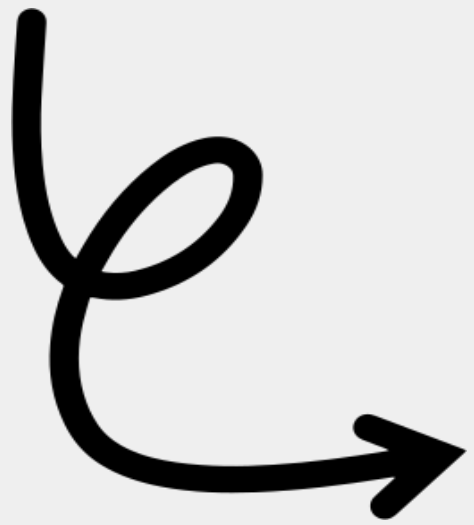
The Android Prep Lab (Kotlin)

Part - 1 of 10
Core Android Components



What is an Activity ?

An **Activity** is a single, focused thing that a user can do. In Android, an activity represents a screen with a user interface. For example, an email app might have one activity showing a list of new emails, another activity to compose an email, and a third to read emails. Activities are the entry points for interacting with the user and often work together to form a cohesive user experience.

Describe the lifecycle of an Activity.

The Activity lifecycle is a series of states an activity goes through from creation to destruction. Key methods include:

- onCreate(): Called when the activity is first created.
- onStart(): Called when the activity becomes visible to the user.

Ctnd…

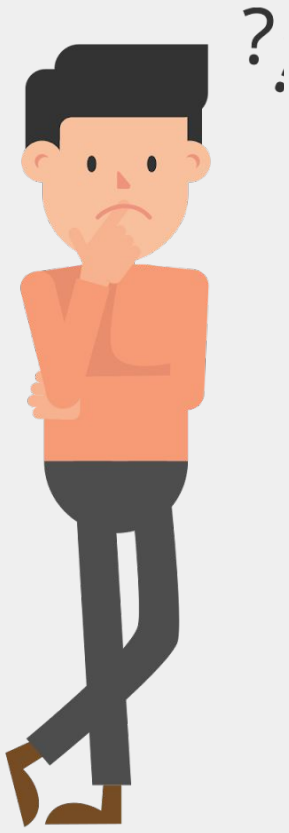
Ctnd…

- onResume(): Called when the activity starts interacting with the user.
- onPause(): Called when the activity is partially obscured.
- onStop(): Called when the activity is no longer visible.
- onRestart(): Called if the activity is coming back to the foreground.
- onDestroy(): Called before the activity is destroyed.

Understanding these states helps in managing resources efficiently and handling user interaction smoothly.

Explain the differences between an Activity and a Service.

- **UI Presence**: Activities have a user interface, while Services do not.
- **Lifecycle**: Activities are visible and have an interactive lifecycle, whereas Services are designed for background operations and have a simpler lifecycle.
- **Use Case**: Activities are for user interaction, while Services handle tasks that should continue running without user interaction.



What is a Broadcast Receiver?

A **Broadcast Receiver** is an Android component that allows the system to send messages to apps. These messages are known as broadcasts and can notify an app of events like battery status, Wi-Fi connectivity, or when the device boots up.

For example, an app can register a Broadcast Receiver to listen for the android.intent.action.BATTERY_LOW intent to take action when the device battery is low.

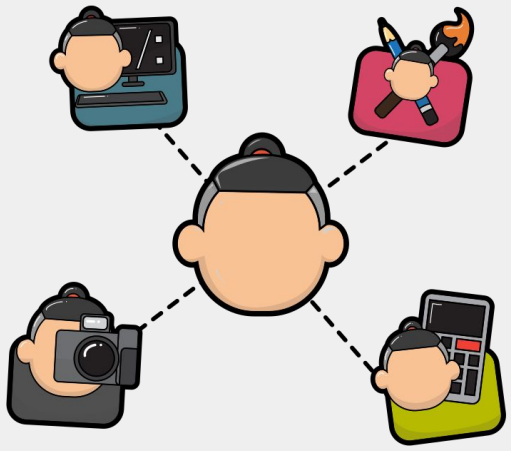
How do you register a Broadcast Receiver?

Broadcast Receivers can be registered in two ways:

- **Static Registration**: Declared in the AndroidManifest.xml file with specific intents.
- **Dynamic Registration**: Registered programmatically in code (typically in an Activity or Service) using registerReceiver().

Dynamic registration allows more flexibility and control over the receiver's lifecycle.

What is a Content Provider, and why is it used?



A **Content Provider** is a component that enables data sharing between applications. It provides a structured interface to access, insert, update, or delete data. Content Providers use URIs to uniquely identify the data and facilitate cross-app data sharing, such as allowing one app to access contact information stored by another app.

Common Content Providers include Contacts, MediaStore, and Calendar

What are Intents in Android?

An **Intent** is a messaging object used to request an action from another app component. Intents facilitate communication between components, such as starting an Activity, Service, or sending a broadcast.

Types of Intents:

- **Explicit Intent**: Specifies the component to start by name. Used for internal app communication.
- **Implicit Intent**: Specifies the type of action to perform, allowing the system to determine the best component to respond. Used for inter-app communication.

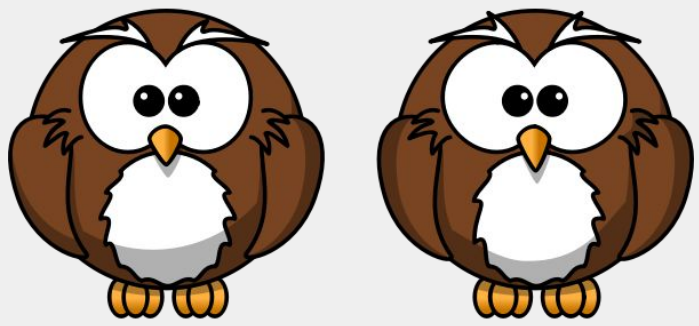
Describe the Intent Filter and its purpose.

An **Intent Filter** is an expression in the app manifest that specifies the types of intents an Activity, Service, or Broadcast Receiver can respond to. It includes:

- **Action**: The general action to be performed (e.g., ACTION_VIEW).
- **Data**: The URI and MIME type of data the intent filter can handle.
- **Category**: Additional information about the action, such as CATEGORY_DEFAULT.

Intent filters allow an app component to be accessible by other apps for specific actions.

What is the difference between an Implicit and an Explicit Intent?



- **Explicit Intent**: Directly specifies the component to be called, such as an Activity in the same app. Example: Intent(this, DetailActivity::class.java).
- **Implicit Intent**: Only specifies the action to be performed, and Android decides which app can best handle it. Example: Intent(Intent.ACTION_VIEW, Uri.parse("http://www.example.com")).

Implicit intents are generally used for inter-app communication, while explicit intents are used within the same app.

How does the onStartCommand() method work in a Service?

onStartCommand() is called whenever a client starts a service using startService(). It takes an Intent parameter and three possible return types:

- **START_STICKY**: Tells the system to recreate the service after it is killed, without redelivering the last intent.
- **START_NOT_STICKY**: Tells the system not to recreate the service unless there are pending intents to deliver.
- **START_REDELIVER_INTENT**: Tells the system to restart the service and redeliver the last intent.

This allows control over the service's behavior after it is stopped or killed.

Explain how data is shared between components in Android.

Data can be shared between components using:

- **Intents and Bundles**: Data is passed between Activities or Services using extras in an Intent.
- **SharedPreferences**: Stores key-value pairs, ideal for lightweight data.
- **Content Providers**: Share structured data between apps using URIs.
- **Internal/External Storage**: For larger files, images, or media.

Each method serves different needs depending on data size and accessibility

What is a PendingIntent and where is it commonly used?

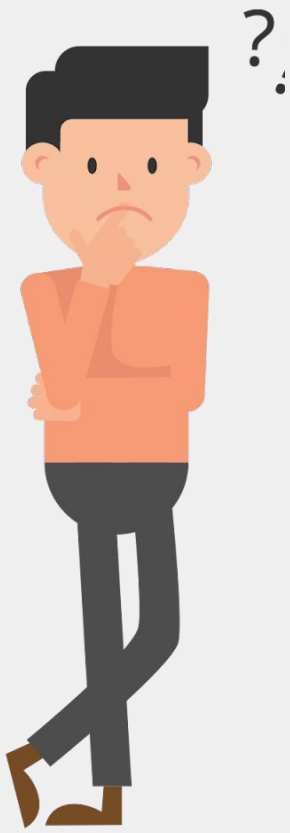
A **PendingIntent** is a token that another app, such as a notification manager, can use to trigger an intent at a later time on behalf of your application. It’s commonly used for:

- **Notifications**: To trigger specific actions when a user taps on a notification.
- **Alarms**: To specify actions that should happen at scheduled times.
- **Widget Updates**: To control widget actions.

A PendingIntent ensures that the specified action runs with the original app’s identity and permissions.

What is an Application class in Android?

The **Application** class in Android represents the entire application context and is the base class for maintaining global application state. It is initialized before any activity, service, or receiver objects when the process for the app is created. Developers often extend this class to set up global variables or services, such as initializing libraries.



What are Content URIs in Android?

A **Content URI** is a URI that identifies data in a Content Provider. The format typically includes:

- **Authority**: Unique name for the Content Provider.
- **Path**: Specifies the type of data.
- **ID (Optional)**: Identifies a specific record.

Example: content://com.example.provider/contacts/123, where 123 identifies a particular contact. Content URIs are essential for accessing and modifying data within Content Providers.

https://www.linkedin.com/in/pavankreddy92/

Follow for More