**Chapter 2 - Android Application Design Essentials**

**1. Anatomy of an Android Application**

Android applications are structured into several components that interact to form a complete app. The basic structure of an Android application includes:

* **Activities**: These are components that handle user interface (UI) interaction. Each screen in an app is typically represented by an Activity.
* **Services**: Background components that perform tasks without a UI, such as playing music or downloading files.
* **Broadcast Receivers**: These listen for system-wide broadcast announcements, such as a change in network status or battery level.
* **Content Providers**: Components used to manage shared data between different apps, such as contacts or media.

Each Android app consists of the following major files:

* **Manifest File**: Describes the app's structure, permissions, and components.
* **Java/Kotlin Source Files**: Contain the app's logic and behavior.
* **Resources**: Layouts, strings, images, and other non-code assets.

**2. Android Terminologies**

Key terminologies in Android development:

* **Context**: A handle to the system, providing access to resources, databases, and system services. It is often used to get access to app-specific data.
* **Activity**: A component that provides the UI and handles user interactions. There can be multiple Activities in an app, each representing a different screen.
* **Service**: A component that runs in the background to perform tasks that don't require direct user interaction (e.g., playing music, downloading data).
* **Intent**: A message object used to request an action or communicate between components (e.g., starting an activity or service).
* **Broadcast Receiver**: A component that listens for and reacts to broadcast messages from other applications or the system itself.
* **Content Provider**: Manages access to a shared set of data, enabling data sharing between apps.

**3. Application Context**

The **Context** in Android is the interface to global information about an application environment. The Context is often used in Android to access resources, databases, shared preferences, and system services. It can be used to:

* Access system services (e.g., location manager, connectivity manager).
* Access application resources (e.g., strings, images, layouts).
* Launch Activities, Services, or Broadcast Receivers.

**Types of Context:**

* **Application Context**: Associated with the entire app and can be used anywhere within the app.
* **Activity Context**: Tied to a specific Activity and should generally be used within that Activity.

**4. Activities**

An Activity in Android represents a single screen with a user interface. When an app is launched, the **MainActivity** is typically opened, and from there, the app can navigate to other Activities.

**Lifecycle of an Activity:**

* **onCreate()**: Called when the activity is first created. This is where the initial setup and UI binding occurs.
* **onStart()**: Called when the activity becomes visible.
* **onResume()**: Called when the activity begins interacting with the user.
* **onPause()**: Called when the activity is no longer in the foreground, typically to save data.
* **onStop()**: Called when the activity is no longer visible.
* **onDestroy()**: Called when the activity is about to be destroyed, such as when it’s finished or when the system needs to reclaim resources.

**5. Services**

A **Service** is a component that runs in the background to perform long-running operations. Unlike Activities, Services do not have a UI and can continue running even if the user navigates away from the app.

Types of Services:

* **Started Service**: Initiated by calling startService(), typically runs until explicitly stopped by the app.
* **Bound Service**: Runs as long as another component is bound to it. It is typically used to perform tasks for multiple components.

Lifecycle of a Service:

* **onCreate()**: Called when the service is first created.
* **onStartCommand()**: Called when a component starts the service.
* **onBind()**: Called when a component binds to the service.
* **onUnbind()**: Called when all clients unbind from the service.
* **onDestroy()**: Called when the service is destroyed.

**6. Intents**

An **Intent** is a messaging object used to request an action. Intents are used for:

* **Starting Activities**: For example, opening a new screen or displaying content.
* **Starting Services**: To perform background operations.
* **Broadcasting Messages**: For communicating between components or apps.

**Types of Intents:**

* **Explicit Intents**: Specify the component (Activity, Service, etc.) to be launched. For example, Intent intent = new Intent(this, AnotherActivity.class);
* **Implicit Intents**: Do not specify the component explicitly but specify the action to be performed. For example, opening a website using an implicit Intent.

**7. Receiving and Broadcasting Intents**

**Broadcasting Intents**: Android allows apps to broadcast messages that can be received by other apps or system components. Common broadcasts include system events such as battery low or Wi-Fi state changes.

java

Copy code

Intent intent = new Intent("com.example.broadcast.MY\_NOTIFICATION");

intent.putExtra("message", "Hello World!");

sendBroadcast(intent);

**Receiving Intents**: Apps can listen for broadcasts by registering **BroadcastReceivers**. These components listen for specific Intent filters to perform actions when a broadcast is received.

java

Copy code

IntentFilter filter = new IntentFilter("com.example.broadcast.MY\_NOTIFICATION");

BroadcastReceiver receiver = new MyReceiver();

registerReceiver(receiver, filter);

**8. Android Manifest File and Its Common Settings**

The **AndroidManifest.xml** file is a crucial part of every Android app. It contains essential information about the app, including:

* **Application Component Declaration**: Lists Activities, Services, Broadcast Receivers, and Content Providers.
* **Permissions**: Declares what resources or actions the app needs access to (e.g., internet, camera).
* **Intent Filters**: Specifies the types of intents that a component can handle.

Example of Manifest:

xml

Copy code

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

package="com.example.myapp">

<application

android:name=".MyApp"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<service android:name=".MyService" />

</application>

</manifest>

**9. Permissions in Android**

Permissions are used to control access to certain features or data in Android. For example, apps need explicit permissions to access the camera, contacts, or internet.

* **Normal Permissions**: These do not affect user privacy, such as accessing the internet.
* **Dangerous Permissions**: These permissions can affect user privacy and need to be explicitly granted by the user (e.g., location, camera).

Example of declaring permissions in the manifest:

xml

<uses-permission android:name="android.permission.INTERNET" />

**10. Managing Application Resources in a Hierarchy**

Android organizes resources (e.g., images, layouts, strings) in a hierarchy:

* **Res Folder**: The res folder stores all application resources.
  + **drawable**: Images and XML files.
  + **layout**: XML layout files.
  + **values**: Strings, colors, and other constants.

Resources are organized in subdirectories (e.g., res/layout/, res/drawable/) to provide different versions of resources for different device configurations (e.g., screen size, orientation).

**11. Working with Different Types of Resources**

Android provides various types of resources:

* **Strings**: Store all text-related resources in res/values/strings.xml.
* **Drawables**: Store image resources in the res/drawable/ folder.
* **Layouts**: Store UI layout files in res/layout/.
* **Colors**: Define color resources in res/values/colors.xml.