click OK.

Android Studio

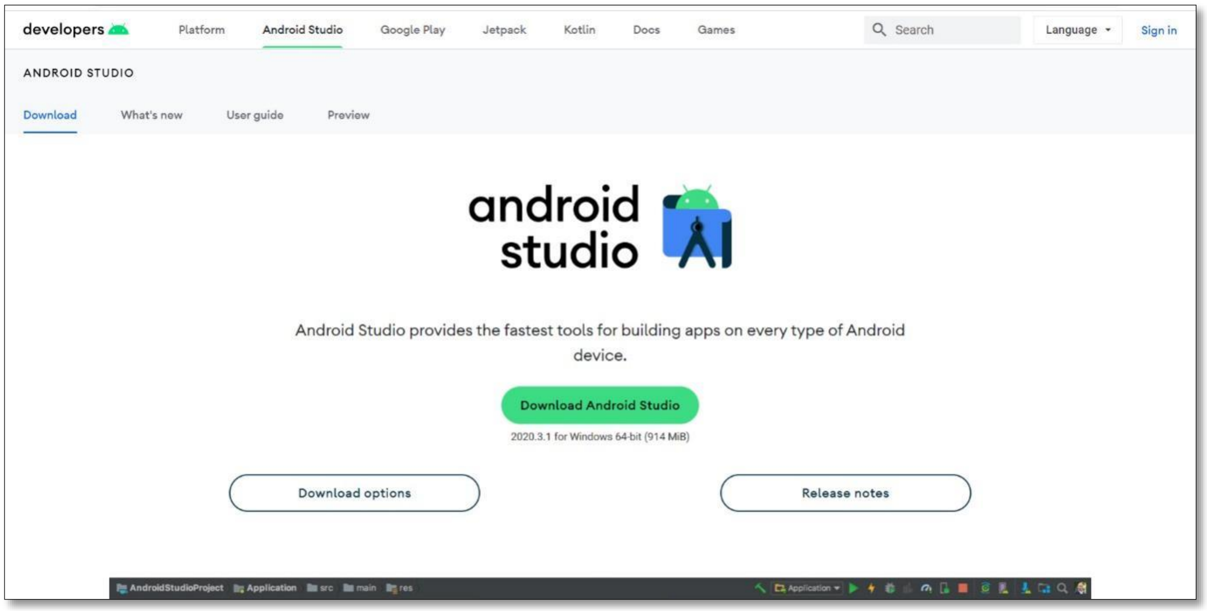
Android Studio is the official Integrated Development Environment (IDE) for Android app development, based onIntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Downloading Android Studio is straightforward. Point your browser to this site:

https://developer.android.com/studio

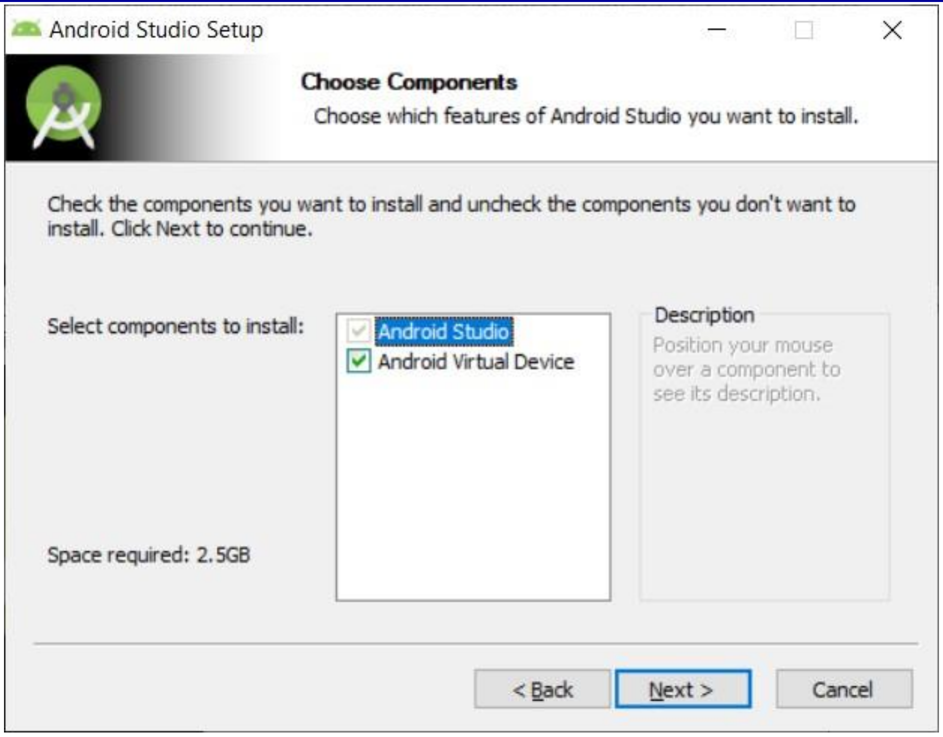
Now click the large green Download Android Studio for your OS button,



Android Studio provides the fastest tools for building apps on every type of Android device.

Figure 1.4: Download Android Studio

After the Installation Wizard begins, move through its screens by clicking the Next buttons until you reach the Choose Components screen. There, select all the component check boxes, shown in Figure. Then click Next. Agree to the terms and conditions once again. When you reach the Configuration Settings: Install Locations screen, shown in Figure 1, select the locations for Android Studio and the Android SDK. To be consistent, we chose to install Android Studio in C:\Java\astudio\ and the Android SDK in C:\Java\asdk\.


Android Studio Setup
Choose Components
Choose which features of Android Studio you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

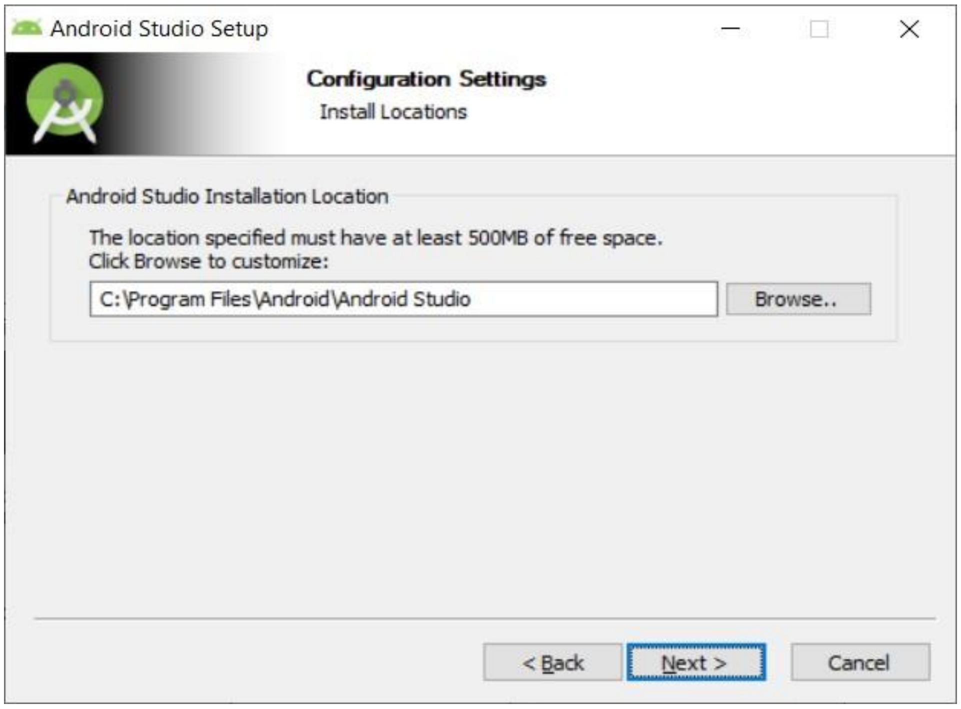
Select components to install:

- [✓] Android Studio
- [✓] Android Virtual Device

Description
Position your mouse over a component to see its description.

Space required: 2.5GB

Figure 1.5 : Choose components



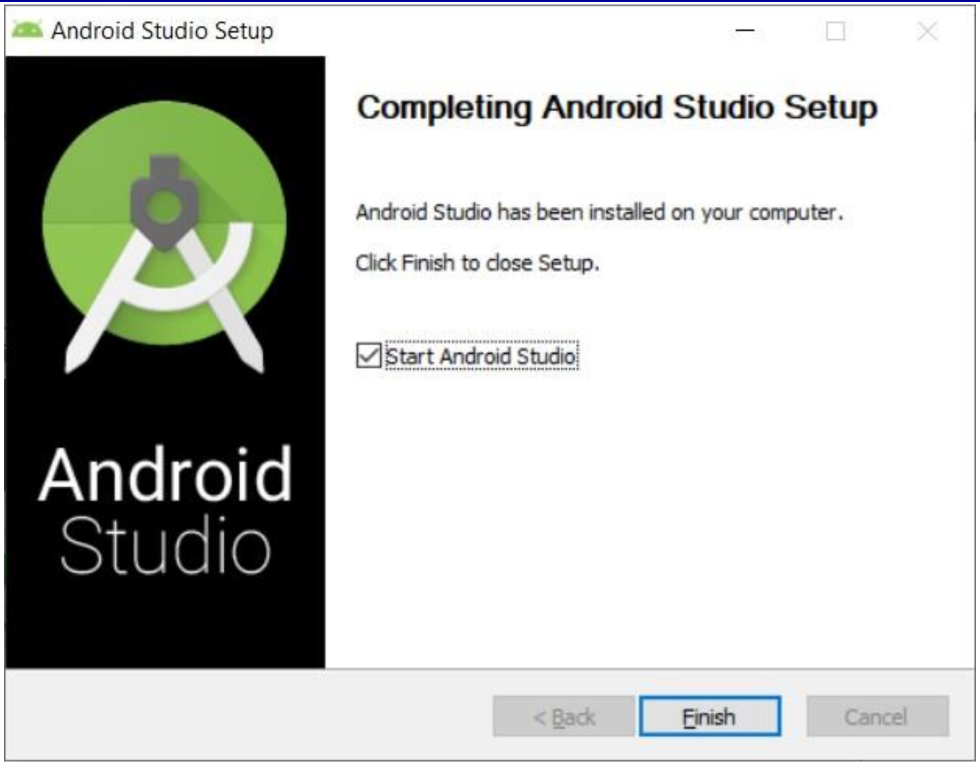
Android Studio Installation Location

The location specified must have at least 500MB of free space.
Click Browse to customize:

C:\Program Files\Android\Android Studio

Figure 1.6: Select locations for Android Studio and the SDK

Click through several Next buttons as you install both Android Studio and the Android SDK. You should eventually arrive at the Completing the Android Studio Setup screen, shown in Figure. The Start Android Studio check box enables Android Studio to launch after you click Finish. Make sure the check box is selected, and then go ahead and click Finish, and Android Studio will launch. Please note that from here on out, you will need to navigate to either the desktop icon or the Start menu to launch Android Studio.


Android Studio Setup

Completing Android Studio Setup

Android Studio has been installed on your computer.

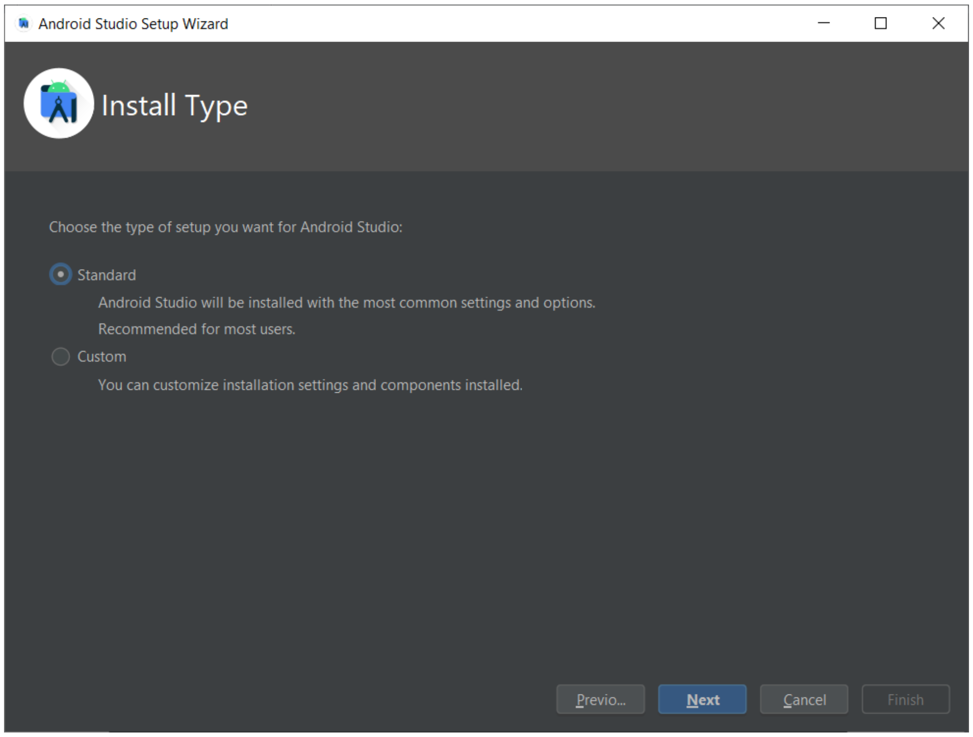
Click Finish to close Setup.

[✓] Start Android Studio

Figure 1.7: Completing the Android Studio setup

Now that Android Studio 2 is installed, you need to adjust the settings and options using the following steps:

Click Continue at the Welcome screen and choose Standard from the Install Type selection screen shown in Figure. Click Next to continue.


Android Studio Setup Wizard

Install Type

Choose the type of setup you want for Android Studio:

(●) Standard

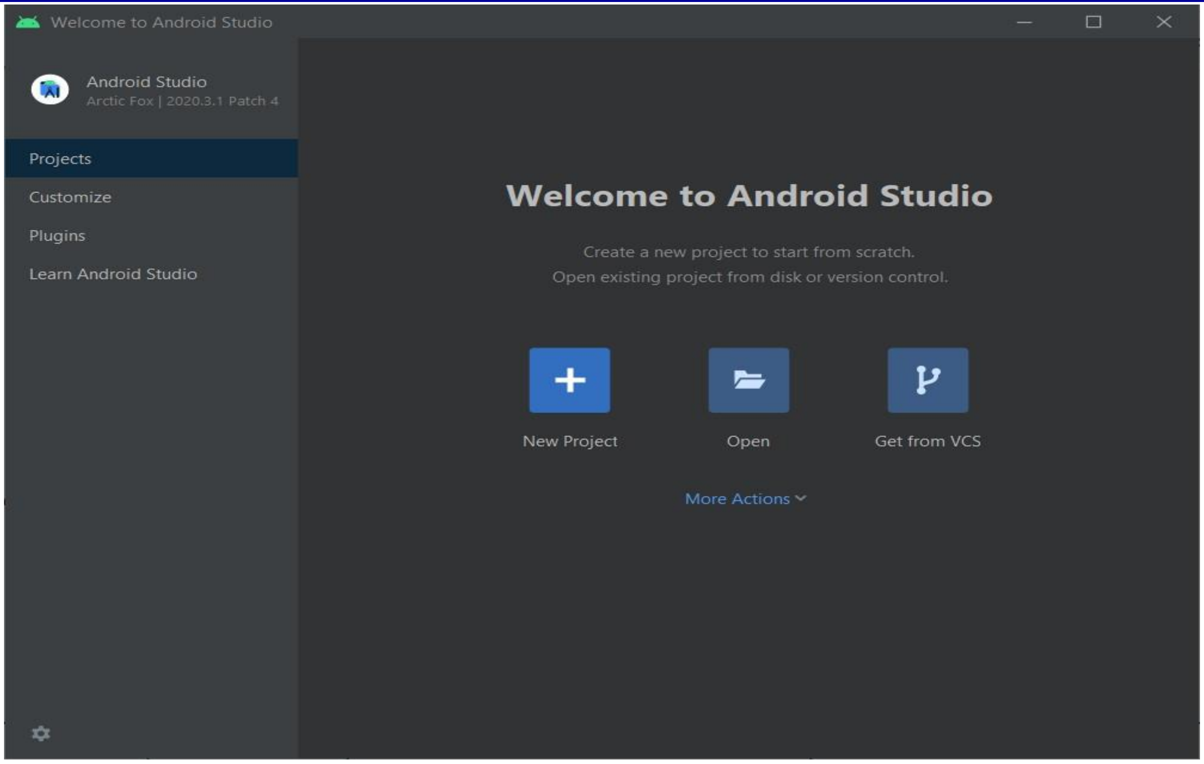
Android Studio will be installed with the most common settings and options.

Recommended for most users.

( ) Custom

You can customize installation settings and components installed.

Click Finish on the Verify Settings screen, and Android Studio 2 finalizes the setup process. You know the process is complete when you are greeted with the Welcome to Android Studio screen (see Figure).


Welcome to Android Studio
Android Studio
Arctic Fox | 2020.3.1 Patch 4
Projects
Customize
Plugins
Learn Android Studio
Welcome to Android Studio
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
More Actions

Now that Android Studio is set up, it’s time to install the latest and greatest Android SDK.

2. Create "Hello World" application

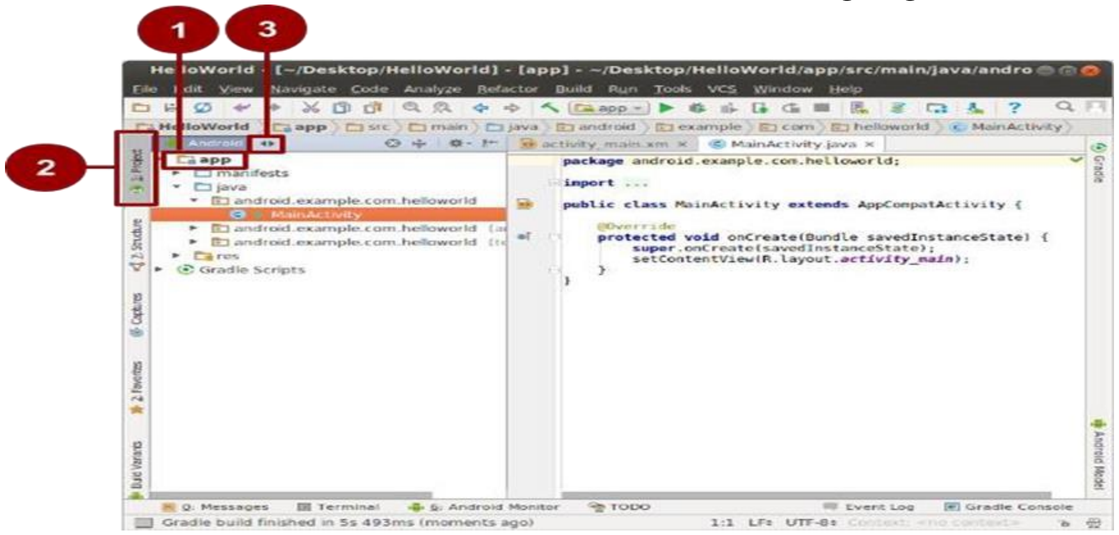
1. Launch Android Studio if it is not already opened.
2. In the main Welcome to Android Studio window, click "Start a new Android Studio project".
3. In the New Project window, give your application an Application Name, such as "Hello World".
4. Verify the Project location, or choose a different directory for storing your project.
5. Choose a unique Company Domain.
    - Apps published to the Google Play Store must have a unique package name. Since domains are unique, prepending your app's name with your or your company's domain name is going to result in a unique package name.
    - If you are not planning to publish your app, you can accept the default example domain. Be aware that changing the package name of your app later is extra work.

6. Verify that the default Project location is where you want to store your Hello World app and other Android Studio projects, or change it to your preferred directory. Click Next.
7. On the Target Android Devices screen, "Phone and Tablet" should be selected. And you should ensure that API 15: Android 4.0.3 IceCreamSandwich is set as the Minimum SDK. (Fix this if necessary.)
    - At the writing of this book, choosing this API level makes your "Hello World" app compatible with 97% of Android devices active on the Google Play Store.
    - These are the settings used by the examples in this book.
8. Click Next.
9. If your project requires additional components for your chosen target SDK, Android Studio will install them automatically. Click Next.
10. Customize the Activity window. Every app needs at least one activity. An activity represents a single screen with a user interface and Android Studio provides templates to help you get started. For the Hello World project, choose the simplest template (as of this writing, the "Empty Activity" project template is the simplest template) available.

11. It is a common practice to call your main activity MainActivity. This is not a requirement.
12. Make sure the Generate Layout file box is checked (if visible).
13. Make sure the Backwards Compatibility (App Compat) box is checked.
14. Leave the Layout Name as activity_main. It is customary to name layouts after the activity they belong to. Accept the defaults and click Finish.
15. Builds your project with Gradle (this may take a few moments). Android Studio uses Gradle as its build system. See the Configure your build developer page for more information.
16. Opens the code editor with your project. And it displays a tip of the day.

- Android Studio offers many keyboard shortcuts, and reading the tips is a great way to learn them over time.

17. The Android Studio window should look similar to the following diagram:


1
3
2
HelloWorld - [~/Desktop/HelloWorld] - [app] - ~/Desktop/HelloWorld/app/src/main/java/andro
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HelloWorld app src main java android example com helloworld MainActivity
Android
app
manifests
java
android.example.com.helloworld
MainActivity
android.example.com.helloworld
android.example.com.helloworld
res
Gradle Scripts
activity_main.xml
MainActivity.java
1: Project
Z: Structure
Captures
2: Favorites
Build Variants
Gradle
Android Model
0: Messages
Terminal
6: Android Monitor
TODO
Event Log
Gradle Console

```
package android.example.com.helloworld;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

```
Gradle build finished in 5s 493ms (moments ago)
```

18. You can look at the hierarchy of the files for your app in multiple ways.
    - Click on the Hello World folder to expand the hierarchy of files (1),
    - Click on Project (2).
    - Click on the Android menu (3).
    - Explore the different view options for your project.

3.Explore the project structure

3.1 Explore the project structure and layout

In the **Project > Android view** of your previous task, there are three top-level folders below your **app** folder: **manifests**, **java**, and **res**.

1. Expand the **manifests** folder.
This folder contains **AndroidManifest.xml.** This file describes all of the components of your Android app and is read by the Android run-time system when your program is executed.

2. Expand the **java** folder. All your Java language files are organized in this folder. The **java** folder contains three subfolders:
    - **com.example.hello.helloworld (or the domain name you have specified):** All the files for a package are in a folder named after the package. For your Hello World application, there is one package and it only contains MainActivity.java (the file extension may be omitted in the Project view).
    - **com.example.hello.helloworld(androidTest):** This folder is for your instrumented

tests, and starts out with a skeleton test file.

   - **com.example.hello.helloworld(test):** This folder is for your unit tests and starts out with an automatically created skeleton unit test file.

3. Expand the **res** folder. This folder contains all the resources for your app, including images, layout files, strings, icons, and styling. It includes these subfolders:
   - **drawable**: Store all your app's images in this folder.
   - **layout**: Every activity has at least one layout file that describes the UI in XML. For Hello World, this folder contains activity_main.xml.
   - **mipmap**: Store your launcher icons in this folder. There is a sub-folder for each supported screen density. Android uses the screen density, that is, the number of pixels per inch to determine the required image resolution. Android groups all actual screen densities into generalized densities, such as medium (mdpi), high (hdpi), or extra-extra-extra-high (xxxhdpi). The ic_launcher.png folder contains the default launcher icons for all the densities supported by your app.
   - **values**: Instead of hardcoding values like strings, dimensions, and colors in your XML and Java files, it is best practice to define them in their respective values file. This makes it easier to change and be consistent across your app.
4. Expand the **values** subfolder within the res folder. It includes these subfolders:
   - **colors.xml**: Shows the default colors for your chosen theme, and you can add your own colors or change them based on your app's requirements.
   - **dimens.xml**: Store the sizes of views and objects for different resolutions.
   - **strings.xml**: Create resources for all your strings. This makes it easy to translate them to other languages.
   - **styles.xml**: All the styles for your app and theme go here. Styles help give your app a consistent look for all UI elements.

3.1. The Gradle build system

Android Studio uses Gradle as its build system. As you progress through these practicals, you will learn more about gradle and what you need to build and run your apps.

1. Expand the **Gradle Scripts** folder. This folder contains all the files needed by the build system.
2. Look for the **build.gradle(Module:app)** file. When you are adding app-specific dependencies, such as using additional libraries, they go into this file.

4. Create a virtual device (emulator)

In this task, you will use the Android Virtual Device (AVD) manager to create a virtual device or emulator that simulates the configuration for a particular type of Android device.
Using the AVD Manager, you define the hardware characteristics of a device and its API level, and save it as a virtual device configuration.

When you start the Android emulator, it reads a specified configuration and creates an emulated device that behaves exactly like a physical version of that device, but it resides on your computer.

Why: With virtual devices, you can test your apps on different devices (tablets, phones) with different API levels to make sure it looks good and works for most users. You do not need to depend on having a physical device available for app development.

4.1 Create a virtual device

In order to run an emulator on your computer, you have to create a configuration that describes the virtual device.

1. In Android Studio, select Tools > Android > AVD Manager, or click the AVD Manager icon in the toolbar.
2. Click the **+Create Virtual Device**…. (If you have created a virtual device before, the window shows all of your existing devices and the button is at the bottom.)

   The Select Hardware screen appears showing a list of preconfigured hardware devices. For each device, the table shows its diagonal display size (Size), screen resolution in pixels (Resolution), and pixel density (Density).

   For the Nexus 5 device, the pixel density is xxhdpi, which means your app uses the launcher icons in the xxhdpi folder of the mipmap folder. Likewise, your app will use layouts and drawables from folders defined for that density as well.
3. Choose the Nexus 5 hardware device and click **Next**.
4. On the **System Image** screen, from the **Recommended** tab, choose which version of the Android system to run on the virtual device. You can select the latest system image.

   There are many more versions available than shown in the Recommended tab. Look at the **x86 Images** and **Other Images** tabs to see them.

5. If a **Download** link is visible next to a system image version, it is not installed yet, and you need to download it. If necessary, click the link to start the download, and click **Finish** when it's done.
6. On **System Image** screen, choose a system image and click **Next**.

Verify your configuration, and click **Finish**. (If the **Your Android Devices** AVD Manager window stays open, you can go ahead and close it.)

5. <u>Run your app on an emulator</u>

In this task, you will finally run your Hello World app.

5.1 Run your app on an emulator

1. In Android Studio, select **Run > Run app** or click the **Run icon** ▶ in the toolbar.
2. In the **Select Deployment Target** window, under **Available Emulators**, select Nexus **5 API 23** and click **OK**.

You should see the Hello World app as shown in the following screenshot.

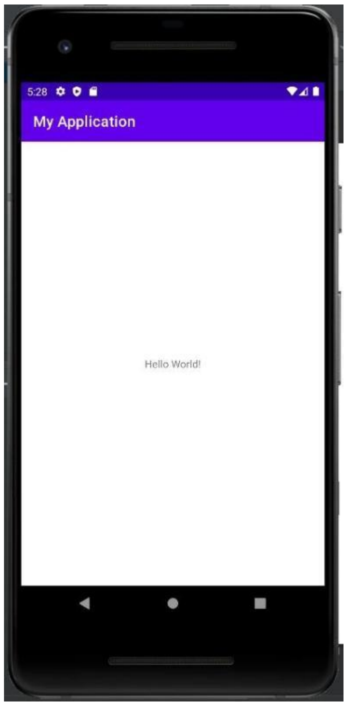


Figure: Display Output on Emulator

Set A

1. Create a simple Android application that displays a "Hello World" message using a Linear Layout.
2. Create an Android application to display your personal information such as Name, Roll Number, Class, and Email ID. Apply different text sizes, text colors, and font families to each text view.
3. Set up an Android Virtual Device (AVD) on your machine using Android Studio.
4. Execute and run the "Hello World" Android application on a physical Android device using Android Studio.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well done [ ]

Assignment No: 2 Activities and Intents

Objectives -:

- To study how to use Activities, Fragments and Intents in your application.
- To study how to link Activities and interaction between Fragments.

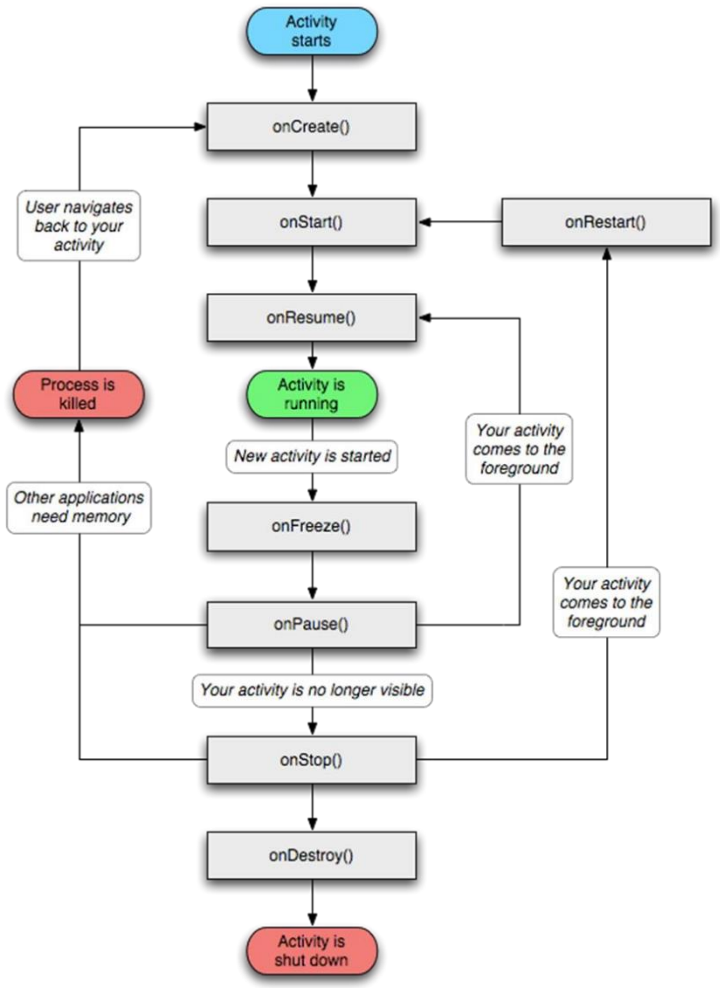
Reading-:

- You should read the following topics before starting this exercise:
  1. Activity, Activity Lifecycle
  2. Linking Activities using Intent.

Ready Reference-:

1. **Activity**-: An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View).

2. **Activity Lifecycle-:** Activities in the system are managed as an activity stack. When a new activity is started, it is placed on the top of the stack and becomes the running activity - the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.


Activity starts
onCreate()
User navigates back to your activity
onStart()
onRestart()
onResume()
Process is killed
Activity is running
Your activity comes to the foreground
New activity is started
Other applications need memory
onFreeze()
Your activity comes to the foreground
onPause()
Your activity is no longer visible
onStop()
onDestroy()
Activity is shut down

The lifecycle is shown diagrammatically in Figure 2-1.

Your activity monitors and reacts to these events by instantiating methods that override the Activity class methods for each event:

onCreate

Called when your activity is first created. This is the place you normally create your views, open any persistent datafiles your activity needs to use, and in general initialize your activity. When calling onCreate, the Android framework is passed a Bundle object that contains any activity state saved from when the activity ran before.

onStart

Called just before your activity becomes visible on the screen. Once onStart completes, if your activity can become the foreground activity on the screen, control will transfer to onResume. If the activity cannot become the foreground activity for some reason, control transfers to the onStop method.

onResume

Called right after onStart if your activity is the foreground activity on the screen. At this point your activity is running and interacting with the user. You are receiving keyboard and touch inputs, and the screen is displaying your user interface. onResume is also called if your activity loses the foreground to another activity, and that activity eventually exits, popping your activity back to the foreground. This is where your activity would start (or resume) doing things that are needed to update the user interface (receiving location updates or running an animation, for example).

onPause

Called when Android is just about to resume a different activity, giving that activity the foreground. At this point your activity will no longer have access to the screen, so you should stop doing things that consume battery and CPU cycles unnecessarily. If you are running an animation, no one is going to be able to see it, so you might as well suspend it until you get the screen back. Your activity needs to take advantage of this method to store any state that you will need in case your activity gains the foreground again—and it is not guaranteed that your activity will resume.

onStop

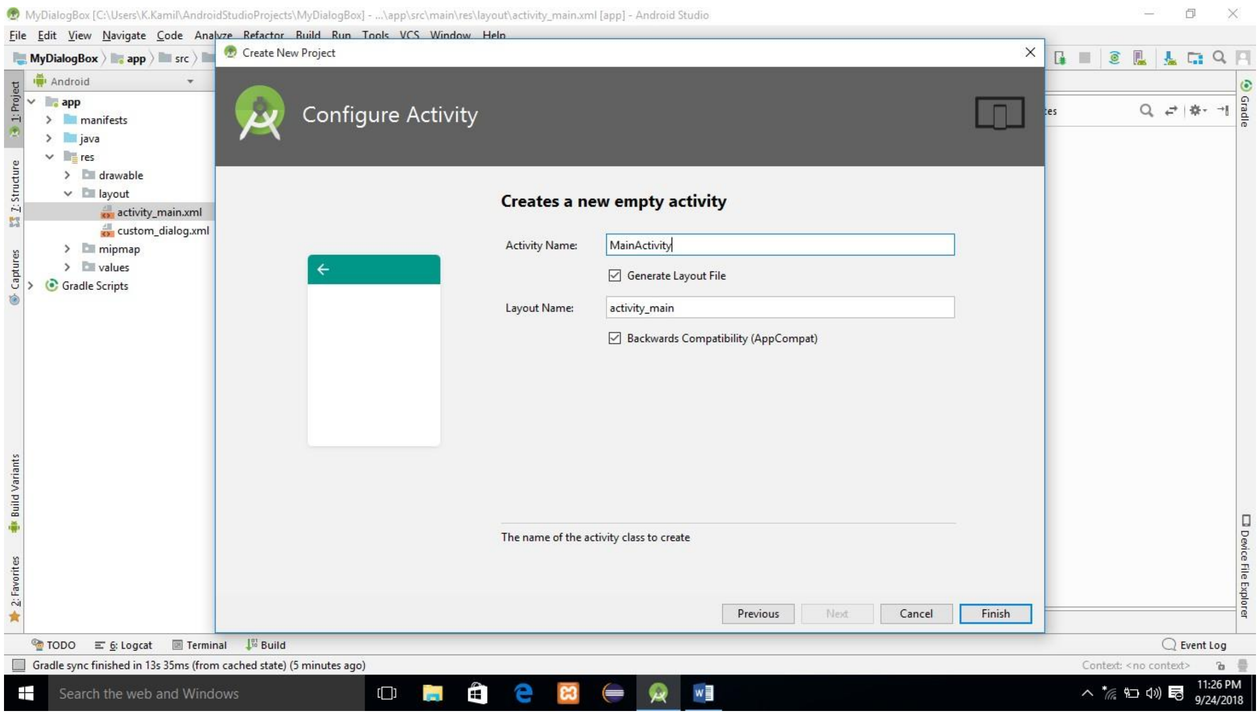
Called when your activity is no longer visible, either because another activity has taken the foreground or because your activity is being destroyed.

onDestroy

The last chance for your activity to do any processing before it is destroyed. Normally you'd get to this point because the activity is done and the framework called its finish method. But as mentioned earlier, the method might be called because Android has decided it needs the resources your activity is consuming.

The best way to understand the various stages experienced by an activity is to in android studio select File->New->create a new project, implement the various events, and then subject the activity to various user interactions.

Step 1:Using Android Studio, create a new Android project and name it as shown in Figure 2.


Configure Activity

Creates a new empty activity

Activity Name: MainActivity

☑ Generate Layout File

Layout Name: activity_main

☑ Backwards Compatibility (AppCompat)

The name of the activity class to create

Gradle sync finished in 13s 35ms (from cached state) (5 minutes ago)

Step 2: In the MainActivity.java file, add the following statements in bold

```
package com.example.kkamil.learnfromkamil;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.util.Log;

public class MainActivityextends AppCompatActivity {

    String tag="Events";

@Override

public void onCreate(Bundle savedInstanceState) {

  super.onCreate(savedInstanceState);

  setContentView(R.layout.activity_main);

  Log.d(tag,"In on Create Event");

   }

public void onStart()
   {
super.onStart();
Log.d(tag, "In the onStart() event");
  }


public void onRestart()
   {
```

```
super.onRestart();
Log.d(tag, "In the onRestart() event");
    }

public void onResume()
    {
super.onResume();
Log.d(tag, "In the onResume() event");
    }

public void onPause()
    {
super.onPause();
Log.d(tag, "In the onPause() event");
    }

public void onStop()
    {
super.onStop();
Log.d(tag, "In the onStop() event");
    }

public void onDestroy()
    {
super.onDestroy();
Log.d(tag, "In the onDestroy() event");
    }
}
```

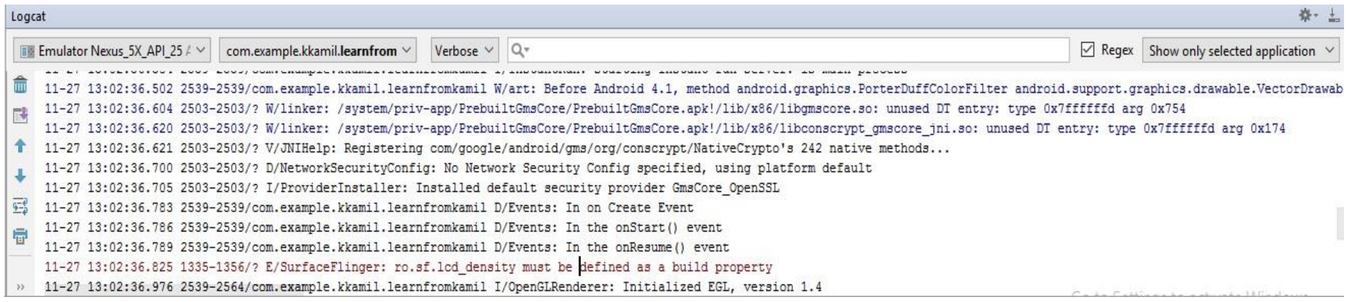
Step 3: Press F11 to debug the application on the Android Emulator

Step 4: When the activity is first loaded, you should see the following in the LogCat window (click on the Debug perspective; see also Figure):

11-27 13:02:36.783 2539-2539/com.example.kkamil.learnfromkamil D/Events: In onCreate() event

11-27 13:02:36.786 2539-2539/com.example.kkamil.learnfromkamil D/Events: In the onStart()event

11-27 13:02:36.789 2539-2539/com.example.kkamil.learnfromkamil D/Events: In the onResume()



```
11-27 13:02:36.502 2539-2539/com.example.kkamil.learnfromkamil W/art: Before Android 4.1, method android.graphics.PorterDuffColorFilter android.support.graphics.drawable.VectorDrawab
11-27 13:02:36.604 2503-2503/? W/linker: /system/priv-app/PrebuiltGmsCore/PrebuiltGmsCore.apk!/lib/x86/libgmscore.so: unused DT entry: type 0x7ffffffd arg 0x754
11-27 13:02:36.620 2503-2503/? W/linker: /system/priv-app/PrebuiltGmsCore/PrebuiltGmsCore.apk!/lib/x86/libconscrypt_gmscore_jni.so: unused DT entry: type 0x7ffffffd arg 0x174
11-27 13:02:36.621 2503-2503/? V/JNIHelp: Registering com/google/android/gms/org/conscrypt/NativeCrypto's 242 native methods...
11-27 13:02:36.700 2503-2503/? D/NetworkSecurityConfig: No Network Security Config specified, using platform default
11-27 13:02:36.705 2503-2503/? I/ProviderInstaller: Installed default security provider GmsCore_OpenSSL
11-27 13:02:36.783 2539-2539/com.example.kkamil.learnfromkamil D/Events: In on Create Event
11-27 13:02:36.786 2539-2539/com.example.kkamil.learnfromkamil D/Events: In the onStart() event
11-27 13:02:36.789 2539-2539/com.example.kkamil.learnfromkamil D/Events: In the onResume() event
11-27 13:02:36.825 1335-1356/? E/SurfaceFlinger: ro.sf.lcd_density must be defined as a build property
11-27 13:02:36.976 2539-2564/com.example.kkamil.learnfromkamil I/OpenGLRenderer: Initialized EGL, version 1.4
```

3. **Intent-:** An Android Intent is an abstract description of an operation to be performed. The intent itself, an Intent object, is a passive data structure holding an abstract description of an operation to be performed.

Method	**Description**
Context.startActivity()	The Intent object is passed to this method to launch a new activity or get an existing activity to do something new
Context.startService()	The Intent object is passed to this method to initiate a service or deliver new instructions to an ongoing service.
Context.sendBroadcast()	The Intent object is passed to this method to deliver the message to all interested broadcast receivers.

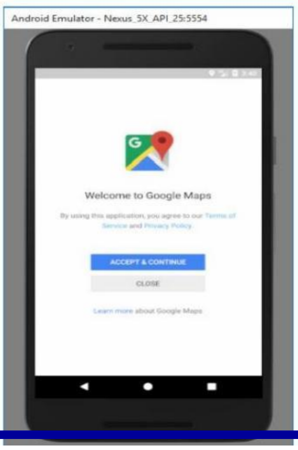
Example:

This example shows how you can open a URL programmatically in the built-in web browser rather than within your application. This allows your app to open up a webpage without the need to include the INTERNET permission in your manifest file.

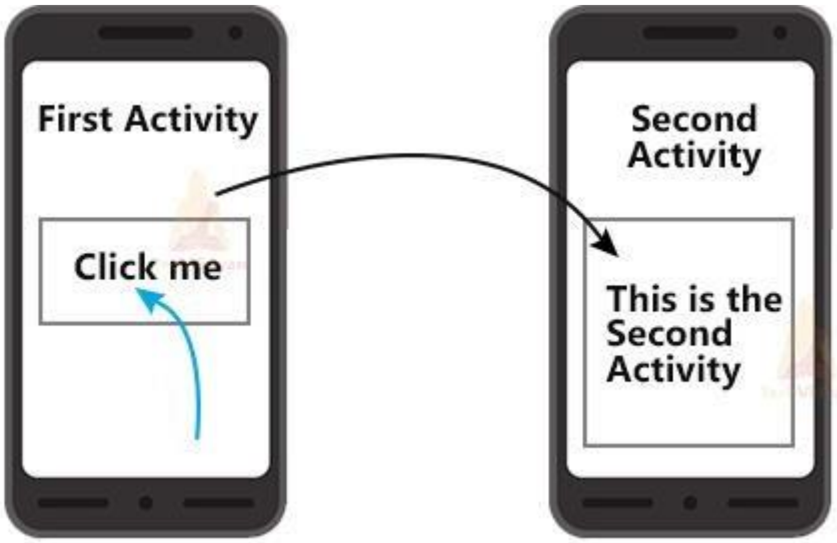
```
public void onBrowseClick(View v) {
    String url = "http://www.google.com";
    Uri uri = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    // Verify that the intent will resolve to an activity
    if (intent.resolveActivity(getPackageManager()) != null) {
        // Here we use an intent without a Chooser unlike the next example
        startActivity(intent);
    }
```

- **Types Of Intent :-**
  1) **Implicit Intent :** These intents do not name a target and the field for the component name is left blank. Implicit intents are often used to activate components in other application.
     Example: Open Google MAP using Implict Intent

```
b3 = (Button) findViewById(R.id.button3);
b3.setOnClickListener(new View.OnClickListener()
 {
public void onClick(View arg0){
 Intent i = new
Intent(Intent.ACTION_VIEW, Uri.parse("geo:37.827500,-
122.481670"));
startActivity(i);  }

  });
```


Android Emulator - Nexus_5X_API_25:5554
Welcome to Google Maps
ACCEPT & CONTINUE
CLOSE
Learn more about Google Maps

2) **Explicit Intent:** Explicit intent going to be connected internal world of application, suppose if you want to connect one activity to another activity, we can do this quote by explicit intent, below image is connecting first activity to second activity by clicking button



Above code will give result as shown below:

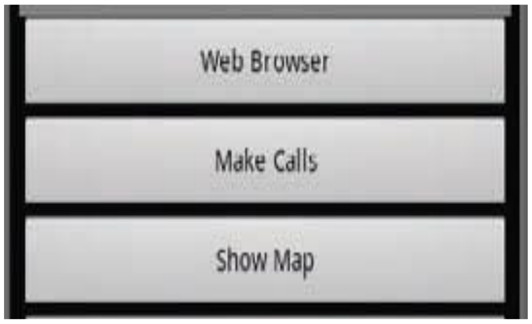
Example:

```
// Explicit Intent by specifying its class name

Intent i=new Intent(FirstActivity.this,SecondActivity.class)

//Starts TargetActivity

startActivity(i);
```

Lab Assignments: -

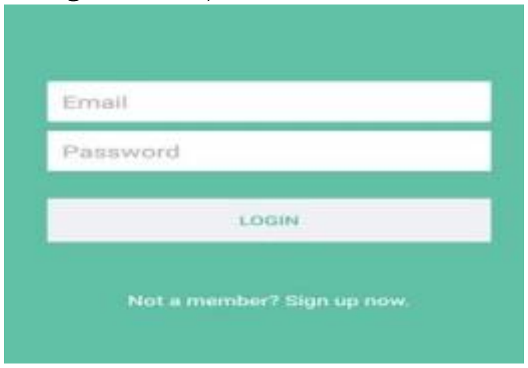
SET A

1. Create a Simple Application Which Shows Life Cycle of Activity.
2. Create an Android application with three buttons as shown in the given image and apply implicit intents for different actions.



SET B

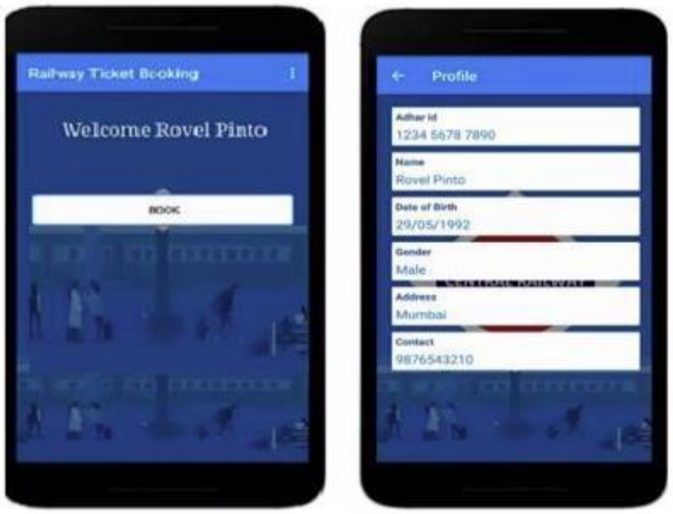
1. Create simple application with Login Screen. On successful login, gives message go to next Activity (Without Using Database).



2. Create First Activity to accept information like Student First Name, Middle Name, Last Name, Date of birth, Address, Email ID and display all information on Second Activity when user click on Submit button.

SET C

1. Design Following Screens Using Intents. On second activity take Button. On clicking it, it should Show Information of profile on Third activity. (Without Using Database)



Adhar id
1234 5678 7890
Name
Rovel Pinto
Date of Birth
29/05/1992
Gender
Male
Address
Mumbai
Contact
9876543210

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment No: 3 Android User Interface

Objective

- Study how to interact with mobile hardware.
- Study how to create user interface in Android.

Reading

- You should read the following topics before starting this exercise:
  1. Views and ViewGroups.
  2. Different layouts available in android.
  3. Understanding concept of screen

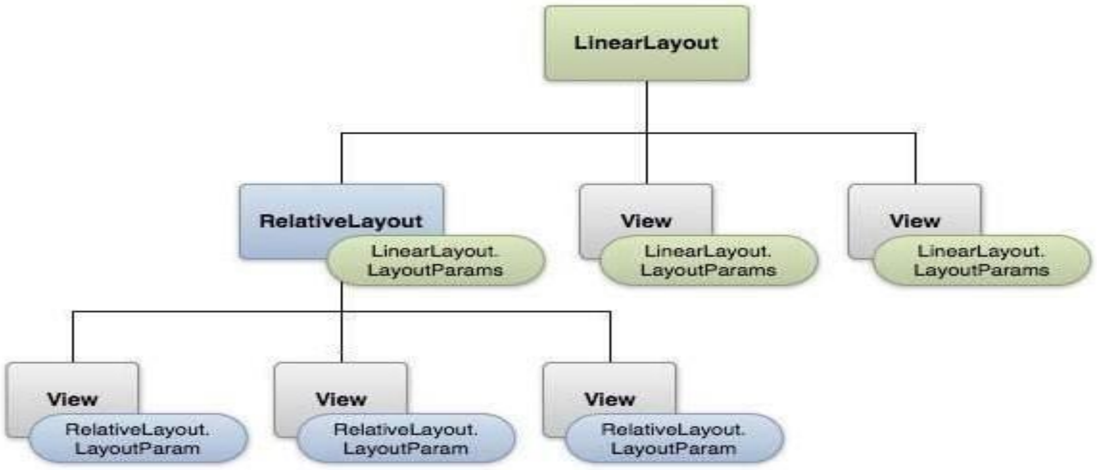
Ready Reference

Views and ViewGroup

Views - View is a basic building block of UI (User Interface) in android. A view is a small rectangular box that responds to user inputs. Eg: EditText, Button, CheckBox, etc.

Views Groups - ViewGroup is an invisible container of other views (child views) and other ViewGroup. Eg: LinearLayout is a ViewGroup that can contain other views in it

At third level we have different layouts which are subclasses of View Group class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using View/View Group objects or you can declare your layout using simple XML file main_layout.xml which is located in the res/layout folder of your project.



Types of Layouts

- **Types of layouts**

1. **Linear Layout:** Linear Layout is a view group that aligns all children in a single direction, vertically or horizontally.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical">
        <TextView
                android:id="@+id/text"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="This is a TextView"/>
</LinearLayout>
```

2. **Absolute Layout:** Absolute Layout enables you to specify the exact location of its children.

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <Button android:layout_width="100dp"
                android:layout_height="wrap_content"
                android:text="OK"
                android:layout_x="50px"
                android:layout_y="361px"/>
</AbsoluteLayout>
```

3. **Table Layout:** Table Layout is a view that groups views into rows and columns.

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TableRow android:layout_width="fill_parent"
                android:layout_height="fill_parent">
        <TextView android:text="First Name"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="1"/>
        </TableRow>
        <TableRow android:layout_width="fill_parent"
                android:layout_height="fill_parent">
        <Button android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Submit"
                android:id="@+id/button"
                android:layout_column="2"/>
        </TableRow>
</TableLayout>
```

4. **RelativeLayout:** Relative Layout is a view group that displays child views in relative positions.

```
<RelativeLayout
                xmlns:android="http://schemas.android.com/apk/res
                /android"android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:paddingLeft="16dp"
                android:paddingRight="16dp">
```

```
        <EditText android:id="@+id/name"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:hint="@string/reminder"/>
        <LinearLayout android:orientation="vertical"
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    android:layout_alignParentStart="true"
                    android:layout_below="@+id/name">
        <Button android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="New Button"
               android:id="@+id/button"/>
        <Button android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="New Button"
               android:id="@+id/button2"/>
        </LinearLayout>
</RelativeLayout>
```

5. FragmentLayout: FrameLayout is a placeholder on screen that you can use to display a single view

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent">
        <TextView android:text="Frame Demo"
                android:textSize="30px"
                android:layout_height="fill_parent"
                android:layout_width="fill_parent"
                android:gravity="center"/>
</FrameLayout>
```

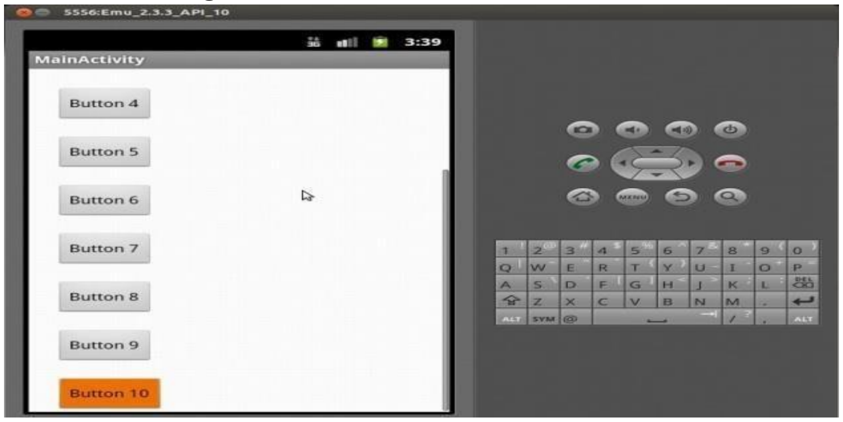
6. **ScrollLayouts:** When an app has layout content that might be longer than the height of the deviceand that content should be vertically scrollable, then we need to use a Scroll View.

```
<ScrollView

            xmlns:android="http://schemas.android.com/apk/res/andro

            id"android:layout_width="fill_parent"

            android:layout_height="wrap_content"

            android:orientation="vertical"

            android:padding="10dp"

            android:fillViewport="false">

    <LinearLayout

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:orientation="vertical">

    <Button android:id="@+id/button"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:text="KNOW MORE"/>

    <TextView android:id="@+id/textView"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:text="@string/title"

    android:textAppearance="?android:attr/textAppearanceLarge"/>

<TextView android:id="@+id/textView2"

           android:textAppearance="?android:attr/textAppearanceSmall"/>

    </LinearLayout>

    </ScrollView>
```

Lab Assignment

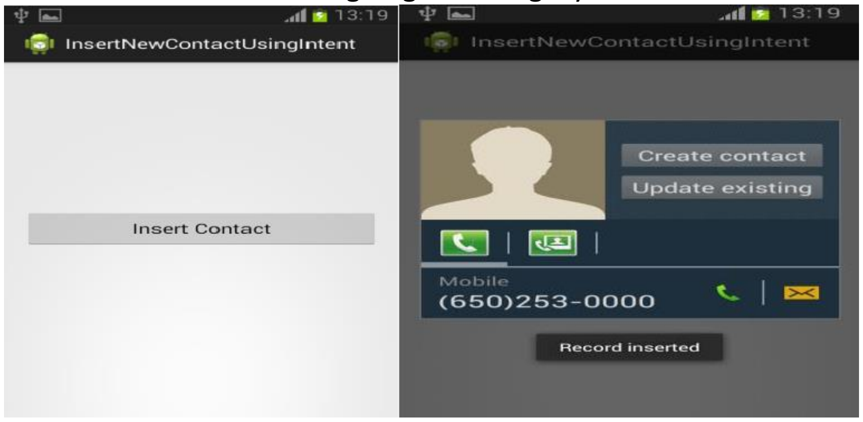
SET A

1. Create a custom "Contact" layout to hold multiple pieces of information, including:Photo, Name, Contact Number, E-mail id.
2. Create following Vertical Scroll View Creation in Android.

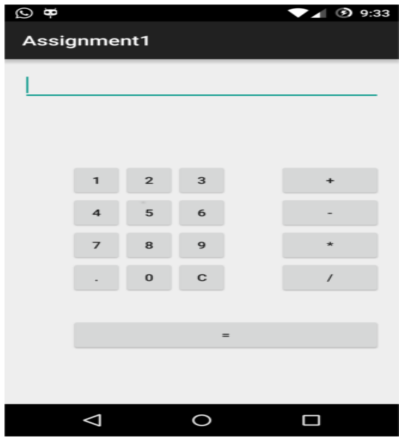


SET B

1. Create new contact for designing following layout



2. Create the simple calculator shown below also perform appropriate operation



SET C

1. Create an application to accept two numbers from the user, and displays them, but reject input if both numbers are greater than 10 and asks for two new numbers.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment No: 4 Designing User Interface with Views and Menus

Objectives

- How to create and handle the views in your application.
- How to display Pictures and menus in android

Reading

You should read the following topics before starting this exercise:

1. UI components available in android.
2. Use of List and Spinner view
3. Views available to create menus

Ready Reference

Views

View is the basic building block of UI(User Interface) in android. View refers to the android.view.View class, which is the super class for all the GUI components like Text View, ImageView, and Button etc.View class extends Object class and implements Drawable.Callback, KeyEvent.Callback and Accessibility Event Source.

View can be considered as a rectangle on the screen that shows some type of content. It can be an image, a piece of text, a button or anything that an android application can display. The rectangle here is actually invisible, but every view occupies a rectangle shape.

Types of View

TextView: TextView used to create text field. A TextView displays text to the user and optionally allows them to edit it.

Attribute Name	**Description**
android:id	This is the ID which uniquely identifies the control.
android:editable	If set to true, specifies that this TextView has an input method.
android:inputType	The type of data being placed in a text field. Phone, Date, Time, Number, Password etc.
android:text	Text to display.
android:textAllCaps	If set to true, display the text in ALL CAPS.

To create TextView make following changes in ***res/layout/activity_main.xml*** file

```
<TextView android:id="@+id/text_id"

          android:layout_width="300dp"

          android:layout_height="200dp"

          android:capitalize="characters"

          android:text="hello_world"/>
```

To access the value of TextView make following changes in ***MainActivity.java*** file

```
TextView txtView = (TextView) findViewById(R.id.text_id);
```

EditText: An EditText is an overlay over TextView that configures itself to be editable.It is the predefined subclass of TextView that includes rich editing capabilities.

Attribute Name	**Description**
android:autoText	If set to true, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
android:drawableBottom	This is the drawable to be drawn below the text.

To create EditText make following changes in ***res/layout/activity_main.xml*** file

```
<EditText android:id="@+id/edittext"

         android:layout_width="fill_parent"

         android:layout_height="wrap_content"

         android:text="@string/enter_text"

         android:inputType="text" />
```

To access the value of EditText make following changes in ***MainActivity.java*** file

```
EditText eText = (EditText) findViewById(R.id.edittext);
```

Button: A Button is a Push-button which can be pressed, or clicked, by the user toperform an action.
To create Button, make following changes in ***res/layout/activity_main.xml*** file

```
<Button  android:layout_width="wrap_content"

          android:layout_height="wrap_content"

          android:text="Button"

         android:id="@+id/button" />
```

To access the value of Button, make following changes in ***MainActivity.java*** file

```
Button b1 = (Button)findViewById(R.id.button);
```

ImageButton: An ImageButton is an AbsoluteLayout which enables you to specify theexact location of its children. This shows a button with an image.

Attribute Name	**Description**
android:adjustViewBounds	Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.
android:baseline	This is the offset of the baseline within this view.
android:baselineAlignBottom	If true, the image view will be baseline aligned with based on its bottom edge.
android:cropToPadding	If true, the image will be cropped to fit within its padding.
android:src	This sets a drawable as the content of this ImageView.

To create **ImageButton** make following changes in ***res/layout/activity_main.xml*** file

```
<ImageButton android:layout_width="wrap_content"

                 android:layout_height="wrap_content"

                 android:id="@+id/imageButton"

                 android:src="@drawable/abc"/>
```

To access the value of **ImageButton** make following changes in ***MainActivity.java*** file

```
ImageButton imgButton = (ImageButton)findViewById(R.id.imageButton);
```

ToggleButton:
A toggle button allows the user to change a setting between two states.

Attribute Name	**Description**
android:disabledAlpha	This is the alpha to apply to the indicator when disabled.
android:textOff	This is the text for the button when it is not checked.
android:textOn	This is the text for the button when it is checked.

To create **ToggleButton** make following changes in ***res/layout/activity_main.xml*** file

```
<ToggleButton android:layout_width="wrap_content"

              android:layout_height="wrap_content"

              android:text="On"

              android:id="@+id/toggleButton"

              android:checked="true" />
```

To access the value of **ToggleButton** make following changes in ***MainActivity.java*** file

```
ToggleButton tg1 = (ToggleButton)findViewById(R.id.toggleButton);
```

RadioButton: A RadioButton has two states: either checked or unchecked. This allowsthe user to select one option from a set.
To create **RadioButton** make following changes in ***res/layout/activity_main.xml*** file

```
<RadioButton  android:layout_width="wrap_content"

              android:layout_height="wrap_content"
              android:text="ANDROID"

              android:id="@+id/radioButton2"

              android:checked="false" />
```

RadioGroup: A RadioGroup class is used for set of radio buttons. If we check one radio button that belongs to a radio group, it automatically unchecks any previously checked radio button within the same group.

Attribute Name	**Description**
android:checkedButton	This is the id of child radio button that should be checked by default within this radio group.

To create **RadioGroup** make following changes in ***res/layout/activity_main.xml*** file

```
<RadioGroup android:layout_width="fill_parent"

                    android:layout_height="90dp"

                    android:layout_marginTop="58dp"

                    android:id="@+id/radioGroup" >

                    <RadioButton android:layout_width="wrap_content"

                                         android:layout_height="55dp"

                                         android:text="Male"

                                         android:id="@+id/radioButton"

                                         android:checked="false" />

                   <RadioButton  android:layout_width="wrap_content"

                                         android:layout_height="wrap_content"

                                         android:text="Female"

                                         android:id="@+id/radioButton2"
                                         android:checked="false"

                                        android:layout_weight="0.13" />

  </RadioGroup>
```

To access the value of **RadioGroup** make following changes in ***MainActivity.java*** file

```
RadioGroup radioGroup = (RadioGroup)findViewById(R.id.radioGroup);

int selectedId=radioGroup.getCheckedRadioButtonId();

RadioButton radioButton = (RadioButton)findViewById(selectedId);
```

CheckBox: A CheckBox is an on/off switch that can be toggled by the user. You should use check-boxes when presenting users with a group of selectable options that are not mutually exclusive.
To create **CheckBox** make following changes in ***res/layout/activity_main.xml*** file

```
<CheckBox android:id="@+id/checkBox2"

                android:layout_width="wrap_content"

                android:layout_height="wrap_content"

                android:text="Do you like android "

                android:checked="false" />
```

To access the value of **CheckBox** make following changes in ***MainActivity.java*** file

```
CheckBox ch1 = (CheckBox)findViewById(R.id.checkBox1);
```

- **Using Menus with Views**

Menus are useful for displaying additional options that are not directly visible on the main UI of an application. There are two main types of menus in Android:

- **Options menu** — Displays information related to the current activity. In Android, you activate the options menu by pressing the MENU key.
- **Context menu** — Displays information related to a particular view on an activity. In Android, to activate a context menu you tap and hold on to it.

- **Creating the Helper Methods**

Before you go ahead and create your options and context menus, you need to create two helper methods. One creates a list of items to show inside a menu, while the other handles the event that is fired when the user selects an item inside the menu.

Example:

1. **Creating the Menu Helper Methods: --**

a. Add the following statements in the MainActivity.java file

```
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.Toast;
private void CreateMenu(Menu menu) {
            MenuItem mnu1 = menu.add(0, 0, 0, –Item 1‖);
            {
                        mnu1.setAlphabeticShortcut(‗a‘);
                        mnu1.setIcon(R.drawable.icon);
            }
            MenuItem mnu2 = menu.add(0, 1, 1, –Item 2‖);
           {
                      mnu2.setAlphabeticShortcut(‗b‘);mnu2.setIcon(R.drawable.icon);
           }
           menu.add(0, 2, 2, –Item 3‖);
}
private boolean MenuChoice(MenuItem item)
{
            switch (item.getItemId()) {
                        case 0:  Toast.makeText(this, –You clicked on Item 1‖,
                                    Toast.LENGTH_LONG).show();
                                    return true;
                        case 1:  Toast.makeText(this, –You clicked on Item 2‖,
                                     Toast.LENGTH_LONG).show();
                                    return true;
                        case 2:  Toast.makeText(this, –You clicked on Item 3‖,
                                      Toast.LENGTH_LONG).show();
                                      return true;
                        }
            return false;
        }
}
```

- **<u>Options Menu</u>**

You are now ready to modify the application to display the options menu when the user presses the MENU button on the Android device.

Example:

- **Displaying an Options Menu**

1. Using the same project created in the previous section, add the following statements intothe MainActivity.java file:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        CreateMenu(menu);
        return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
        return MenuChoice(item);
}
```

2. Press F11 to debug the application on the Android Emulator.

- **<u>Context Menu</u>**

A context menu is usually associated with a view on an activity, and it is displayed when the user long clicks an item. For example, if the user taps on a Button view and hold it for a few seconds, a context menu can be displayed.

Exmple:

- **Displaying a Context Menu**

1. Using the same project created in the previous section, add the following statements in theMainActivity.java file:

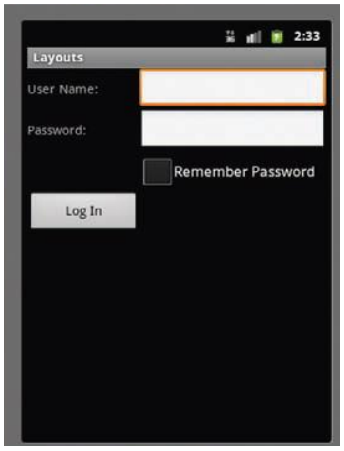
```
import android.view.View;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
public void onCreate(Bundle savedInstanceState) {
        Button btn = (Button) findViewById(R.id.btn1);
        btn.setOnCreateContextMenuListener(this);
}
@Override
public void onCreateContextMenu(ContextMenu menu, View view,
ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, view, menuInfo);
        CreateMenu(menu);
}
@Override
public boolean onContextItemSelected(MenuItem item) {
```

2. Press F11 to debug the application on the Android Emulator.

Lab Assignments

SET A

1. Construct an app that toggles a light bulb on and off when the user clicks on toggle button.
2. Create an Android application to demonstrate the use of CheckBox to select multiple hobbies and display the selected hobbies.
3. Create an Android application to design the following screen using Table Layout.

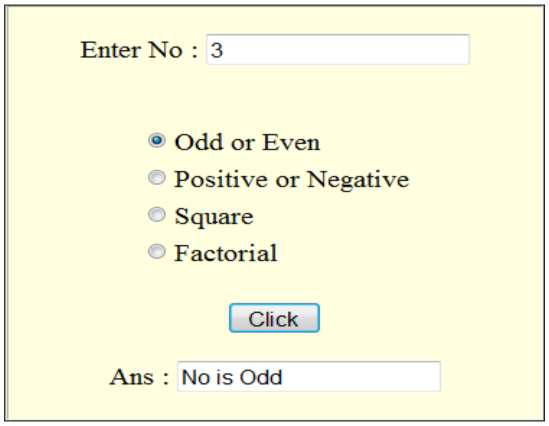


4. Create an Android application to demonstrate the use of AutoCompleteTextView for displaying a list of different colors.
5. Create an Android application to display a list of cities using ListView. Display the selected city using a Toast message when a city is selected.

6. Create an Android application to select a course using Spinner and display the selected course using a Toast message.

SET B

1. Perform following numeric operation according to user selection of radio button

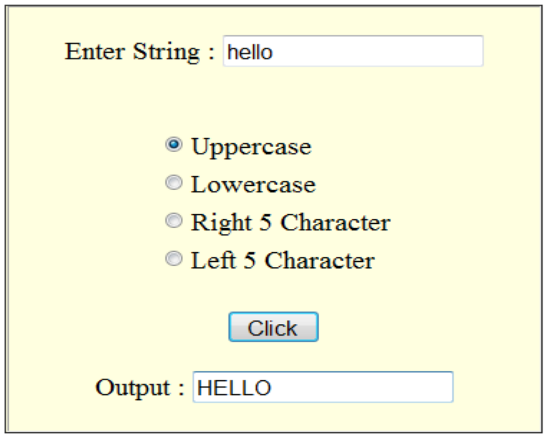


Enter No : 3

- (●) Odd or Even
- ( ) Positive or Negative
- ( ) Square
- ( ) Factorial

Ans : No is Odd

2. Perform following string operation according to user selection of radio button.



Enter String : hello

- (●) Uppercase
- ( ) Lowercase
- ( ) Right 5 Character
- ( ) Left 5 Character

Output : HELLO

3. Create an Android application to demonstrate the use of an Option Menu with items such as New, Open, Save, and Exit. Display a Toast message when any menu item is selected.
4. Create an Android application to demonstrate the use of a Context Menu on a Button. On long-pressing the Button, a Context Menu with options Factorial and Armstrong Number should be displayed, and the selected operation should be performed.

SET C

1. Create application to demonstrate file explorer (Use ListView).

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment No: 5 SQLite Database

Objectives

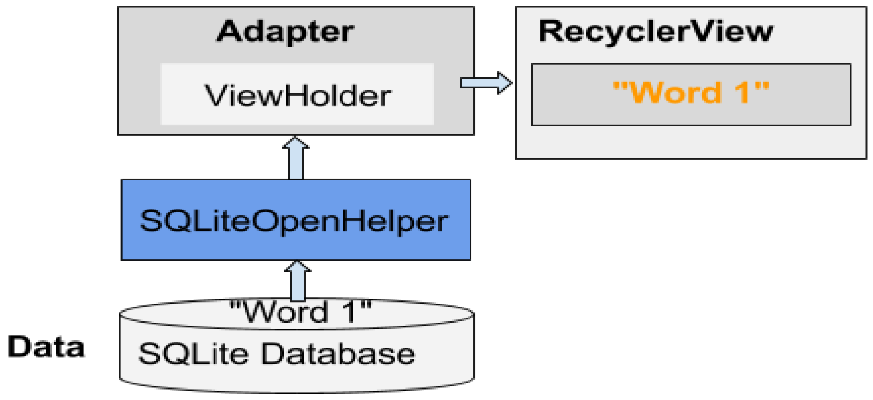
- How to use Multimedia Programming
- How to use database in an application.
- How to create and execute database queries.

Reading

You should read the following topics before starting this exercise:
1. Multimedia Programming
2. SQLite database
3. Classes used for SQLite database

Ready Reference

SQLite is an open-source SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation. SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like ODBC.



- **<u>SQLiteOpenHelper</u>**

Android database sqlite.SQLiteOpenHelper manages database creation, up gradation, down gradation, version management and opening it. We need to create sub class of SQLiteOpenHelper and override onCreate and onUpgrade and optionally onOpen. If database is

not created, onCreate is called where we write script for database creation. If already created, then onOpen is called which opens database.

Constructors	**Description**
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)	Create a helper object to create, open, and/or manage a database
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler)	Create a helper object to create, open, and/or manage a database.

Public Methods	**Description**
void close()	Close any open database object.
String getDatabaseName()	Return the name of the SQLite database being opened, as given to the constructor.
SQLiteDatabase getReadableDatabase()	Create and/or open a database.
SQLiteDatabase getWritableDatabase()	Create and/or open a database that will be used for reading and writing.
void onConfigure(SQLiteDatabase db)	Called when the database connection is being configured, to enable features such as write-ahead logging or foreign key support.
abstract void onCreate(SQLiteDatabase db)	Called when the database is created for the first time.
void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion)	Called when the database needs to be downgraded.
void onOpen(SQLiteDatabase db)	Called when the database has been opened.
void setIdleConnectionTimeout(long idleConnectionTimeoutMs)	Sets the maximum number of milliseconds that SQLite connection is allowed to be idle before it is closed and removed from the pool.
public synchronized void close ()	Closes the database object.

Example:

```
public class DBHelper extends SQLiteOpenHelper{

        public DBHelper(){
                super(context,DATABASE_NAME,null,1);
        }
        public void onCreate(SQLiteDatabase db){
               // Creates new database
               // Create the tables
               // execute query with the help of execSQL();

               // Add initial data
             ………..
          }
          public void onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion){
          }
}
```

- **SQLite Database:** The main package is android.database.sqlite that contains the classesto manage your own databases:

  1. **Database – Creation**

In order to create a database, you just need to call this method openOrCreateDatabase with your database name and mode as a parameter. Its syntax is given below:

```
SQLiteDatabase mydatabase = openOrCreateDatabase("Database name",MODE_PRIVATE,null);
```

Public Methods	**Description**
openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler)	This method only opens the existing database with the appropriate flag mode.
openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)	It is similar to the above method but it does not define any handler to handle the errors of databases
openOrDatabase(String path, SQLiteDatabase.CursorFactory factory)	It not only opens but creates the database if it not exists. This method is equivalent to openDatabase method.

openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)	This method is similar to above method but it takes the File object as a path rather than a string. It is equivalent to file.getPath()

2. Database – Insertion

We can create table or insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below

Example:

```
mydb.execSQL(―CREATE TABLE IF NOT EXISTS Login(Username varchar,Password
Varchar);‖);
mydb.execSQL(―INSERT  INTO  Login  VALUES(‗admin‘,‗admin‘);‖);
```

Another method that also does the same job but take some additional parameter is given below.
execSQL(String sql, Object[] bindArgs): This method not only inserts data, but also used to update or modify already existing data in database using bind arguments.

3. Database – Fetching

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

Example:

```
Cursor resultSet=mydb.rawQuery(―Select * from Login‖,null);
 resultSet.moveToFirst();
String username= resultSet.getString(0);
String password= resultSet.getString(1);
```

Public Methods	**Description**
int getColumnCount()	Returns the total number of columns of the table.
int getColumnIndex(String columnName)	Returns the index number of a column by specifying the name of the column.

String getColumnName(int columnIndex)	Returns the name of the column by specifying the index of the column.
String[ ] getColumnNames()	Returns the array of all the column names of the table.
int getCount()	Returns the total number of rows in the cursor.
int getPosition()	Returns the current position of the cursor in the table.
boolean isClosed()	Returns true if the cursor is closed.

InsertFormat: long insert (String table, String nullColumnHack, ContentValues values)

- First argument is the table name.
- Second argument is a String nullColumnHack
  - Workaround that allows you to insert empty rows
  - Use null
- Third argument must be a ContentValues with values for the row.
- Returns the id of the newly inserted item.

Example:

newId = mWritableDB.insert(TABLE_NAME, null, values);

DeleteFormat:

int delete (String table, String whereClause, String[] whereArgs)

- First argument is table name
- Second argument is WHERE clause
- Third argument are arguments to WHERE clause

Example:

deleted = mWritableDB.delete(WORD_LIST_TABLE, KEY_ID + " =? ", new String[] {String.valueOf(id) });

UpdateFormat:

int update(String table, ContentValues values, String whereClause, String[] whereArgs)

- First argument is table name
- Second argument must be ContentValues with new values for the row
- Third argument is WHERE clause
- Fourth argument are the arguments to the WHERE clause

Example:

```
ContentValues  values = new ContentValues();

values.put(KEY_WORD, word);


mNumberOfRowsUpdated  =  mWritableDB.update(  WORD_LIST_TABLE,  values,  // new values toinsert

KEY_ID + " = ?", new String[]{String.valueOf(id)});
```

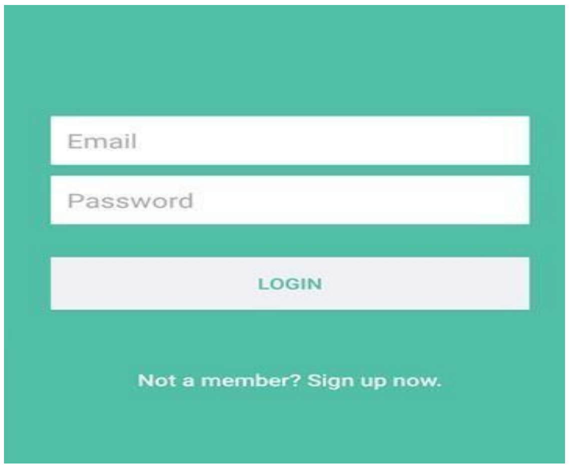
Lab Assignments

SET A

1. Create table Customer (id, name, address, phno). Create Application for Performing the following operation on the table. (Using sqlite database)
   i) Insert New Customer Details.
   ii) Show All the Customer Details
2. Create Following Table:
   Emp (emp_no,emp_name,address,phone,salary) Dept (dept_no,dept_name,location)
   Emp-Dept is related with one-many relationships.
   Create application for performing the following Operation on the table
   Add Records into Emp and Dept table.
   Accept Department name from User and delete employee information which belongs to that department.

SET B

1. Create sample application with login module (Check username and password). On successful login, pass username to next screen and on failing login, alert user using Toast (Hint: Use Login (username, password) Table.)



.

SET C

1. Create table Student (Sid, Sname). Create Application for Performing thefollowing operation on the table.
   i) Insert New Student Details.
   ii) Show All the Students Details.
   iii) Update student name
   iv) Delete Student.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]