**Java**

Java Tutorial or Core Java Tutorial or Java Programming Tutorial is a widely used robust technology. Let's start learning of java from basic questions like what is java tutorial, core java, where it is used, what type of applications are created in java and why use java.

**What is Java**

Java is a **programming language** and a **platform**.

Java is a high level, robust, secured and object-oriented programming language.

**Platform**: Any hardware or software environment in which a program runs, is known as a platform. Since Java has its own runtime environment (JRE) and API, it is called platform.

**Where it is used?**

According to Sun, 3 billion devices run java. There are many devices where java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus etc.
2. Web Applications such as irctc.co.in, javatpoint.com etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games etc.

**Types of Java Applications**

There are mainly 4 type of applications that can be created using java programming:

**1) Standalone Application**

It is also known as desktop application or window-based application. An application that we need to install on every machine such as media player, antivirus etc. AWT and Swing are used in java for creating standalone applications.

**2) Web Application**

An application that runs on the server side and creates dynamic page, is called web application. Currently, servlet, jsp, struts, jsf etc. technologies are used for creating web applications in java.

**3) Enterprise Application**

An application that is distributed in nature, such as banking applications etc. It has the advantage of high level security, load balancing and clustering. In java, EJB is used for creating enterprise applications.

**4) Mobile Application**

An application that is created for mobile devices. Currently Android and Java ME are used for creating mobile applications.

# Features of Java

There is given many features of java. They are also known as java buzzwords. The Java Features given below are simple and easy to understand.

Java Features

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Dynamic
9. Interpreted
10. High Performance
11. Multithreaded
12. Distributed

### Simple

|  |
| --- |
| According to Sun, Java language is simple because: |
| syntax is based on C++ (so easier for programmers to learn it after C++). |
| removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc. |
| No need to remove unreferenced objects because there is Automatic Garbage Collection in java. |

### Object-oriented

|  |
| --- |
| Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behaviour. |
| Object-oriented programming(OOPs) is a methodology that simplify software development and maintenance by providing some rules. |
| Basic concepts of OOPs are: |
| 1. Object 2. Class 3. Inheritance 4. Polymorphism 5. Abstraction 6. Encapsulation |

### Platform Independent

java is platform independent

A platform is the hardware or software environment in which a program runs.

There are two types of platforms software-based and hardware-based. Java provides software-based platform.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, Mac/OS etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere(WORA).

### Secured

Java is secured because:

* **No explicit pointer**
* **Java Programs run inside virtual machine sandbox**

how java is secured

* **Classloader:** adds security by separating the package for the classes of the local file system from those that are imported from network sources.
* **Bytecode Verifier:** checks the code fragments for illegal code that can violate access right to objects.
* **Security Manager:** determines what resources a class can access such as reading and writing to the local disk.

These security are provided by java language. Some security can also be provided by application developer through SSL, JAAS, Cryptography etc.

### Robust

Robust simply means strong. Java uses strong memory management. There are lack of pointers that avoids security problem. There is automatic garbage collection in java. There is exception handling and type checking mechanism in java. All these points makes java robust.

### Architecture-neutral

There is no implementation dependent features e.g. size of primitive types is fixed.

In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. But in java, it occupies 4 bytes of memory for both 32 and 64 bit architectures.

### Portable

We may carry the java bytecode to any platform.

### High-performance

|  |
| --- |
| Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++) |

### Distributed

|  |
| --- |
| We can create distributed applications in java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet. |

### Multi-threaded

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications etc.

## Definition - What does Java Development Kit (JDK) mean?

The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development.

People new to Java may be confused about whether to use the JRE or the JDK. To run Java applications and applets, simply download the JRE. However, to develop Java applications and applets as well as run them, the JDK is needed.

Java developers are initially presented with two JDK tools, java and javac. Both are run from the command prompt. Java source files are simple text files saved with an extension of .java. After writing and saving Java source code, the javac compiler is invoked to create .class files. Once the .class files are created, the 'java' command can be used to run the java program.

For developers who wish to work in an integrated development environment (IDE), a JDK bundled with Netbeans can be downloaded from the Oracle website. Such IDEs speed up the development process by introducing point-and-click and drag-and-drop features for creating an application.

There are different JDKs for various platforms. The supported platforms include Windows, Linux and Solaris. Mac users need a different software development kit, which includes adaptations of some tools found in the JDK

**Android**

Android is an open source and Linux-based **Operating System** for mobile devices such as smartphones and tablet computers. Android was developed by the *Open Handset Alliance*, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 **Jelly Bean**. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance.

The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

**Why Android ?**



**Features of Android**

Android is a powerful operating system competing with Apple 4GS and supports great features. Few of them are listed below −

|  |  |
| --- | --- |
| **Sr.No.** | **Feature & Description** |
| 1 | **Beautiful UI**  Android OS basic screen provides a beautiful and intuitive user interface. |
| 2 | **Connectivity**  GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX. |
| 3 | **Storage**  SQLite, a lightweight relational database, is used for data storage purposes. |
| 4 | **Media support**  H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP. |
| 5 | **Messaging**  SMS and MMS |
| 6 | **Web browser**  Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3. |
| 7 | **Multi-touch**  Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. |
| 8 | **Multi-tasking**  User can jump from one task to another and same time various application can run simultaneously. |
| 9 | **Resizable widgets**  Widgets are resizable, so users can expand them to show more content or shrink them to save space. |
| 10 | **Multi-Language**  Supports single direction and bi-directional text. |
| 11 | **GCM**  Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution. |
| 12 | **Wi-Fi Direct**  A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection. |
| 13 | **Android Beam**  A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together. |

**Android Applications**

Android applications are usually developed in the Java language using the Android Software Development Kit.

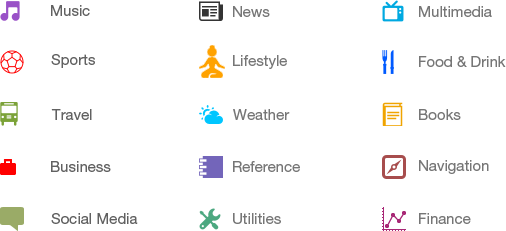
Once developed, Android applications can be packaged easily and sold out either through a store such as **Google Play**, **SlideME**, **Opera Mobile Store**, **Mobango**, **F-droid** and the **Amazon Appstore**.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide.

This tutorial has been written with an aim to teach you how to develop and package Android application. We will start from environment setup for Android application programming and then drill down to look into various aspects of Android applications.

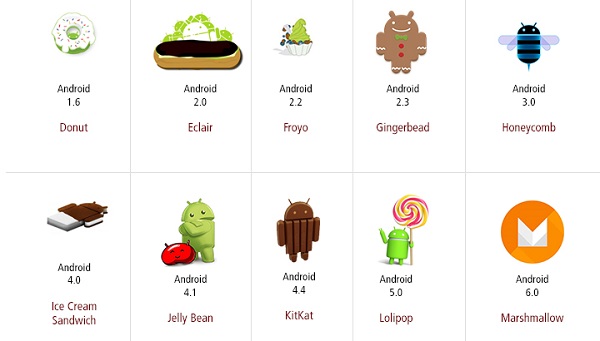
**Categories of Android applications**

There are many android applications in the market. The top categories are −



**History of Android**

The code names of android ranges from A to N currently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwitch, Jelly Bean, KitKat, Lollipop and Marshmallow. Let's understand the android history in a sequence.



**What is API level?**

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.

|  |  |  |  |
| --- | --- | --- | --- |
| **Platform Version** | **API Level** | **VERSION\_CODE** |  |
| Android 6.0 | 23 | MARSHMALLOW |  |
| Android 5.1 | 22 | LOLLIPOP\_MR1 |  |
| Android 5.0 | 21 | LOLLIPOP |  |
| Android 4.4W | 20 | KITKAT\_WATCH | KitKat for Wearables Only |
| Android 4.4 | 19 | KITKAT |  |
| Android 4.3 | 18 | JELLY\_BEAN\_MR2 |  |
| Android 4.2, 4.2.2 | 17 | JELLY\_BEAN\_MR1 |  |
| Android 4.1, 4.1.1 | 16 | JELLY\_BEAN |  |
| Android 4.0.3, 4.0.4 | 15 | ICE\_CREAM\_SANDWICH\_MR1 |  |
| Android 4.0, 4.0.1, 4.0.2 | 14 | ICE\_CREAM\_SANDWICH |  |
| Android 3.2 | 13 | HONEYCOMB\_MR2 |  |
| Android 3.1.x | 12 | HONEYCOMB\_MR1 |  |
| Android 3.0.x | 11 | HONEYCOMB |  |
| Android 2.3.4  Android 2.3.3 | 10 | GINGERBREAD\_MR1 |  |
| Android 2.3.2  Android 2.3.1  Android 2.3 | 9 | GINGERBREAD |  |
| Android 2.2.x | 8 | FROYO |  |
| Android 2.1.x | 7 | ECLAIR\_MR1 |  |
| Android 2.0.1 | 6 | ECLAIR\_0\_1 |  |
| Android 2.0 | 5 | ECLAIR |  |
| Android 1.6 | 4 | DONUT |  |
| Android 1.5 | 3 | CUPCAKE |  |
| Android 1.1 | 2 | BASE\_1\_1 |  |
| Android 1.0 | 1 | BASE |  |

# Android - Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.



**Linux kernel**

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

**Libraries**

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

**Android Libraries**

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows −

* **android.app** − Provides access to the application model and is the cornerstone of all Android applications.
* **android.content** − Facilitates content access, publishing and messaging between applications and application components.
* **android.database** − Used to access data published by content providers and includes SQLite database management classes.
* **android.opengl** − A Java interface to the OpenGL ES 3D graphics rendering API.
* **android.os** − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
* **android.text** − Used to render and manipulate text on a device display.
* **android.view** − The fundamental building blocks of application user interfaces.
* **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
* **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications.

Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

**Android Runtime**

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

**Application Framework**

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services −

* **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
* **Content Providers** − Allows applications to publish and share data with other applications.
* **Resource Manager** − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
* **Notifications Manager** − Allows applications to display alerts and notifications to the user.
* **View System** − An extensible set of views used to create application user interfaces.

**Applications**

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

**SQLite Database**

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database [file format](https://www.sqlite.org/fileformat2.html) is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between [big-endian](http://en.wikipedia.org/wiki/Endianness) and [little-endian](http://en.wikipedia.org/wiki/Endianness) architectures. These features make SQLite a popular choice as an [Application File Format](https://www.sqlite.org/appfileformat.html). Think of SQLite not as a replacement for [Oracle](http://www.oracle.com/database/index.html) but as a replacement for [fopen()](http://man.he.net/man3/fopen)

SQLite is a compact library. With all features enabled, the [library size](https://www.sqlite.org/footprint.html) can be less than 500KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function inlining and loop unrolling can cause the object code to be much larger.) If optional features are omitted, the size of the SQLite library can be reduced below 300KiB. SQLite can also be made to run in minimal stack space (4KiB) and very little heap (100KiB), making SQLite a popular database engine choice on memory constrained gadgets such as cellphones, PDAs, and MP3 players. There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments.

SQLite is [very carefully tested](https://www.sqlite.org/testing.html) prior to every release and has a reputation for being very reliable. Most of the SQLite source code is devoted purely to testing and verification. An automated test suite runs millions and millions of test cases involving hundreds of millions of individual SQL statements and achieves [100% branch test coverage](https://www.sqlite.org/testing.html#coverage). SQLite responds gracefully to memory allocation failures and disk I/O errors. Transactions are [ACID](http://en.wikipedia.org/wiki/ACID) even if interrupted by system crashes or power failures. All of this is verified by the automated tests using special test harnesses which simulate system failures. Of course, even with all this testing, there are still bugs. But unlike some similar projects (especially commercial competitors) SQLite is open and honest about all bugs and provides [bugs lists](http://www.sqlite.org/src/rptview?rn=1) and minute-by-minute [chronologies](http://www.sqlite.org/src/timeline) of code changes.

**SQLite in Android**

SQLite is embedded into every Android device. Using an SQLite database in Android does not require a setup procedure or administration of the database.

You only have to define the SQL statements for creating and updating the database. Afterwards the database is automatically managed for you by the Android platform.

Access to an SQLite database involves accessing the file system. This can be slow. Therefore it is recommended to perform database operations asynchronously.

If your application creates a database, this database is by default saved in the directory DATA/data/APP\_NAME/databases/FILENAME.

The parts of the above directory are constructed based on the following rules. DATA is the path which the Environment.getDataDirectory() method returns. APP\_NAME is your application name. FILENAME is the name you specify in your application code for the database.