**Service**

* A [Service](https://developer.android.com/reference/android/app/Service) is an [application component](https://developer.android.com/guide/components/fundamentals#Components) that can perform long-running operations in the background. It does not provide a user interface.
* A service can run continuously in the background even if the application is closed or the user switches to another application.
* application components can bind to a service to interact with it and even perform interprocess communication (IPC). For example, a service can handle network transactions, play music, perform file I/O, or interact with a content provider,
* There is a major difference between android services and threads, one must not be confused between the two. Thread is a feature provided by the Operating system to allow the user to perform operations in the background. While service is an [android component](https://www.geeksforgeeks.org/components-android-application/) that performs a long-running operation about which the user might not be aware of as it does not have UI.
* A service runs in the main thread of its hosting process; the service does not create its own thread and does not run in a separate process unless you specify otherwise.
* You should run any blocking operations on a separate thread within the service to avoid Application Not Responding (ANR) errors.

These are the three different types of services:

1. **Foreground Services**

Services that notify the user about its ongoing operations are termed as Foreground Services. Foreground services must display a [Notification](https://developer.android.com/develop/ui/views/notifications)

1. **Background Services**

Background services do not require any user intervention. These services do not notify the user about ongoing background tasks and users also cannot access them.

1. **Bound Services**

Bound services perform their task as long as any application component is bound to it.

More than one component is allowed to bind themselves with a service at a time, but when all of them unbind, the service is destroyed.

the life cycle of service will follow two different paths Started or Bound.

1. **Started Service (Unbounded Service):**

**startService()**  **–** application component call this method to **Start the**

service .

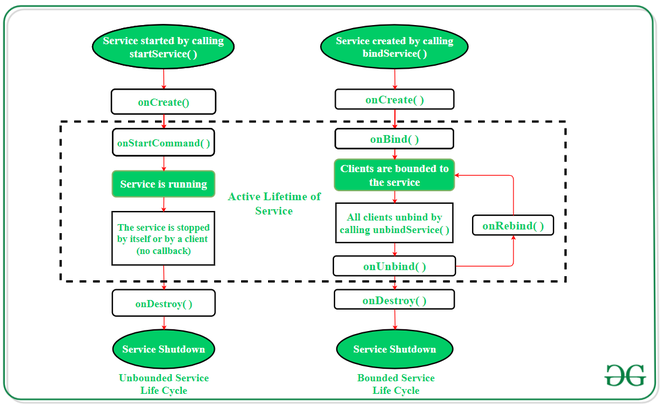
**stopService() –** to stop the execution of service.

**stopSelf() –** The service can stop itself by calling.

#### Bounded Service:

**bindService() –** to bind an application component with a service

**unbindService() – to** unbind all the components from the service



| Methods | Description |
| --- | --- |
| onStartCommand() | * The Android service calls this method when a component (e.g. activity) * requests to start a service using startService(). Once the service is started, * it can be stopped explicitly using stopService() or stopSelf() methods. |
| onBind() | * This method is mandatory to implement in android service. * Invoked whenever an application component calls the bindService() method in order to bind itself with a service. * User-interface is also provided to communicate with the service effectively by returning an IBinder object. * If the binding of service is not required then the method must return null. |
| onUnbind() | * The Android system invokes this method when all the clients get disconnected from a particular service interface. |
| onRebind() | * Once all clients are disconnected from the particular interface of service and there is a need to connect the service with new clients,   the system calls this method. |
| onCreate() | * Whenever a service is created either using onStartCommand() or onBind(), the android system calls this method. This method is necessary to perform a one-time-set-up. |
| onDestroy() | * When a service is no longer in use, the system invokes this method just before the service destroys as a final clean up call. * Services must implement this method in order to clean up resources like registered listeners, threads, receivers, etc. |