Android Application Components

The basic components of an Android application are:

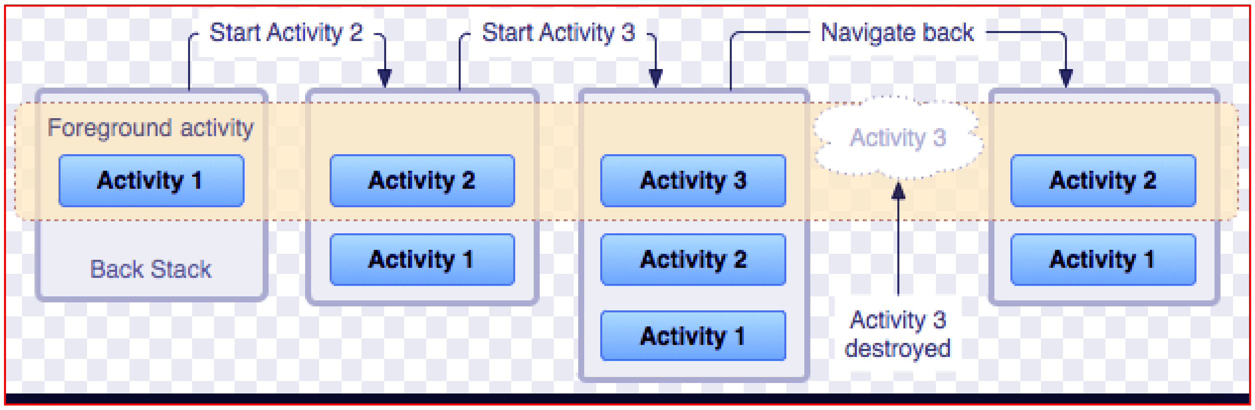
1. Activities

An activity is a class that is considered as an entry point for users that represents a single screen. A messenger application might have an activity that shows a new notification, another activity which reads messages and another which composes a new message.

Each activity is independent of one another. **For example –** camera application can be started in an email application to compose an email that shares an image. The picture below depicts how each new activity adds an item to back stack and how the current activity is destroyed and previous activity is resumed. We will study the**life cycle of activity** in detail in our ***Android Activity*** article.

To implement an activity, extend the Activity class in your subclass:

```
public class MainActivity extends Activity {
//code
}
```



2. Services

A service is a component that runs in the background, it acts as an invisible worker of our application. It keeps updating data sources and activities. It also **broadcasts intents** and performs tasks when applications are not active. An example of service is we can surf the internet or use any other application while listening to music.
To execute services, extend the Services class in your sub-class:

```
public class MyService extends Services {
//code
}
```

Explore Android Services Tutorial and get a detailed insight into the concept

3. Content Providers

Content Provider is a component that allows applications to share data among multiple applications. It hides the details of the database and can be used to read and write private data of the application which is not shared. It would be a mess to access data from other applications without ***content providers***.

For example – you can consider looking for contact details in contact list. Or You might want photos from the gallery which are also provided by Content Provider.
To implement this, extend ContentProvider in your subclass:

```
public class Provider_Name extendsContentProvider {
//code
}
```

4. Broadcast Receiver

Broadcast Receiver is a component that responds to broadcast messages from another application or the same system. It can also deliver broadcasts to applications that are not running. **For example –** notify the user that the battery is low. Android developers can use broadcast messages in the application or outside the normal flow.

To implement this, extend BroadcastReceiver to your receiver:

```
public class Broadcast_Name extendsBroadcastReceiver {
//code
}
```

Get a thorough understanding of [Android Broadcast Receiver]

Additional Components of Android Application

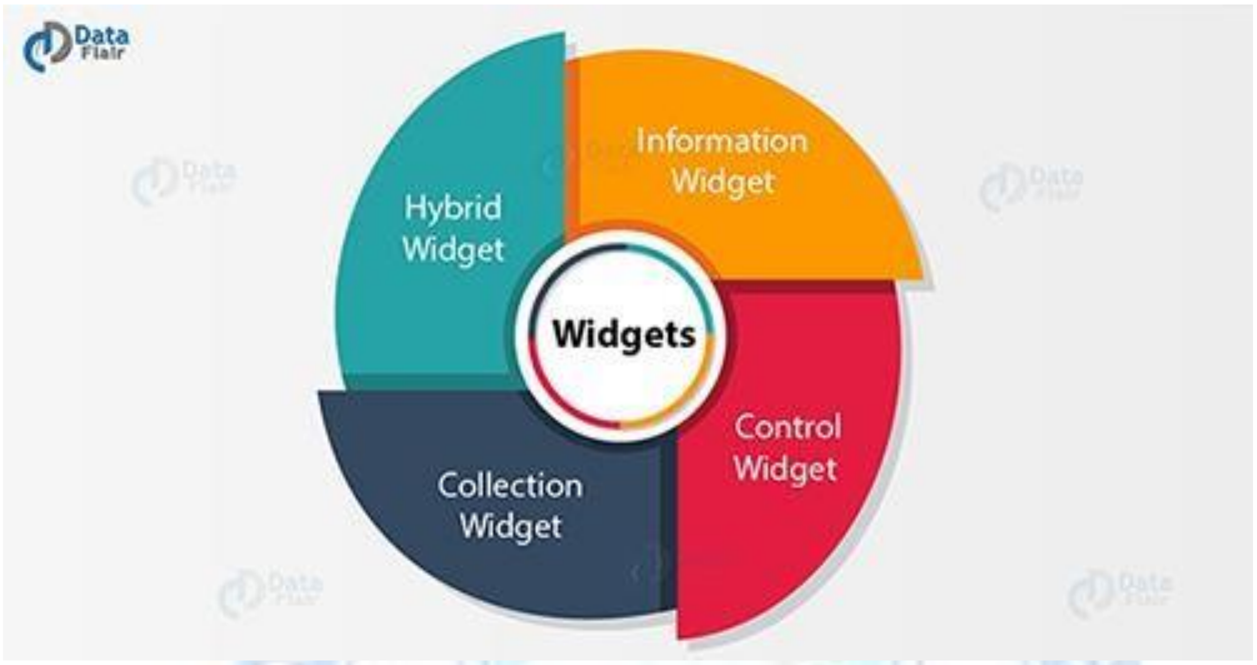
Some additional components of an android application:

1. Intents

It is an inter-application message passing framework for communication between android components. It is also used for transferring data between different Activities as well as to start a new service and display a list of contacts in ListView. Example – the camera application sends an intent to the operating system when the user decides to share a picture.

2. Widgets

Widgets are variations of Broadcast Receivers and essential aspects of **home screen customization**. They display data and allow users to perform actions on them. There are various types of widgets:



- **Information widget:** These widgets display crucial information and track how the information changes over time. Example – Clock widgets and widgets that display weather and time information.
- **Collection widget:** As the name depicts, collection widgets are a collection of information of the same type. Its use is for browsing information and opening any one of the elements to view details. Example – music widgets, as we can skip pause and play music outside the music application.
- **Control widget:** These widgets display functionalities and by using them, the user can trigger from home screen

without opening the application. Example – pause and play the video outside the application.

- **Hybrid widget:** These widgets combine features of all the other three widgets. Example – music player widget is a control widget but it also informs the user about which track is playing currently, which means it is a combination of control and information thus it is termed as hybrid widget.

3. Views

View is responsible for drawing and event handling. They are rectangular elements on the screen. Some of the views are **EditText, ImageView Button, CheckBox and ImageButton.**

4. Notifications

It alerts users when the application is not visible or is inactive. This alert flashes on the screen and then disappears. Example – Notification of the new incoming message popped on the screen.

5. Fragments

A fragment is a portion of the total user interface. Users can combine more than one fragment in a single activity and these fragments can be reused in multiple activities. A fragment generally contains **Views** and **ViewGroups** inside them.

6. Layout XML Files

Layout is the structure for the user interface in the application. XML files provide different types of layouts for the different type of screen, it also specifies which **GUI component**, an activity or fragment holds.

7. App APK files

Apk file is the package file format that contains the program's code, resources, assets. The Android operating system uses them for installing mobile applications and middleware.

8. Resources

Resources in Android is for defining Images, texts, string values. Everything is defined in the resource file and it can be referenced within the source code. We will learn about Android Resources, in detail in our next upcoming article on Resources.

Android Activity Lifecycle with Callback Methods & Usage

Android Activity provides users with a frame to interact with & perform their actions. Since an activity interacts with the user, it designs a window to hold UI elements. An interactive application has many activities providing a screen & interacting with each other.

What is Activity?

We know by now, that an activity is a single screen of an application that lets us see and interact to perform an activity. Usually, an application contains many screens and each screen extends **Activity()** class.

Data Flair
Android Activity Tutorial
Android Apps
Main Activity
Child Activity

When we work on an application what we see is a UI on the screen which is an activity. Most of the applications that we use have many activities. Among all those activities, one is the **MainActivity()** & the rest are its **ChildActivities()**. Generally, the first page that appears on the screen when an application opens is referred to as MainActivity(). This main activity interacts with the child activities and lets the user access them.

Activities are stored in a stack of Activities, wherein the current activity holds the highest position.

Android Activity Lifecycle

An activity can have four states, which are :

1. Running
2. Paused
3. Resumed
4. Stopped

The above are the four states that Android activity can achieve during its whole lifecycle.

1. Running State

An activity is in the **running** state if it’s shown in the foreground of the users’ screen. Activity is in the running state when the user is interacting with it.

2. Paused State

When an activity is not in the focus but is still alive for the user, it’s in a **paused**state. The activity comes in this state when some other activity comes in with a higher position in the window.

3. Resumed State

It is when an activity goes from the paused state to the foreground that is an **active**state.

4. Stopped State

When an activity is no longer in the activity stack and **not visible** to the users.

Android Activity Methods

Android activities go through four states during their entire lifecycle. These activities have callback methods() to describe each activity in each of the four stages. These methods need to be overridden by the implementing subclass. In Android, we have the following 7 callback methods that activity uses to go through the four states:

1. onCreate()
2. onStart()
3. onPause()
4. onRestart()
5. onResume()
6. onStop()
7. onDestroy()

We'll understand these in the following:

1. onCreate()

The Android oncreate() method is called at the very start when an activity is created. An activity is created as soon as an application is opened. This method is used in order to create an Activity.

Syntax:

```
@Override protected void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
...
}
```

2. onStart()

The Android onstart() method is invoked as soon as the activity becomes visible to the users. This method is to start an activity. The activity comes on the forescreen of the users when this method is invoked.

Syntax:

```
@Override protected void onStart()
{
super.onStart();
...
}
```

3. onPause()

The Android onPause() method is invoked when the activity doesn’t receive any user input and goes on hold. In the pause state, the activity is partially visible to the user. This is done when the user presses the back or home buttons. Once an activity is in the pause state, it can be followed by either **onResume()** or **onStopped()**callback method.

Syntax:

```
@Override protected void onPause()
{
super.onPause();
...
}
```

4. onRestart()

The Android onRestart() method is invoked when activity is about to start from the stop state. This method is to restart an activity that had been active some time back. When an activity restarts, it starts working from where it was paused.

Syntax:

```
@Override protected void onRestart()
{
super.onRestart();
...
}
```

5. onResume()

The Android onResume() method is invoked when the user starts interacting with the user. This callback method is followed by **onPause()**. Most of the functionalities of an application are implemented using **onResume()**.

Syntax:

```
@Override protected void onResume()
{
super.onResume();
...
}
```

6. onStop()

The Android onStop() method is invoked when the activity is no longer visible to the user. The reason for this state is either activity is getting destroyed or another existing activity comes back to **resume** state.

Syntax:

```
@Override protected void onStop()
{
```

```
super.onStop();
...
}
```

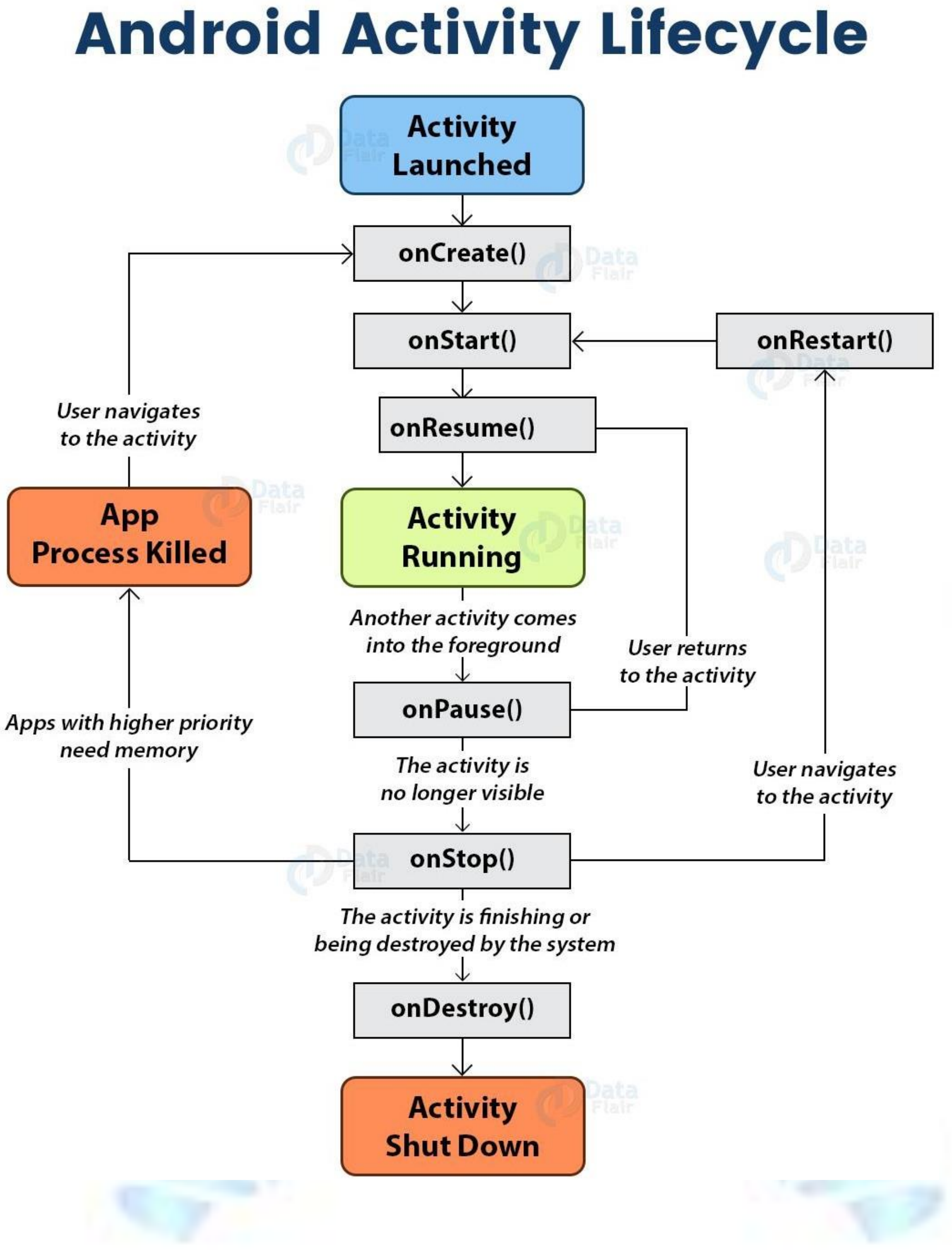
7. onDestroy()

The Android onDestroy() is the method that is called when an activity finishes and the user stops using it. It is the final callback method received by activity, as after this it is destroyed.

Syntax:

```
@Override protected void onDestroy()
{
super.onDestroy();
...
}
```

Understand the flow of Activity through the four states using the seven methods from the diagram:


Android Activity Lifecycle
Activity Launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App Process Killed
Activity Running
Another activity comes into the foreground
User returns to the activity
onPause()
Apps with higher priority need memory
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity Shut Down