

Jimma University
JIT
**Faculty of computing
& Informatics**

**Mobile Application
development**

Chapter Three

Intents and Services

Chapter Objectives

At the end the chapter students should able to know:

- • Describe Intents and services
- • Understand when and how to use Intents and services

Android Intent

- ▶ **Android Intent** is an object carrying an intent .
- ▶ Intent is the message that is passed between components , such as activities, content providers, broadcast receivers, services etc, *with-in* the application or *outside* the application.
- ▶ The intent itself, an Intent object, is a passive data structure holding an abstract description of an operation to be performed.

Android Intent


Android intents are mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

Android Intent

Component	Description
Starting an activity	By Sending Intent object to <u>startActivity()</u> method we can start new activity
Starting a service	By Sending Intent object to <u>startService()</u> method we can start new service or send required instruction to an existing service
Delivering broadcast	By Sending Intent object to <u>startBroadcast()</u> method we can deliver our message to other app broadcast receiver

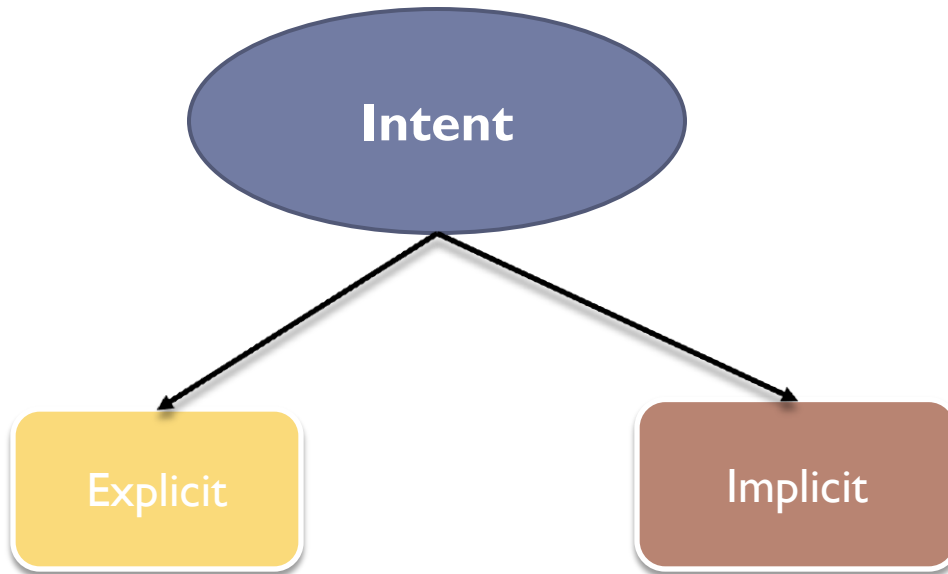
Intent



Android intent my
take the following
parameters

- **Component:** name of application component
- **Action :** general task that need to be performed by the component
- **Category:** it's a string containing additional information about the kind of component that should handle the intent.
- **Data:** URI
- **Extras:** additional information that should be delivered to the component(key value pairs)
- **Flags:** instruct the Android system how to launch an activity, and how to treat it after it's launched etc

Types of Intent



EXPLICIT INTENTS

- ▶ These intents designate the target component by its name and they are typically used for *application-internal messages*
- ▶ such as an activity starting a subordinate service or launching a sister activity(*application component*)

```
// Explicit Intent by specifying its class name
Intent i = new Intent(this, TargetActivity.class);
i.putExtra("Key1", "ABC");
i.putExtra("Key2", "123");

// Starts TargetActivity
startActivity(i);
```


EXPLICIT INTENTS

- ▶ The target component which receives the intent can use the **getExtras()** method to get the extra data sent by the source component. For example:

```
// Get bundle object at appropriate place in your code
Bundle extras = getIntent().getExtras();

// Extract data using passed keys
String value1 = extras.getString("Key1");
String value2 = extras.getString("Key2");
```

IMPLICIT INTENTS

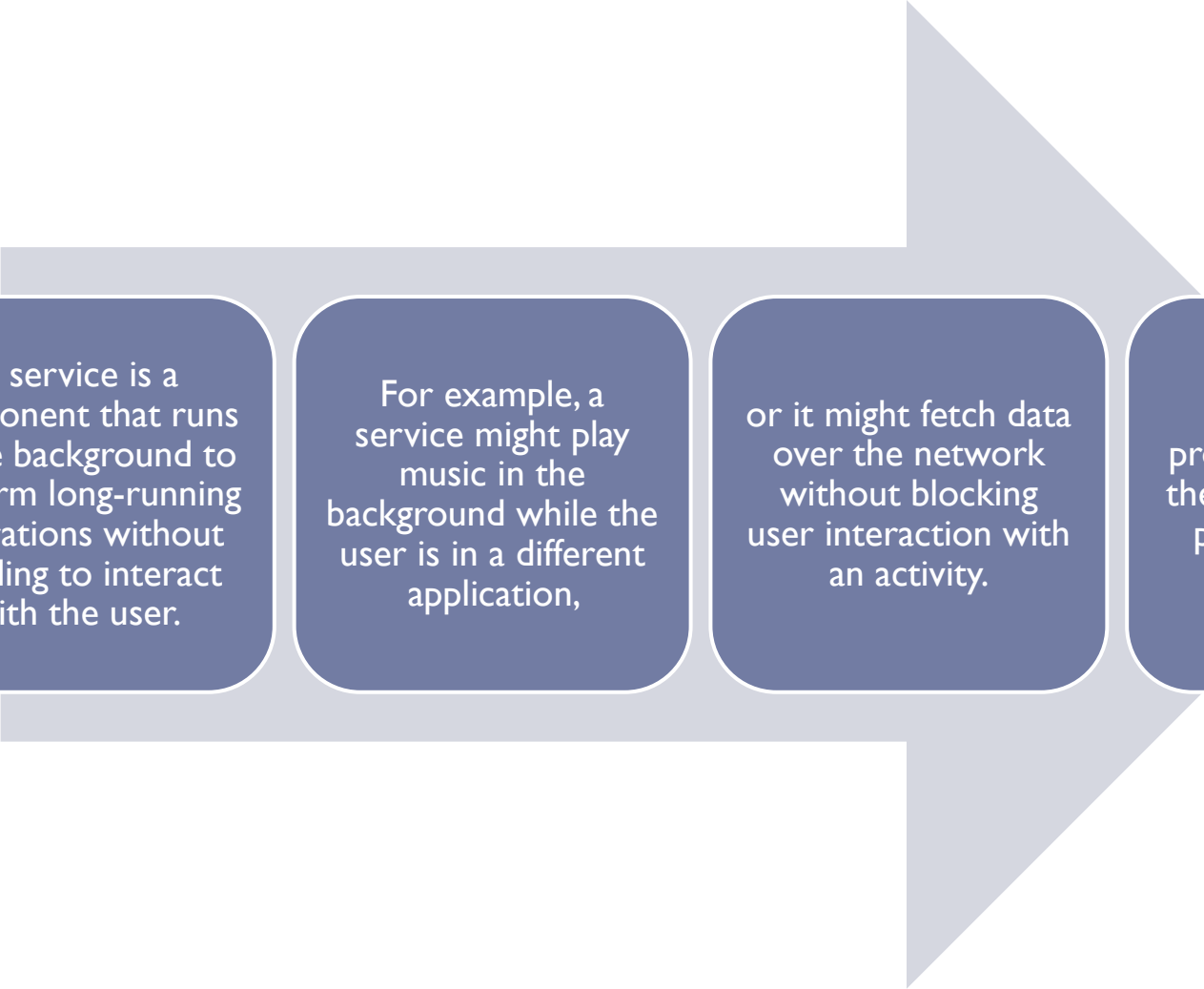
- ▶ These intents do not name a target and the field for the component name is left blank.
- ▶ Implicit intents are often used to activate components in other applications. For example:

```
// Implicit Intent by specifying a URI
Intent i = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.example.com"));

// Starts Implicit Activity
startActivity(i);
```

- ▶ This is useful when your applications is not capable of doing it but it is important action

Services



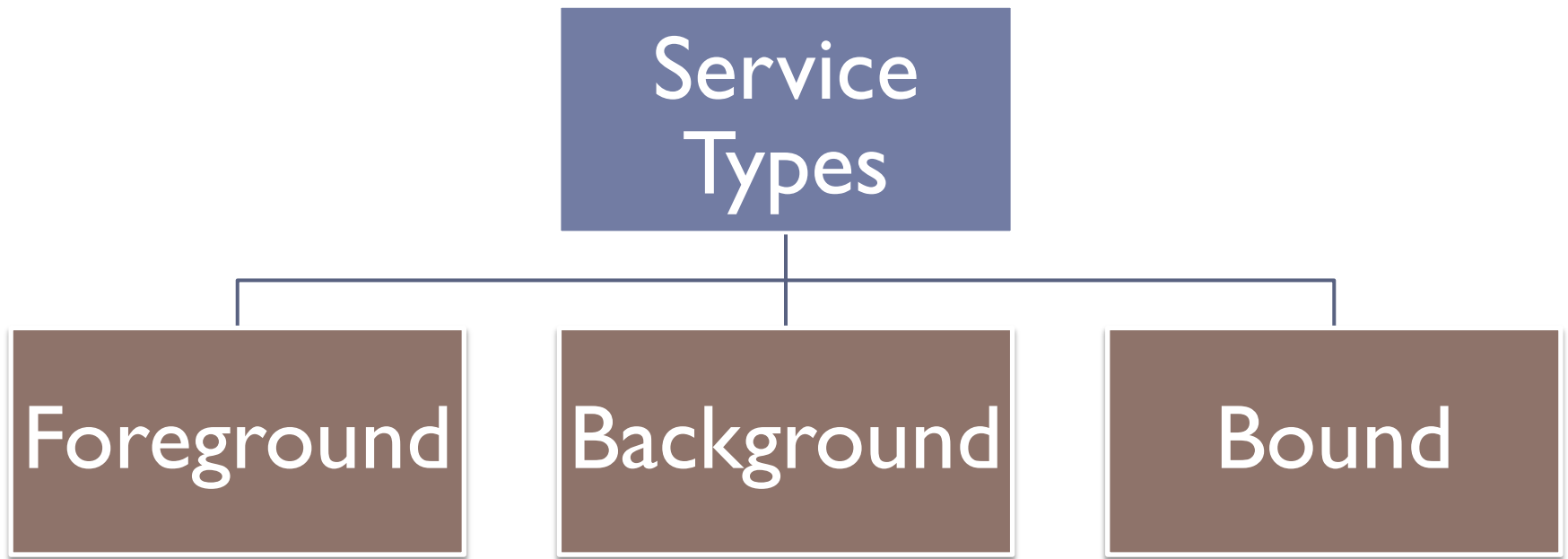
A service is a component that runs in the background to perform long-running operations without needing to interact with the user.

For example, a service might play music in the background while the user is in a different application,

or it might fetch data over the network without blocking user interaction with an activity.

don't need to provide some UI for the operations to be performed in the background.

Service Types



Foreground Service

- ▶ **Foreground Service** are services that visible to users
- ▶ It is used for operation that is noticeable (status bar notification) to the user.
- ▶ Foreground services continue running even when the user isn't interacting with the app.
- ▶ A perfect example is **Music player** and **downloading**
- ▶ This notification cannot be dismissed unless the service is either stopped or removed from the foreground.

Background Service

- ▶ A background service performs an operation that *isn't directly* noticed by the user.
- ▶ Service that run background, such that the user cant see or access them
- ▶ These are the tasks that don't need the user to know
- ▶ Example : Syncing and storing data
- ▶ But for the API level 21 or higher, the Android System imposes some restrictions while using the Background Service.(Be aware before using it)

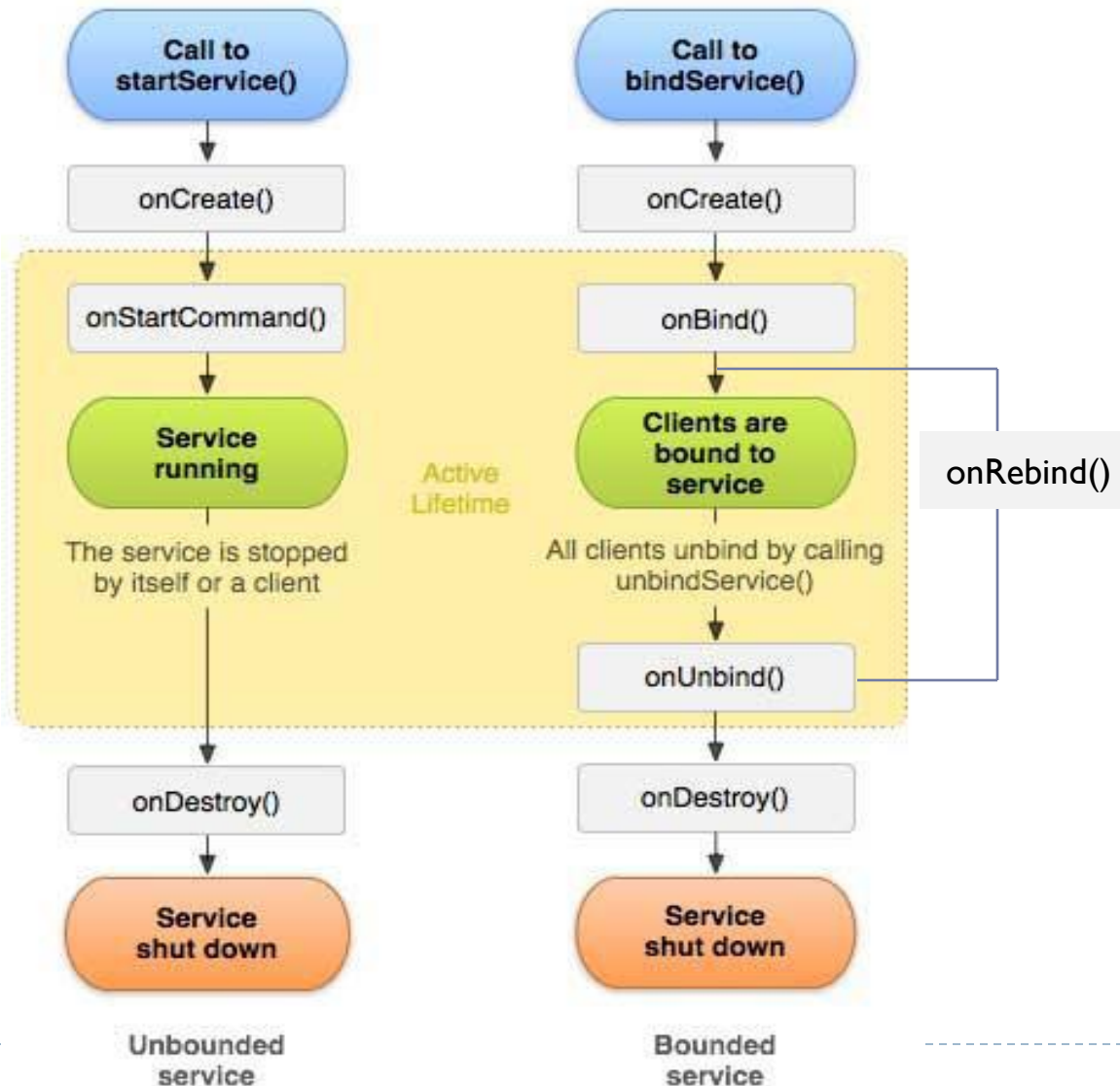
Bound Service

- ▶ **Bound Services** run as long as some other application component bind to it.
- ▶ A service is bound when an application component binds to it by calling bindService() method.
- ▶ This type of service runs until other application components are attached, or bound, to them.
- ▶ Multiple components can be bound to a service and in this case, the service only dies when all of these multiple components unbind from the service.

Life cycle of Android Services

No.	State & Description
1	<p><u>Started</u></p> <p>A service is started when an application component, such as an activity, starts it by calling <u>startService()</u>. Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.</p>
2	<p><u>Bound</u></p> <p>A service is bound when an application component binds to it by calling <u>bindService()</u>. 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 interprocess communication (IPC)</p>

Life cycle of Android Services



Methods of Android Services

Callback	Description
<code>onStartCommand()</code>	The system calls this method when another component, such as an activity, requests that the service be started, by calling <code>startService()</code> . If you implement this method, it is your responsibility to stop the service when its work is done, by calling <code>stopSelf()</code> or <code>stopService()</code> methods.
<code>onBind()</code>	The system calls this method when another component wants to bind with the service by calling <code>bindService()</code> . If you implement this method, you must provide an interface that clients use to communicate with the service, by returning an <code>IBinder</code> object. You must always implement this method, but if you don't want to allow binding, then you should return null.
<code>onUnbind()</code>	The system calls this method when all clients have disconnected from a particular interface published by the service.

Methods of Android Services

Callback	Description
<code>onRebind()</code>	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 <code>onUnbind(Intent)</code>
<code>onCreate()</code>	The system calls this method when the service is first created using <code>onStartCommand()</code> or <code>onBind()</code> . This call is required to perform one-time setup.
<code>onDestroy()</code>	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.



End !!!

Reading assignment

Intent Filter