CHAPTER-11:

CREATING ANDROID ACTIVITIES

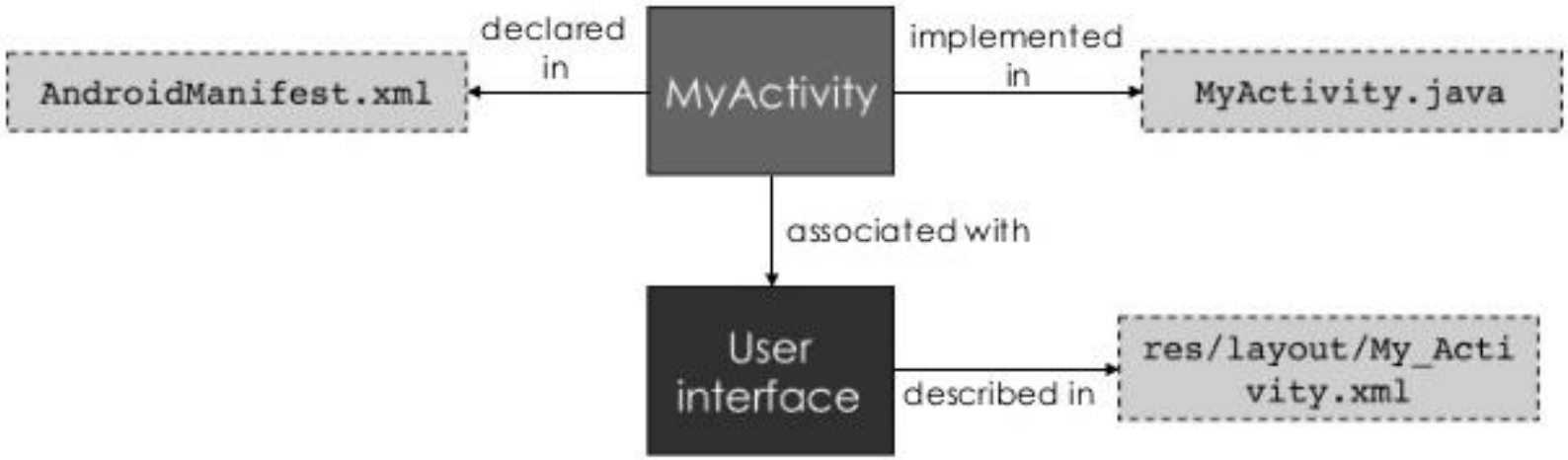
BBC 3, Business Telecommunications

Academic Year, 2023/ 2024, Semester – I

By Ddamba Abdul
0703348992, addamba@mubs.ac.ug
@MUBS

Introduction

- **Every screen in an Android application is an Activity.**
  - An activity is a single, focused thing that the user can do
  - Example: visualize the Read emails activity
  - Activity will always have a User Interface



What is a Activity?

An Activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map.

Each activity is given a window in which to draw its user interface. The window typically fills the screen, but may be smaller than the screen and float on top of other windows

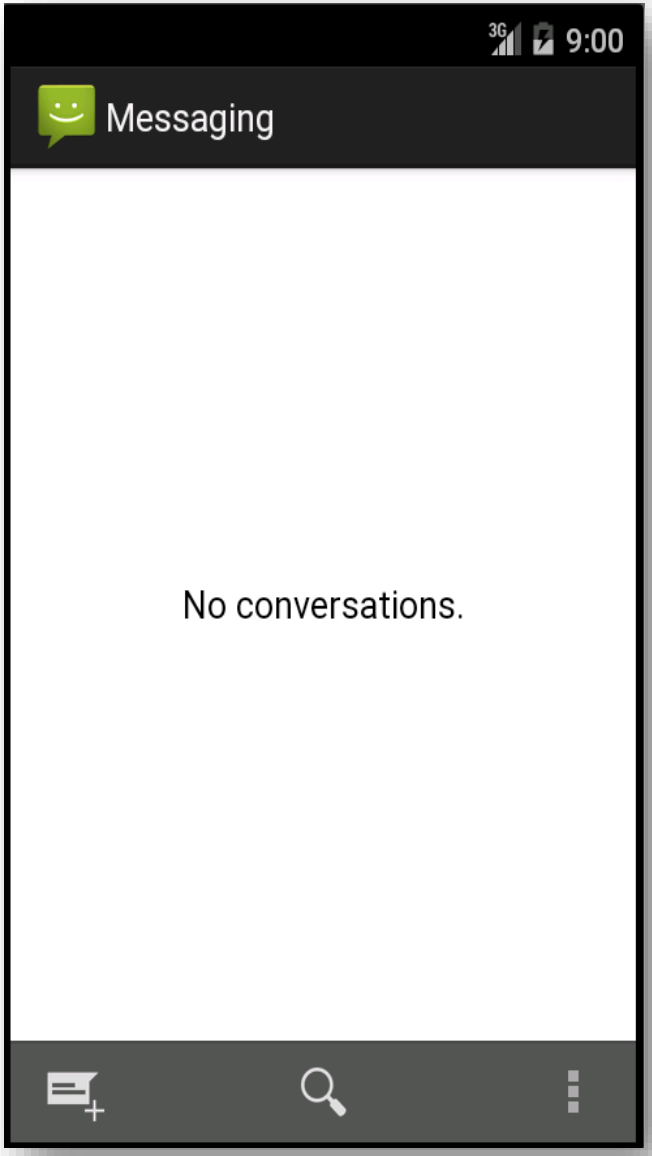
An application usually consists of multiple activities that are loosely bound to each other. The "first" activity gets opened when the application launches

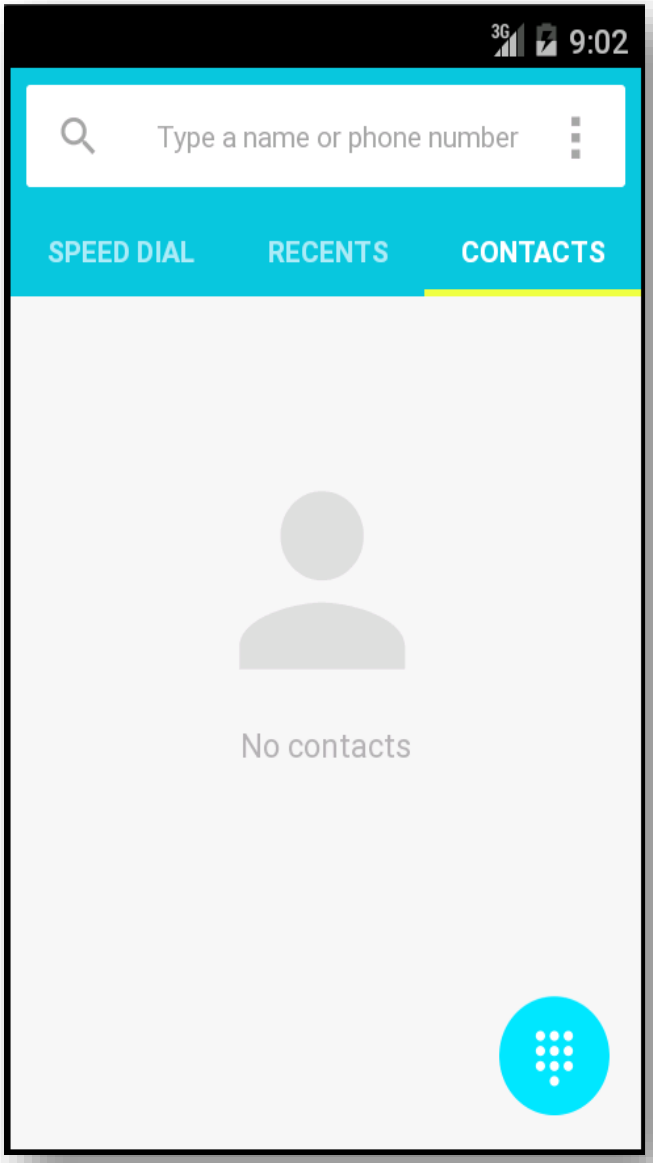
What happens when an Activity is Launched?

Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack .

When a new activity starts, it is pushed onto the back stack and takes user focus.

Android Activity





Apps, Memory & Storage

❑ Storage

- Your device has apps and files installed and stored on its internal disk, SD card, etc.

❑ Memory

- Some subset of apps might be currently loaded into the device's RAM and are either running or ready to be run
  - If the user loads an app, it is loaded from storage into memory
  - When the user exits an app, it might be cleared from memory, or might remain in memory so you can go back to it later
  - See which apps are in memory
    - *Settings >> Apps >> Running*

Activity States

- **In Android, an activity goes through different states during its lifecycle**

  - The various activity states are:

    - *Active or Running*
    - *Paused*
    - *Stopped/ Backgrounded*
    - *Restarted/ Resumed*
    - *Destroyed*

Activity States

1. Active or Running State

- When users touch an app's icon, the app is started & becomes visible (foreground) to the users
  - In this state, the app is done loading & now visible on the screen
  - The app is at the top of the activity stack and the users can interact with it and start using it
  - This is the highest priority activity in the Android Activity stack,
    - Therefore, it will only be killed by the OS in extreme situations, e.g. if the activity tries to use more memory than is available on the device as this could cause the UI to become unresponsive

Activity States

2. Paused

- The app enters a paused state, when a user is interrupted while using the app
  - The interruption is caused by;
    - A pop-up window or a notification window showing up,
    - A user being distracted and not touch the screen for a period of time and the device will go to sleep
  - The app will fade but still be partially visible or partially visible
  - Paused activities are still alive, i.e. that is, they maintain all state and member information
  - It is considered to be the second highest priority activity in the activity stack, and the OS will kill to free resources for another

Activity States

3. Stopped/ Backgrounded

- In this state, the screen goes black, or activity becomes obscured by another. This happens when;
  - A user don't touch the screen for a longer period
  - A user starts another app and the original app becomes invisible
- Stopped activities still try to retain their state and member information for as long as possible
- These activities are considered to be the lowest priority of the three states
  - Therefore, the OS will kill activities in this state first to satisfy the resource requirements of higher priority activities

Activity States

4. Restarted/ Resumed

- It happens when the app that was previously in a paused or stopped state becomes fully active
  - This happens when a user;
    - Turns its screen on or presses a back button to navigate back to the activity
    - Closes a temporary pop-up screen
- In this state, the app is restored to its previously saved state, and then displayed to the user.

Activity States

5. Destroyed

- An app can remain in the Stopped state for quite some time, however it only gets destroyed by the OS when;
  - A device is rebooted
  - A user runs a number of other apps before coming back to the original app
- Destroying the app is needed to free up resources for other apps that the user is actually interacting with.

Activity Stack

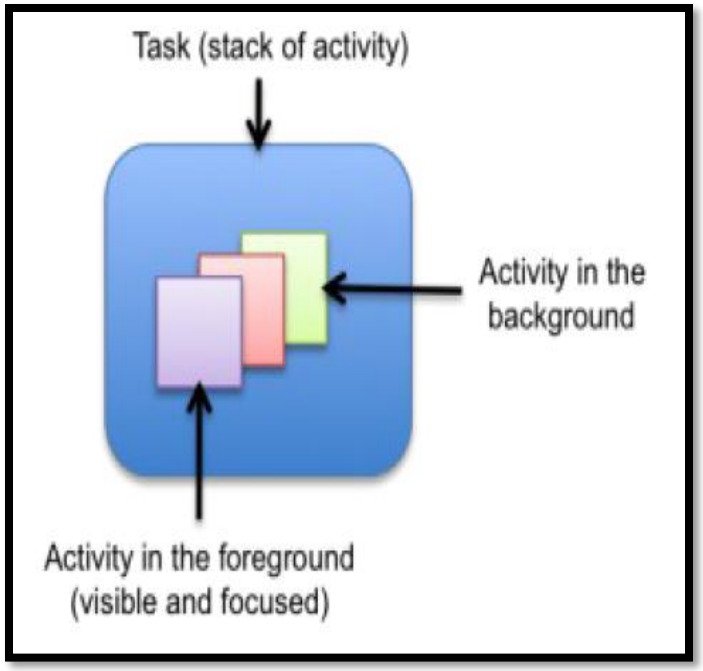
- **Activities in the system are managed as an *activity stack*. An activity has essentially four states:**

If an activity in the foreground of the screen (at the top of the stack), it is *active* or *running*.

If an activity is completely obscured by another activity, it is *stopped*.

If an activity has lost focus, it is *paused*.

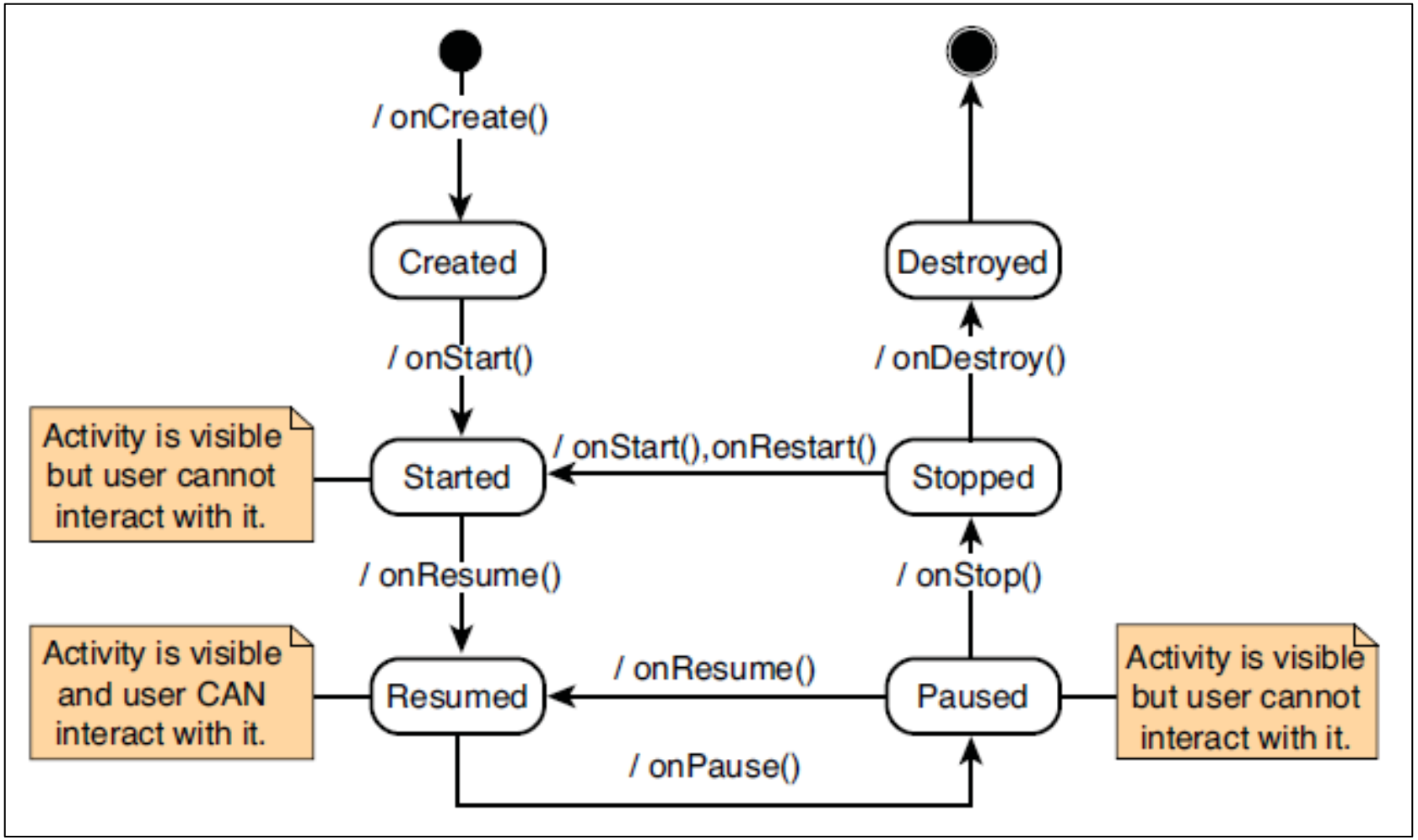
If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process.



Activity Lifecycle

- **Includes a sequence of methods & events that occur as an activity transitions through its various states**
  - The key lifecycle methods include:
    - `onCreate`
    - `onStart()`
    - `oResume()`
    - `onPause()`
    - `onStop()`
    - `onRestart`
    - `onDestroy()`
  - Each method serves a specific purpose for a programmer
    - They allowing programmers to manage and control the behavior of an activity at different points in its lifecycle

Illustrating the Activity Lifecycle



Activity Lifecycle Methods

❑ onCreate() method

- Android OS calls it when the activity is first created
- A programmer uses it for;
  - One-time initialization, such as;
    - Setting up user interface components,
    - Initializing variables,
    - Layout inflation for the activity (converting xml into java objects)

```
public class FooActivity extends Activity {
    ...
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);    // always call super
        setContentView(R.layout.activity_foo); // set up layout
        any other initialization code;    // anything else you need
    }
}
```

  - After this, the Activity object exists think of this as the "constructor" of the activity

Activity Lifecycle Methods

❑ `onStart()` method

- Called when the activity becomes visible but not yet active
  - Programmers can use this method to;
    - Perform any actions that should happen when the activity becomes visible, e.g. registering broadcast receivers or preparing the UI for user interaction

Activity Lifecycle Methods

❑ `onResume()` method

- This method is called when the activity is in the active state and actively interacting with the user

  - Programmers use it to;

    - Start animations,
    - Acquire resources that should be exclusive to the activity, and
    - Generally set up anything that requires user interaction.

Activity Lifecycle Methods

❑ `onPause()` method

- Called when the activity is partially obscured or another activity is in the foreground

  - Programmers use this method to:

    – Release resources that are no longer needed when the activity is not in the foreground, such as stopping animations or saving user data

Activity Lifecycle Methods

❑ `onStop()` method

- This method is called when the activity is no longer visible but still in memory
  - The app might be running but the activity is not
  - It happens when a user
    - Chose another app,
    - Starts a different activity in the app, or
    - Receives a phone call while in the app
- It's a good place to
  - Release resources that are no longer needed,
  - Stop background services, or
  - Save persistent data

Activity Lifecycle Methods

`onRestart()` method

- It is called when activity was stopped but is started again later (all but the first start)
- It is not as commonly used; favor `onResume()`
- Re-open any resources that `onStop()` closed

Activity Lifecycle Methods

❑ `onDestroy()` method

- Called when the activity is being terminated or removed from memory
  - Programmers use this method to perform cleanup actions, such as:
    - Releasing all resources,
    - Unregistering broadcast receivers, and
    - Finalizing any remaining tasks.

When are these Methods Invoked?

- **When open a new app**
  - onCreate() --> onStart() --> onResume()
- **When back button pressed and exit the app**
  - onPaused() -- > onStop() --> onDestory()
- **When home button pressed**
  - onPaused() --> onStop()
- **After pressed home button when again open app from recent task list or clicked on icon**
  - onRestart() --> onStart() --> onResume()

When are these Methods Invoked?

- **When open app another app from notification bar or open settings**
  - onPaused() --> onStop()
- **Back button pressed from another app or settings then used can see our app**
  - onRestart() --> onStart() --> onResume()
- **When any dialog open on screen**
  - onPause()
- **After dismiss the dialog or back button from dialog**
  - onResume()

When are these Methods Invoked?

- **Any phone is ringing and user in the app**
  - onPause() --> onResume()
- **When user pressed phone's answer button**
  - onPause()
- **After call end**
  - onResume()
- **When phone screen off**
  - onPaused() --> onStop()
- **When screen is turned back on**
  - onRestart() --> onStart() --> onResume()

Creating Android Activities

- **In Android, Activities**
  - Are Java classes containing Java codes
  - Activities supports the user-interface screen
  - Therefore, to create an Activity, you create a Java class that extends the Activity base class

Sample Basic Java Activity File

```
package com.ddamba.displaycapturedtext;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

    }
}
```


Components of Basic Activity File

- **A basic Activity file in Android typically consists of several components**
  - These work together to define the behavior & appearance of the activity and include the following;
    - *Package Declaration*
    - *Import Statements*
    - *Class Declaration*
    - *onCreate() Method*
    - *Additional Methods*
    - *Layout Inflation*
    - *User Interface (UI) Components*
    - *Event Handlers*

Components of Basic Activity File

❑ *Package Declaration*

- This is the first line in the file and specifies the package in which the activity is located

  - Whenever you create a new project in Android, the IDE will create a package to contain your activity classes

  - The IDE automatically created a package name using the given app name & domain name in a reverse order as shown below;

```
package com.ddamba.myApp;
```


Components of Basic Activity File

❑ *Import Statements*

- These statements include necessary classes & libraries to be used in the activity

  - There are two import statements in the MainActivity.java file to support class functions in our application:

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
```


Components of Basic Activity File

❑ *Import Statements*

```
import androidx.appcompat.app.AppCompatActivity;
```

- The above import statement includes the **`AppCompatActivity`** class & its functionalities in your Java class
- What is the **`AppCompatActivity`** class?
  - It is a base class part of the **`AndroidX`** library from which user-defined activity classes are created to provide a basis for building interactive user interfaces and handling user interactions
  - It offers compatibility with modern Android features and user interface components, even on older Android devices

Components of Basic Activity File

❑ *Import Statements*

```
import androidx.appcompat.app.AppCompatActivity;
```

- **import** - A keyword is used to bring classes or packages from external libraries or modules into your current Java class.
- **androidx** - A package that contains extended Android libraries, an improvement to the previous original Android Support library provided by Google
- **appCompat** - A package that contains support libraries which can be used to make the apps developed with newer versions work with older versions.
- **app** - A package contains classes that are needed for the creation of Android apps, e.g., the; **AppCompatActivity** class that allows us to define UI's.
- **AppCompatActivity** is ultimately the imported super class for any activity that we create, e.g., **MainActivity**

Components of Basic Activity File

❑ *Import Statements*

```
import android.os.Bundle;
```

is used to bring the **Bundle** class from the Android OS library into your current activity class

- **import** – A keyword used to include classes or packages from external libraries into your current Java class.
- **android.os** – A package containing classes related to the Android operating system and core functionalities.
- **Bundle** – It's a class located under the **os** package

The Bundle class is used for;

- Storing and passing a set of data (in a key pair value) between Android activities
- Stores the current state (data) of the activity and passes data back to the activity itself when it re-creates the activity in the new orientation

Components of Basic Activity File

❑ *Cass Declarations*

- Create an `Activity` class extending the `AppCompatActivity` SDK-provided base class
  - This class is given a name that follows Java naming conventions
  - To use the base class, you must import it in your current activity file, as previously seen

```
public class MainActivity extends AppCompatActivity {

    // All the Activity Functionali is added in here

}
```


Components of Basic Activity File

❑ *onCreate() Method Declaration*

- This is a lifecycle method invoked by the Android OS when the activity is being created

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

}
```

Components of Basic Activity File

❑ *onCreate() Method Declaration*

- **`@Override`**
  - A keyword that is automatically put before each of the inherited methods declaration inside an activity class
  - This keyword tells the Android compiler how you can modify the `onCreate()` method of the `AppCompatActivity` in order to provide your own implementation of that method
- **`onCreate()`**
  - This is an `AppCompatActivity` class method
  - It is inherited & overridden by the current `MainActivity` class
  - It's the first method that is executed by the activity when the app starts to run, i.e. this method drives your app to run

Components of Basic Activity File

❑ *onCreate() Method Declaration*

- **What is the `onCreate()` method used for?**
  - It is under this method where you place all the application start-up code logic needed to:
    - Setup the user interface
    - Create views, e.g. buttons, TextViews, etc.
    - Initialize class-scope variables
    - Create other methods that defines custom-app logic
    - Inflate (connect) the UI `xml` elements with the Activity Java code
    - Bind data to lists, e.g. connecting UI views such as `ListView` to a data source
    - Initialize background threads

Components of Basic Activity File

onCreate() Method Declaration

- **The `(Bundle savedInstanceState)` Parameter of the `onCreate()` Method**
  - The data type of this parameter is a **`Bundle`**
  - The name of this parameter is **`savedInstanceState`**
    - This object stores information on the state of the Activity that will be restored, in case it was destroyed in an orientation change
    - In case, there was no data available in the activity, then the **`savedInstanceState`** will be null

Components of Basic Activity File

❑ The Contents of the `onCreate()` Method

- **`super.onCreate(savedInstanceState)`**

  - The above statement calls the **`onCreate()`** method of the super class (**`AppCompatActivity`**) to create and initialize your current user-defined activity

    - It is passed the **`savedInstanceState`** bundle

Components of Basic Activity File

❑ The Contents of the `onCreate()` Method

- `setContentView(R.layout.activity_main);`
  - This is a method that **sets**, **loads** & **displays** the user interface layout on to the screen for an activity defined in the XML file (in this case, `activity_main.xml`)
  - It sets up what the user sees & interacts with when they open the activity
    - This process is called "`inflating`" the layout, and it means that the XML layout is converted into a view hierarchy that can be displayed on the device's screen

Components of Basic Activity File

❑ A breakdown of the setContentView Method

- '`setContentView`' – An SDK-provided library method by the Android '`Activity`' class
- '`(`' & '`)`' - These parentheses enclose method parameters
- '`R.layout.activity_main`' - The argument passed to the `setContentView` method
- '`R`'- It's a Java class automatically generated by the Android build system, it contains resource IDs for all the resources used in the app, e.g., as layouts, strings, and images
- '`.`' (do) - Used to access the members (resources) of the `R` class
- `Layout` – An XML resource type representing the XML layout file in the "`res/layout`" directory
- `activity_main` –This is the name of the XML layout file you want to set as the content view for the current activity

-THE END-

