Mobile App Development (Mids) - Jaish Khan

Key Terms

1. **IDE** (Integrated Development Environment) → A software that provides the environment for writing, testing and debugging code.
2. **Android** → The open-source mobile operating system developed by Google.

- To build apps for android we use the **Android Studio IDE** by Google.
- Android Apps are bundled as APKs (Android Packages).
- APKs contain all the necessary components: code, resources and manifest file.

3. **XML** → The markup language used to define structure/layout of an Android App.
4. **Java** → An object-oriented programming language used to define behaviour of an Android App.
5. **Compiletime Errors** → An error that occurs during the compilation of an Android application. These are detected by the compiler before the application execution.
6. **Runtime Errors** → An error that occurs during the execution of an Android application. These are detected by the runtime environment while the app is running.
7. **Instance Variables** → Variables that are declared within a class and outside a function.
8. **Callback Function** → A function that can be passed as an argument to another function.
9. **Debugging** → Process of identifying and resolving issues/bugs.
10. **Production Mode** → The state of an Android application when it is deployed and released to end-users.
11. **Development Mode** → The state during the app's development and testing phase. Debugging and Logging features are used.
12. **Exceptions** → Unexpected events or errors that occur during execution.
13. **Try-Catch** → Keywords used to handle exceptions. The code inside of the `try` block is executed and if any exceptions are generated then the code inside of the `catch` block is executed.

> **How to solve or see runtime errors?**
>
> Developers can use logcat and debugger in Android Studio.

Stages of Mobile App Development

1. Planning
2. Design
3. Development
4. Testing
5. Deployment
6. Maintenance

Basics

There are two main files in every android studio project: `MainActivity.java` and `activity_main.xml`.

- `MainActivity.java` resides in the app\src\java\com\example\projectname folder.
- `activity_main.xml` resides in the app\src\res\layout folder.

App Preview

There are two ways of previewing an app that we have built in Android Studio:

1. **AVD** (Android Virtual Device) → Emulates an android device, allowing developers to run and test their app inside of Android Studio.
2. **ADB** (Android Debug Bridge) → Enables installing the app onto your own Android device to be able to test them. (requires enabling of Developer Options → USB Debugging).

> AVD is advantageous compared to ADB because it allows us to test for multiple different devices and screen sizes (on the same PC).

Logs

Logging allows developers to record information about the application's execution including errors and warnings.

Logcat → A tool inside of Android Studio that displays the system's as well as the application's log messages. It is a valuable tool for debugging.

Android Attributes

1. `id` → A unique name given to an element, to be able to reference it (in the java file).
    1. The `findViewById(R.id.name)` method is used to specify the id in Java.
2. `text` → Visible text inside of **TextView**, **Button** etc. elements.
    1. `textSize` → For font size (preferably in `sp`).

    2. `textColor` → For font color (using a Hex value or Color resource).
3. `background` → Sets the background color or drawable.
4. `hint` → Placeholder text inside **EditText** elements.
5. `inputType` → Specifies the type of the input (like textPassword, number etc).
6. `src` → To define the source image of an ImageView.
7. `visibility` → Can be visible, invisible or gone.

Sizing

`layout_width` and `layout_height` are used to define the width and height of the element itself. They can have these values:

1. A specific size (preferably in `dp`) like 480dp, 700dp etc.
2. `match_parent` → This element now takes up all the available space in its parent.
3. `wrap_content` → This element only takes up as much space, as is required by its content.

Units

Android Studio has these two unique units:

1. `sp` → scale-independent pixel: which is preferred for fonts.
2. `dp` → density-independent pixel: which is used for everything else (margins/paddings).

Assets

Android allows us to have many different types of assets (static resources) in our app.

1. **Cliparts** → Premade vector images provided by Android.
2. **XML Shapes** → We can also draw our own vector images using XML.
3. **Gradients** → Smooth color transitions which can be defined as XML.
4. **Images** → Can be `.png`, `.jpg` or `.webp`.
5. **HTML Files** → To view them using WebView.

Widgets

They are reusable UI components/elements (also called **Views**) used to build the user interface of the app. They have equivalent classes, **with the same name**, in Java which allows them to be imported into the Java file.

Widget	Used for
TextView	Read-only text

Widget	Used for
EditText	Editable text field
Button	Clickable UI field
WebView	Displaying images
ImageView	Displaying web content

A `Toast` element, which is not a widget for building the UI, is used for displaying temporary and breif messages on the screen.

Important Points

1. When using `WebView`
    1. We have to enable the internet permission in the `AndroidManifest.xml` file.
    2. We have to enable javascript using `setJavaScriptEnabled(True)` method.

Click Event Handling

There are two ways of handling click events in Android Java (usually on Buttons).

1. Using the `android:onClick="function"` attribute to specify a function in XML.
2. Using `setOnClickEventListener(view → function)` method to attach an event listener in Java.

You should never use the *first way* due to many reasons. Almost always use the **second way**.

Attaching Resources

Different widgets like `ImageView` and `WebView` require some resources to be specified. We can do that by placing our resources in the **drawable** folder (inside the *res* folder).

> We can then use methods like `setImageResource(R.drawable.image)` and `loadUrl("https://www.example.com")` to specify the resource for our ImageView and WebView elements, respectively.

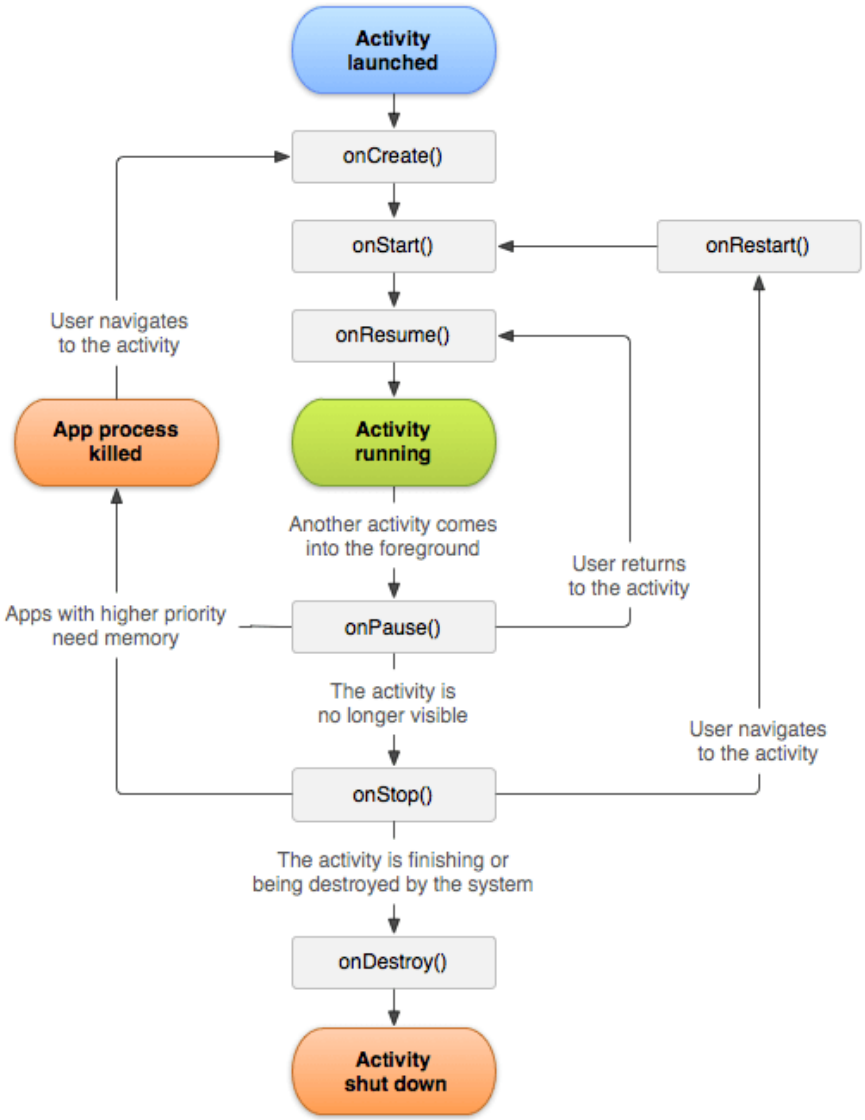
Activities

An **activity** is a "screen" for the user. Each activity has a xml layout file and a java file. We can also create new activities in Android and every activity we create generates two more files for it: one for Java and one for XML.

Activity Life Cycle

Every activity has these 7 functions which make up the "Activity Life Cycle".

Method	Runs When
onCreate	activity is first created
onStart	activity is becoming visible to the user
onResume	activity will start interacting with the user
onPause	activity is not visible to the user.
onStop	activity is no longer visible to the user.
onRestart	after your activity is stopped, prior to start.
onDestroy	before the activity is destroyed.



Intents

A fundamental concept of Android. They are basically like *messages* and are used for **navigation** and **message-passing**.

> The general syntax is to create an object of the `Intent` class and give the source activity and destination activity name to it. Then use the `startActivity()` method to execute the intent.

There are two main types of intents: Explicit and Implicit.

Explicit Intent

Where it is specified which component of which application will answer the intent. They are used for in-app actions like navigating to another acitivity/screen of the same app or starting a service to download a file in the background.

Implicit Intent

Where the component isn't specified and instead a general action is declared, allowing other apps to handle the intent. They are used for navigating to other apps or retrieving information from other apps.

Extras

They are key-value pairs which can be attached using the various `putExtra()` methods and carry additional information to be passed on to another activity or app. We can also create a `Bundle` object to send all the extra data at once.

Actions

Android uses actions to specify the type of task to be performed upon an intent.

1. `ACTION_SEND` → Sending information generally like Text, Images and Files.
2. `ACTION_SENDTO` → Sending information specifically like Emails and Phone Numbers.
3. `ACTION_VIEW` → Used to show something; like Webpages and Images, to the user.

ViewGroups

An invisible container that defines the layout structure of the Activity and can hold multiple Views. They are also called *Layouts* (they're not the same). **Layouts** are a subclass of ViewGroups can define how children are arranged on the screen and the structure of the UI.

In Android there are many different ViewGroups that can be used inside a Layout file (activity_main.xml):

1. **LinearLayout** → Arranges children in a single row (horizontal) or column (vertical).
2. **RelativeLayout** → Positions children relative to the each other or parent container.
3. **ConstraintLayout** → *The most flexible viewgroup*. Allows placing children relative to each other or parent using constraints.
4. **TableLayout** → Organizes children in rows and columns using `TableRows`.
5. **GridView** → Places children in a grid-like structure of rows and columns.
6. **FrameLayout** → Displays a single view or multiple views layered on top of each other.
7. **ScrollView** → A special type of FrameLayout as it only accepts a single view. We usually use a LinearLayout for this and it then allows scrolling through the children of the LinearLayout.
8. **HorizontalScrollView** → Same as ScrollView but it allows scrolling horizontally.

Aligning

We can use these attributes in XML to align elements.

1. `gravity` → To align the content inside this element.
2. `layout_gravity` → To align this element inside its parent.
3. `padding` → To define the space between this element's boundary and content.
4. `layout_margin` → To define the space outside this element's boundary.
5. `scaleType` → Used to scale images to the bounds of an ImageView.