Android Services

Android - Service

- A Service is an application component that can perform long running operations in the background and does not provide a user interface.
- Another application component can start a service and it will continue to run in the background even if the user switches to another application.
- Additionally, a component can bind to a service to interact with it and even perform inter process communication (IPC).
- For example, a service might handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

Android - Service

- A **Background** Service is a service that runs only when the app is running so it'll get terminated when the app is terminated.
- A **Foreground** Service is a service that stays alive even when the app is terminated.
- A **Bound** Service is a service that runs only if the component it is bound to is still active.

Background Service

- To create a Background Service : a) create a new Class and have it extend the Service class.
  b) Inside the class, override the onBind() and onStartCommand() methods.
- The **onStartCommand()** method is called every time we start the service by calling startService(). This is where we want to define what the service will do.
- Go to the manifests file. Inside the application element, add a service element and use the **android:name** attribute to add the service to the app.
- Call **startService()** and pass in the intent.

Android - Service

- A service can essentially take two forms:

State	Description
Started	A service is **started** when an application component, such as an activity, starts it by calling *startService()*. Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.
Bound	A service is **bound** when an application component binds to it by calling *bindService()*. A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with inter-process communication (IPC).

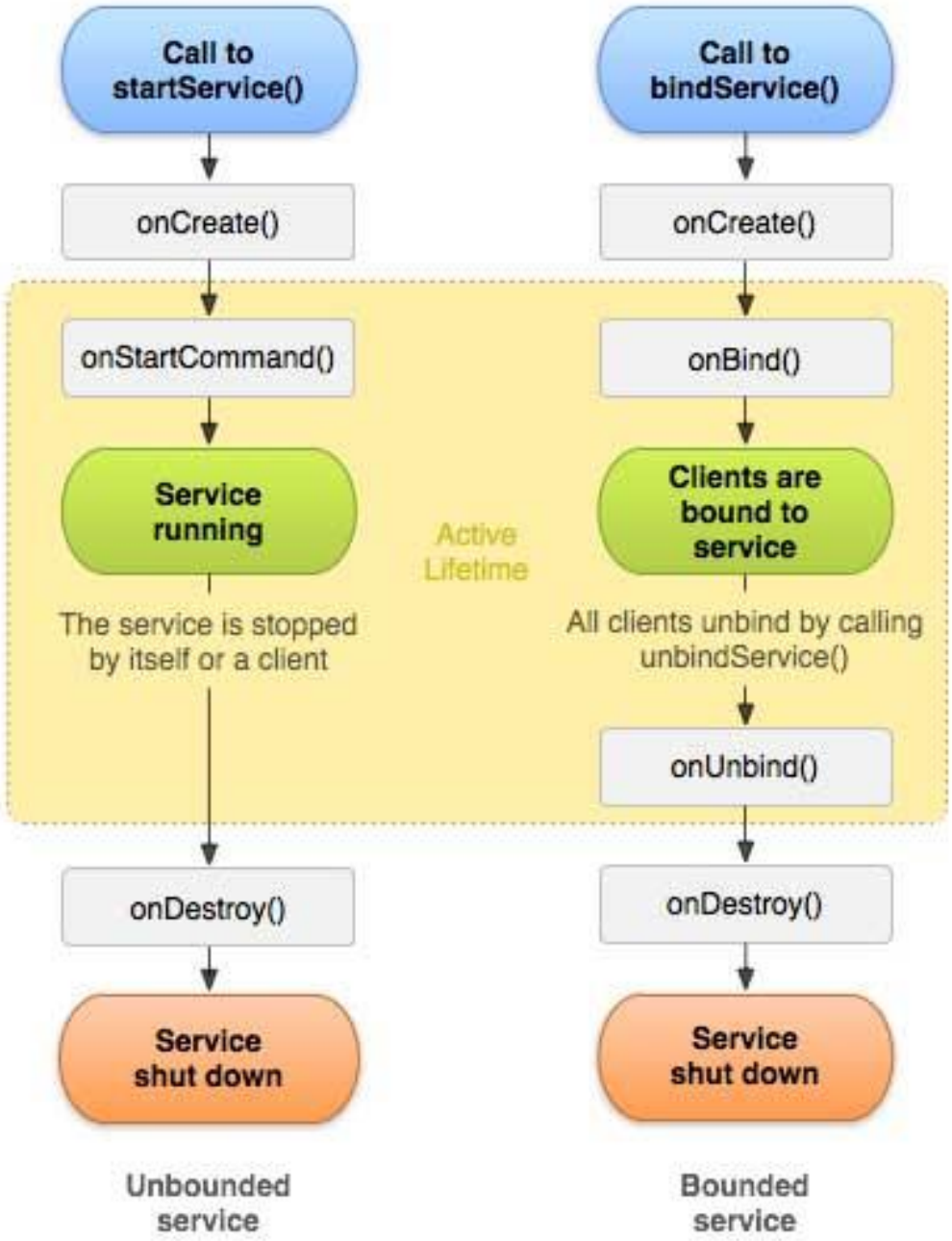
Service Forms

- **Started (Unbounded) Service** : Started service is started when an application component calls a startService() method. Once the service is initiated, it can run continuously even if the component which started the service is being destroyed. This service is seen in the Gmail application when it syncs the data in the background. The sync operation will still run if we close the application from the background. This service can stop by calling stopService() method.

Service Forms

- **Bounded Service :** The Bounded service is started when an application component calls bindService() method. The service started using bindService() is linked with the component from which we have started that service. This service is stopped when the component linked to that service is unbind.
- This service is seen in the Music player application when the music is being played in the background of the application and when we close the application from the background the component bound with the service unbounds and hence the service stops. This service can stop by calling the unbindService() method.

Service Life Cycle



Callback	Description
onStartCommand()	The system calls this method when another component, such as an activity, requests that the service be started, by calling *startService()*. If you implement this method, it is your responsibility to stop the service when its work is done, by calling *stopSelf()* or *stopService()* methods.
onBind()	The system calls this method when another component wants to bind with the service by calling *bindService()*. If you implement this method, you must provide an interface that clients use to communicate with the service, by returning an *IBinder* object. You must always implement this method, but if you don't want to allow binding, then you should return *null*.
onUnbind()	The system calls this method when all clients have disconnected from a particular interface published by the service.
onRebind()	The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its *onUnbind(Intent)*.
onCreate()	The system calls this method when the service is first created using *onStartCommand()* or *onBind()*. This call is required to perform one-time setup.
onDestroy()	The system calls this method when the service is no longer used and is being destroyed. Your service should implement this to clean up any resources such as threads, registered listeners, receivers, etc.