Sr. Software Engineer (SSE)

Abstract
This is one of the subject from my personal notes series named “Coding-With-Arqam” that I am developing from the start of my professional development career.

Subject
Android

ANDROID

--> General Points:

-> Android is an open source and Linux-based operating system.

-> Led by Google.

-> Developed in the Java language using the Android Software Development Kit.

--> Components & Description:

-> Activities:

-> They dictate the UI and handle the user interaction to the smart phone screen.

-> They are screens like login, register, etc.

-> An activity represents a single screen with a user interface,in-short Activity performs actions on the screen.

-> An activity is implemented as a subclass of Activity class as follows:

-> public class MainActivity extends Activity {}

-> Callback:

-> onCreate: This is the first callback and called when the activity is first created.

-> onStart: This callback is called when the activity becomes visible to the user.

-> onResume: This is called when the user starts interacting with the application.

-> onPause: This is called when the current activity is being paused and the previous activity is being resumed

-> onStop: This callback is called when the activity is no longer visible.

-> onDestroy: This callback is called before the activity is destroyed by the system.

-> onRestart: This callback is called when the activity restarts after stopping it.

-> Services:

-> They handle background processing associated with an application to perform long-running operations.

-> A service is implemented as a subclass of Service class as follows:

-> public class MyService extends Service {}

-> Broadcast Receivers:

-> They handle communication between Android OS and applications.

-> Content Providers:

-> They handle data and database management issues.

-> Fragments:

-> Represents a portion of user interface in an Activity.

-> Views:

-> UI elements that are drawn on-screen including buttons, lists forms etc.

-> Layouts:

-> View hierarchies that control screen format and appearance of the views.

-> Intents:

-> Messages wiring components together.

-> Allows users to coordinate the functions of different activities.

-> Resources:

-> External elements, such as strings, constants and draw able pictures.

-> Manifest:

-> Configuration file for the application.

-> The View objects are usually called "widgets" and can be one of many subclasses, such as Button or TextView.

Muhammad Arqam

-> The ViewGroup objects are usually called "layouts" can be one of many types that provide a different layout structure, such as LinearLayout or ConstraintLayout .

--> Build a UI with Layout Editor:

-> The Layout Editor appears when you open an XML layout file.

-> It makes to add things (like btn) easily.

-> Components:

-> Palette, Component Tree, Toolbar, Design editor, Attributes, View mode, Zoom and pan controls.

--> UI Layouts:

-> View is the base class for widgets, which are used to create interactive UI components like buttons, text fields, etc.

-> The ViewGroup is a subclass of View and provides invisible container that hold other Views or other ViewGroups and define their layout properties.

-> Types:

-> LinearLayout (Horizontal + Vertical):

-> LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.

-> You can specify the layout direction with the android:orientation attribute.

-> RelativeLayout:

-> View group that displays child views in relative positions to sibling elements.

-> Attributes:

-> android:id

-> This is the ID which uniquely identifies the layout.

-> android:gravity:

-> This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.

-> android:ignoreGravity

-> This indicates what view should not be affected by gravity.

-> PercentFrameLayout and PercentRelativeLayout.

-> GridLayout

-> CoordinatorLayout

-> Layout Weight:

-> LinearLayout also supports assigning a weight to individual children with the android:layout_weight attribute.

-> This attribute assigns an "importance" value to a view in terms of how much space it should occupy on the screen.

-> Default weight is zero.

-> Equal distribution:

-> Set android:layout_height & android:layout_width as "0dp".

-> Unequal distribution:

-> If there are three text fields and two of them declare a weight of 1, while the other is given no weight, the third text field without weight doesn't grow. Instead, this third text field occupies only the area required by its content. The other two text fields, on the other hand, expand equally to fill the space remaining after all three fields are measured.

--> UI Controls:

-> TextView

-> EditText

-> AutoCompleteTextView

-> Button

-> ImageButton

-> CheckBox

Muhammad Arqam

-> ToggleButton

-> RadioButton

-> RadioGroup

-> ProgressBar

-> Spinner

-> TimePicker

-> DatePicker

--> Event Handling:

-> Events are a useful way to collect data about a user's interaction with interactive components of Applications.

-> The Android framework maintains an event queue as first-in, first-out (FIFO) basis.

-> 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.

-> Types:

-> onClick, onLongClick, onFocusChange, onKey, onTouch, onMenuItemClick onCreateContextMenu

--> Drawables:

-> Image files like .png, .jpg, .gif or XML files that are compiled into bitmaps, state lists, shapes, animation drawable. They are saved in res/drawable/ and accessed from the R.drawable class.

--> Accessing Resources in Code:

-> When your Android application is compiled, a R class gets generated, which contains resource IDs for all the resources available in your res/ directory. You can use R class to access that resource using sub-directory and resource name or directly resource ID.

-> Example:

-> To access res/drawable/myimage.png and set an ImageView you will use following code −

-> ImageView imageView = (ImageView) findViewById(R.id.myimageview);

// Above Line: make use of R.id.myimageview to get ImageView defined with id myimageview in a Layout file

imageView.setImageResource(R.drawable.myimage);

// Above Line: makes use of R.drawable.myimage to get an image with name myimage available in drawable sub-directory under /res.

--> Android Advanced Concepts:

-> The Drag/Drop Process

-> Notifications

-> Location Based Services

-> Sending Email

-> Sending SMS

-> Phone Calls

Muhammad Arqam

--> Organize resource in Android Studio:

-> MyProject/

app/

manifest/

AndroidManifest.xml

java/

MyActivity.java

res/

drawable/

icon.png

layout/

activity_main.xml

info.xml

values/

strings.xml

--> Topics covered in android:

-> XML layouts

-> image manipulation using bitmaps

-> file system of Android

-> datastructers of Android like Lists, HashMaps

-> photoviewer layouts

Muhammad Arqam