









- [Go to your profile](#)
- [Hire a developer](#)
- [Apply as a developer](#)
- [Log in](#)

•

- [Top 3%](#)
- [Why](#)
- [Clients](#)
- [Enterprise](#)
- [Community](#)
- [Blog](#)
- [About Us](#)
- [Go to your profile](#)
- [Hire a developer](#)
- [Apply as a developer](#)
- [Log in](#)
- - Questions?
 - [Contact Us](#)
 - 
 - 
 - 

• Questions?

- [Contact Us](#)
- 
- 
- 

[Hire a developer](#)

20 Essential Android Interview Questions *

- 1.4Kshares



•



•



•

[Submit an interview question](#)[Submit a question](#)

Looking for experts? Check out Toptal's [Android developers](#).



There are four Java classes related to the use of sensors on the Android platform. List them and explain the purpose of each.

View the answer → Hide answer



The four Java classes related to the use of sensors on the Android platform areL

- Sensor: Provides methods to identify which capabilities are available for a specific sensor.
- SensorManager: Provides methods for registering sensor event listeners and calibrating sensors.
- SensorEvent: Provides raw sensor data, including information regarding accuracy.
- SensorEventListener: Interface that defines callback methods that will receive sensor event notifications.

To learn more about [sensors](#), refer to Android developer's guide.



What is a `ContentProvider` and what is it typically used for?

[View the answer](#) → [Hide answer](#)



A `ContentProvider` manages access to a structured set of data. It encapsulates the data and provide mechanisms for defining data security. `ContentProvider` is the standard interface that connects data in one process with code running in another process.

More information about content providers can be found [here](#) in the Android Developer's Guide.



Under what condition could the code sample below crash your application? How would you modify the code to avoid this potential problem? Explain your answer.

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType(HTTP.PLAIN_TEXT_TYPE); // "text/plain" MIME type
startActivity(sendIntent);
```

[View the answer](#) → [Hide answer](#)



An implicit intent specifies an action that can invoke any app on the device able to perform the action. Using an implicit intent is useful when your app cannot perform the action, but other apps probably can. If there is more than one application registered that can handle this request, the user will be prompted to select which one to use.

However, it is possible that there are no applications that can handle your intent. In this case, your application will crash when you invoke `startActivity()`. To avoid this, before calling `startActivity()` you should first verify that there is at least one application registered in the system that can handle the intent. To do this use `resolveActivity()` on your intent object:

```
// Verify that there are applications registered to handle this intent
// (resolveActivity returns null if none are registered)
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```

See the Android developer's guide for [more information](#) about implicit intents.

Find top Android developers today. Toptal can match you with the best engineers to finish your project.

[Hire Toptal's Android developers](#)



The last callback in the lifecycle of an activity is `onDestroy()`. The system calls this method on your activity as the final signal that your activity instance is being completely removed from the system memory. Usually, the system will call `onPause()` and `onStop()` before calling `onDestroy()`. Describe a scenario, though, where `onPause()` and `onStop()` would **not** be invoked.

[View the answer](#) → [Hide answer](#)



`onPause()` and `onStop()` will not be invoked if `finish()` is called from within the `onCreate()` method. This might occur, for example, if you detect an error during `onCreate()` and call `finish()` as a result. In such a case, though, any cleanup you expected to be done in `onPause()` and `onStop()` will not be executed.

Although `onDestroy()` is the last callback in the lifecycle of an activity, it is worth mentioning that this callback may not always be called and should not be relied upon to destroy resources. It is better have the resources created in `onStart()` and `onResume()`, and have them destroyed in `onStop()` and `onPause()`, respectively.

See the Android developer's guide for [more information](#) about the activity lifecycle.



Which of the code snippets below is the correct way to check if a Compass sensor is present on the system? Explain your answer.

Answer 1:

```
PackageManager m = getPackageManager();
if (!m.hasSystemFeature(PackageManager.FEATURE_SENSOR_COMPASS)) {
    // This device does not have a compass, turn off the compass feature
}
```

Answer 2:

```
SensorManager m = getSensorManager();
if (!m.hasSystemFeature(SensorManager.FEATURE_SENSOR_COMPASS)) {
    // This device does not have a compass, turn off the compass feature
}
```

Answer 3:

```
Sensor s = getSensor();
if (!s.hasSystemFeature(Sensor.FEATURE_SENSOR_COMPASS)) {
    // This device does not have a compass, turn off the compass feature
}
```

View the answer → Hide answer



The correct answer is **Answer 1**, the version that uses `PackageManager`.

`SensorManager` and `Sensor` are part of Android Sensor Framework and are used for direct access and acquisition of raw sensor data. These classes do not provide any method like `hasSystemFeature()` which is used for evaluation of system capabilities.

Android defines feature IDs, in the form of ENUMs, for any hardware or software feature that may be available on a device. For instance, the feature ID for the compass sensor is `FEATURE_SENSOR_COMPASS`.

If your application cannot work without a specific feature being available on the system, you can prevent users from installing your app with a `<uses-feature>` element in your app's manifest file to specify a non-negotiable dependency.

However, if you just want to disable specific elements of your application when a feature is missing, you can use the `PackageManager` class. `PackageManager` is used for retrieving various kinds of information related to the application packages that are currently installed on the device.

To learn more about [compatibility and handling different types of devices](#) or [sensors](#) please refer to the Android developer's guide.



Describe three common use cases for using an Intent.

View the answer → Hide answer



Common use cases for using an Intent include:

- **To start an activity:** You can start a new instance of an Activity by passing an Intent to `startActivity()` method.
- **To start a service:** You can start a service to perform a one-time operation (such as download a file) by passing an Intent to `startService()`.
- **To deliver a broadcast:** You can deliver a broadcast to other apps by passing an Intent to `sendBroadcast()`, `sendOrderedBroadcast()`, or `sendStickyBroadcast()`.

More information about [intents](#) can be found in Android developer's guide.



Suppose that you are starting a service in an Activity as follows:

```
Intent service = new Intent(context, MyService.class);
startService(service);
```

where `MyService` accesses a remote server via an Internet connection.

If the Activity is showing an animation that indicates some kind of progress, what issue might you encounter and how could you address it?

View the answer → Hide answer



Responses from a remote service via the Internet can often take some time, either due to networking latencies, or load on the remote server, or the amount of time it takes for the remote service to process and respond to the request.

As a result, if such a delay occurs, the animation in the activity (and even worse, the entire UI thread) could be blocked and could appear to the user to be “frozen” while the client waits for a response from the service. This is because the service is started on the main application thread (or UI thread) in the Activity.

The problem can (and should) be avoided by relegating any such remote requests to a background thread or, when feasible, using an asynchronous response mechanism.

Note well: Accessing the network from the UI thread throws a runtime exception in newer Android versions which causes the app to crash.



Normally, in the process of carrying out a screen reorientation, the Android platform tears down the foreground activity and recreates it, restoring each of the view values in the activity's layout.

In an app you're working on, you notice that a view's value is not being restored after screen reorientation. What could be a likely cause of the problem that you should verify, at a minimum, about that particular view?

View the answer → Hide answer



You should verify that it has a valid id. In order for the Android system to restore the state of the views in your activity, each view must have a unique ID, supplied by the `android:id` attribute.

More information is available [here](#).



What is DDMS? Describe some of its capabilities.

View the answer → Hide answer



DDMS is the [Dalvik Debug Monitor Server](#) that ships with Android. It provides a wide array of debugging features including:

- port-forwarding services
- screen capture
- thread and heap information
- network traffic tracking
- incoming call and SMS spoofing
- simulating network state, speed, and latency
- location data spoofing



What is the relationship between the life cycle of an AsyncTask and an Activity? What problems can this result in? How can these problems be avoided?

View the answer → Hide answer



An AsyncTask is not tied to the life cycle of the Activity that contains it. So, for example, if you start an AsyncTask inside an Activity and the user rotates the device, the Activity will be destroyed (and a new Activity instance will be created) but the AsyncTask will *not* die but instead goes on living until it completes.

Then, when the AsyncTask does complete, rather than updating the UI of the new Activity, it updates the *former* instance of the Activity (i.e., the one in which it was created but that is not displayed anymore!). This can lead to an Exception (of the type `java.lang.IllegalArgumentException: View not attached to window manager` if you use, for instance, `findViewById` to retrieve a view inside the Activity).

There's also the potential for this to result in a memory leak since the AsyncTask maintains a reference to the Activity, which prevents the Activity from being garbage collected as long as the AsyncTask remains alive.

For these reasons, using AsyncTasks for *long-running* background tasks is generally a bad idea . Rather, for *long-running* background tasks, a different mechanism (such as a service) should be employed.



What is an Intent? Can it be used to provide data to a ContentProvider? Why or why not?

View the answer → Hide answer



The Intent object is a common mechanism for starting new activity and transferring data from one activity to another. However, you cannot start a ContentProvider using an Intent.

When you want to access data in a `ContentProvider`, you must instead use the `ContentResolver` object in your application's `Context` to communicate with the provider as a client. The `ContentResolver` object communicates with the provider object, an instance of a class that implements `ContentProvider`. The provider object receives data requests from clients, performs the requested action, and returns the results.



What is the difference between a fragment and an activity? Explain the relationship between the two.

View the answer → Hide answer



An [activity](#) is typically a single, focused operation that a user can perform (such as dial a number, take a picture, send an email, view a map, etc.). Yet at the same time, there is nothing that precludes a developer from creating an activity that is arbitrarily complex.

Activity implementations can optionally make use of [the Fragment class](#) for purposes such as producing more modular code, building more sophisticated user interfaces for larger screens, helping scale applications between small and large screens, and so on. Multiple fragments can be combined within a single activity and, conversely, the same fragment can often be reused across multiple activities. This structure is largely intended to foster code reuse and facilitate economies of scale.

A fragment is essentially a modular section of an activity, with its own lifecycle and input events, and which can be added or removed at will. It is important to remember, though, that a fragment's lifecycle is directly affected by its host activity's lifecycle; i.e., when the activity is paused, so are all fragments in it, and when the activity is destroyed, so are all of its fragments.

More information is available [here](#) in the Android Developer's Guide.



What is difference between Serializable and Parcelable ? Which is best approach in Android ?

View the answer → Hide answer



`Serializable` is a standard Java interface. You simply mark a class `Serializable` by implementing the interface, and Java will automatically serialize it in certain situations.

`Parcelable` is an Android specific interface where you implement the serialization yourself. It was created to be far more efficient than `Serializable`, and to get around some problems with the default Java serialization scheme.



What are "launch modes"? What are the two mechanisms by which they can be defined? What specific types of launch modes are supported?

View the answer → Hide answer



A "launch mode" is the way in which a new instance of an activity is to be associated with the current task.

Launch modes may be defined using one of two mechanisms:

- **Manifest file.** When declaring an activity in a manifest file, you can specify how the activity should associate with tasks when it starts. Supported values include:
 - **standard** (default). Multiple instances of the activity class can be instantiated and multiple instances can be added to the same task or different tasks. This is the common mode for most of the activities.
 - **singleTop**. The difference from **standard** is, if an instance of the activity already exists at the top of the current task and the system routes the intent to this activity, no new instance will be created because it will fire off an `onNewIntent()` method instead of creating a new object.
 - **singleTask**. A new task will always be created and a new instance will be pushed to the task as the root. However, if any activity instance exists in any tasks, the system routes the intent to that activity instance through the `onNewIntent()` method call. In this mode, activity instances can be pushed to the same task. This mode is useful for activities that act as the entry points.
 - **singleInstance**. Same as **singleTask**, except that the no activities instance can be pushed into the same task of the **singleInstance**'s. Accordingly, the activity with launch mode is always in a single activity instance task. This is a very specialized mode and should only be used in applications that are implemented entirely as one activity.
- **Intent flags.** Calls to `startActivity()` can include a flag in the [Intent](#) that declares if and how the new activity should be associated with the current task. Supported values include:
 - **FLAG_ACTIVITY_NEW_TASK**. Same as **singleTask** value in Manifest file (see above).
 - **FLAG_ACTIVITY_SINGLE_TOP**. Same as **singleTop** value in Manifest file (see above).
 - **FLAG_ACTIVITY_CLEAR_TOP**. If the activity being started is already running in the current task, then instead of launching a new instance of that activity, all of the other activities on top of it are destroyed and this intent is delivered to the resumed instance of the activity (now on top), through `onNewIntent()`. *There is no corresponding value in the Manifest file that produces this behavior.*

More information about launch modes is available [here](#).



What is the difference between Service and IntentService? How is each used?

View the answer → Hide answer



Service is the base class for Android services that can be extended to create any service. A class that directly extends Service runs on the main thread so it will block the UI (if there is one) and should therefore either be used only for short tasks or should make use of other threads for longer tasks.

IntentService is a subclass of Service that handles asynchronous requests (expressed as “Intents”) on demand. Clients send requests through `startService(Intent)` calls. The service is started as needed, handles each Intent in turn using a worker thread, and stops itself when it runs out of work. Writing an IntentService can be quite simple; just extend the IntentService class and override the `onHandleIntent(Intent intent)` method where you can manage all incoming requests.



How do you supply construction arguments into a Fragment?

View the answer → Hide answer



Construction arguments for a Fragment are passed via Bundle using the `Fragment#setArgument(Bundle)` method. The passed-in Bundle can then be retrieved through the `Fragment#getArguments()` method in the appropriate Fragment lifecycle method.

It is a common mistake to pass in data through a custom constructor. Non-default constructors on a Fragment are not advisable because the Fragment may be destroyed and recreated due to a configuration change (e.g. orientation change). Using `#setArguments()/getArguments()` ensures that when the Fragment needs to be recreated, the Bundle will be appropriately serialized/deserialized so that construction data is restored.



What is ANR, and why does it happen?

[View the answer](#) → [Hide answer](#)



‘ANR’ in Android is ‘Application Not Responding.’ It means when the user is interacting with the activity, and the activity is in the `onResume()` method, a dialog appears displaying “application not responding.”

It happens because we start a heavy and long running task like downloading data in the main UI thread. The solution of the problem is to start your heavy tasks in the background using Async Task class.



Which method is called only once in a fragment life cycle?

[View the answer](#) → [Hide answer](#)



`onAttached()`



Is it possible to create an activity in Android without a user interface ?

[View the answer](#) → [Hide answer](#)



Yes, an activity can be created without any user interface. These activities are treated as abstract activities.



What is a broadcast receiver?

[View the answer](#) → [Hide answer](#)



The broadcast receiver communicates with the operation system messages such as “check whether an internet connection is available,” what the battery level should be, etc.

* There is more to interviewing than tricky technical questions, so these are intended merely as a guide. Not every “A” candidate worth hiring will be able to answer them all, nor does answering them all guarantee an “A” candidate. At the end of the day, [hiring remains an art, a science — and a lot of work.](#)
Submit an interview question

Submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.

Name
Email
Enter your question here
Enter your answer here
All fields are required
<input type="checkbox"/> I agree with the Terms and Conditions of Toptal, LLC's Privacy Policy .
<input type="button" value="Submit a Question"/>

Thanks for submitting your question.

Our editorial staff will review it shortly. Please note that submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.

Looking for Android experts? Check out Toptal's [Android developers](#).

[View full profile »](#)

[Pablo Pera](#)

United States

Pablo has taken over 10 mobile apps from concept to millions of users as the lead engineer of multiple teams in various companies, including his own. Before focusing full-time on Android apps, he worked for Google and CERN (home of the LHC particle accelerator) as well as various tech startups in NY. He has invaluable experience in all areas related to Android apps from design and coding to acquisition and growth.

[AndroidJavaGitJavaScriptGitHub+4 more](#)

[Hire Pablo](#)

[View full profile »](#)

[Oleksii Masnyi](#)

Sweden

Alex started coding on Soviet computers without hard drives. He has never lost his enthusiasm for developing great things with attention to details. He has over nine years of broad professional experience with Java, including five years on Android by developing software for high-profile companies like Samsung and Sony. He brings a passion for perfect user experience and UI into every Android project.

[AndroidJavaSQLAndroid SDKSQLiteGit](#)

[Hire Oleksii](#)

[View full profile »](#)

[Antoon Groenewoud](#)

Germany

Antoon is a top programmer and designer with experience ranging from entrepreneur to freelancer to CTO. He is active within the games industry with a knack for novel, complex solutions, and the ability to quickly integrate into any team and adapt to the problem at hand be it UI/UX or back-end or anything in between.

[AndroidC++C#.NETOpenGL+5 more](#)

[Hire Antoon](#)

Toptal connects the [top 3%](#) of freelance talent all over the world.

Join the Toptal community.

[Hire a developer](#)

or

[Apply as a developer](#)

Highest In-Demand Talent

- [iOS Developers](#)
- [Front-End Developers](#)
- [UX Designers](#)
- [UI Designers](#)
- [Financial Modeling Consultants](#)
- [Interim CFOs](#)
- [Digital Project Managers](#)

About

- [Top 3%](#)
- [Clients](#)
- [Freelance Developers](#)
- [Freelance Designers](#)
- [Freelance Finance Experts](#)
- [Freelance Project Managers](#)
- [Freelance Product Managers](#)
- [About Us](#)

Contact

- [Contact Us](#)
- [Press Center](#)
- [Careers](#)
- [FAQ](#)

Social





Hire the top 3% of freelance talent™

- © Copyright 2010 - 2019 Toptal, LLC
- [Privacy Policy](#)
- [Website Terms](#)

By continuing to use this site you agree to our [Cookie Policy](#).
Got it

Find a world-class Android developer for your team. [Hire Toptal's Android developers](#)