Core Java

**Introduction**

Why java ?

**Easy to learn:**

Learning anything new isn’t typically easy. Java is a robust language and it would be difficult to get productive with it in a short span of time, especially if you have a steep learning curve. Many would be surprised to see once you are familiar with the program and its initial hurdles it is pretty easy to write a program in Java.

Java is the most human-readable language. It uses syntax similar to the English language with minimal symbols making it easier to read and learn quickly. This becomes one of the top reasons for learning Java or considering it as the best programming language. Especially if you are coming from C, C++ background you will find Java quite similar to those in terms of its syntax. Professionals who are experts in Oracle, SCRUM, and Sun can easily combine their knowledge with Java to develop web or mobile applications. Also, Java being one of the oldest programming languages, certainly has no shortage of resources including websites, tutorials, books, and classes available to help learn the language.

**Higher cross-functionality:**

Java offers high cross-functionality and portability as programs written in one platform can run across desktops, mobiles, embedded systems. Java is used in a lot of different places. Its write once runs anywhere philosophy allows it to be used in many ways on a wide variety of different devices. It can easily be moved from one computer to another. Most Java applications are built in windows and can run on Linux.

Companies such as Airbnb, Uber, and eBay use Java, for at least part of their web sites and applications. If you want to develop applications for Android-based smartphones and tablets, Java works there too. Java is a big player in the internet of things and connected device market.

Behind the scene, Java is controlling things like ATMs, industrial robots, medical devices, and with the rapid growth of connected home devices from light bulbs to garage door openers you find the Java-powered logo. They are even in the entire lines of home devices that are connected to the internet and leverage Java programming.

**Object-oriented language:**

Java is an object-oriented language. It’s one of the few 100 percents object-oriented language, which makes developing OOP applications very easy. That’s another reason that made it a popular language. Because you cannot write a simple program in Java without it being object-oriented. Once you have the knowledge of basic OOP concepts, you can use them with Java. Unlike other programming languages like C++, JavaScript, and Python, OOP is optional whereas in Java it’s really baked into the language as an essential part of Java. Encapsulation, Abstraction, Inheritance, and Polymorphism, are some of the best practices and design patterns in its library.

**Awesome tools:**

Eclipse and Netbeans are the most popular IDEs in the development environment used by software developers. It played a huge role to make Java one of the best programming languages. It also has other IDEs like Notepad++, DOS editor and other text editors that all have excellent support for Java built-in.

There is a vast assortment of libraries and frameworks available in Java that can make you more efficient as a program as some the task you are going to perform while developing an application are already solved with the existing library. Other helpful tools include Maven, ANT, Jenkins, Decompilers, Visual VM and JConsole that not only provide help in code completion but has powerful debugging capabilities, which is very essential in real-time web development. It’s easy to integrate Java into a desktop, server-side web application or mobile application that makes Java development much faster, easier, and fluent. In desktop, Java is widely used to developing desktop applications and Java2 enterprise addition is very popular in server-side web programming. In mobile applications, Java is the most popular programming language for the Android operating system.

**Mobile app development:**

Java is considered to be very useful for scripting games and mobile applications. Java can be used in order to successfully facilitate mobile development, specifically beneficial within the [Android app development](https://www.aalphaindia.com/hire-developers/hire-android-app-developers) market. Eclipse is a very popular IDE used for developing Android applications. With the introduction of Android Studio IDE, which is specifically designed for developing Android applications, you can install professional tools that you will need for developing mobile applications.

utomated testing is also included with Junit along with the test-driven development. Although there is another language called Kotlin developed by JetBrains. It’s basically Java’s answer to Swift for iOS instead of Object-C. [Kotlin](https://www.aalpha.net/blog/heres-why-kotlin-holds-the-future-for-android-app-development/) does the same thing for [android app development](https://www.aalpha.net/services/mobile-app-development/android-app-india/). It’s faster, easier, and lighter, than Java. But, Android relies heavily on Java and uses Java as a base language for developing mobile apps. Many of the standard Java libraries like graphics, data structure, math, networking, as well as some special libraries are already included in Android SDK. These existing libraries help developers build awesome Android application.

**Open source:**

Java has been very lucky in having a thriving community, otherwise, it wouldn’t survive. It is estimated there are as many as 10 million Java developers around the world. Java runs on a wide variety of devices from sports cars to medical devices and that means a wide variety of industries employed [Java developers](https://www.aalphaindia.com/hire-developers/hire-java-developers). That leads to a broad and diverse community with many different ways to get involved. They held conferences and meetups for Java users across the globe. Community is a huge factor when learning any language. It has a lot of forums and several Java user groups where Java community is available to answer your questions. Since it’s a mature language there are many resources available and many questions have been asked already with great answers provided. Whether you are using Java to build your website or using to power the internet of things project there is a community for you.

And on top of that, it’s free. Cost is the biggest factor for anyone who wants to learn a programming language or for an organization to use that technology. With Java, you do not have to pay anything to create your Java application.

**Legacy enterprise:**

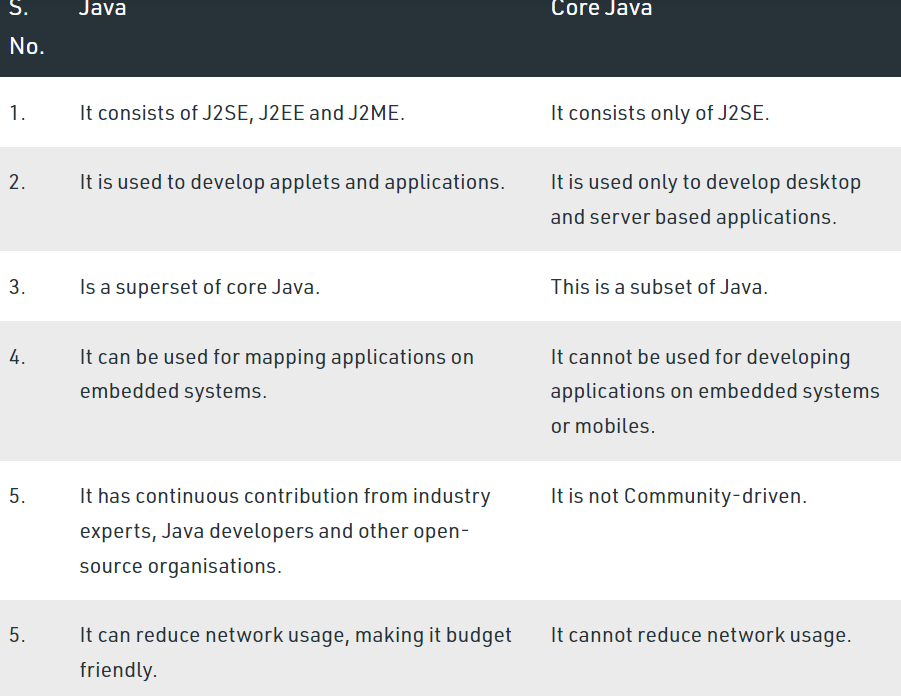
Java is something relegated to big enterprises. It is mostly used in legacy enterprise application development. Java was a new hot language back in the late ’90s and all the big corporations adopted Java. Because these big companies got involved with Java, it was sort of designed via consensus with huge corporations. It was super complex and a bit slower back then but has been simplified later to some extent. Being highly popular at the enterprise, Java has a large active user community and support available. Java is a very good choice for you if you are going to be working with very large corporation’s projects that may last for six months or year with a big team.

**Multithreading:**

Thread is the smallest unit of processing in any programming. Java has the capability for a program to perform several tasks simultaneously within a program with its multithreading feature. Multithreading allows you to maximize the utilization of CPU time.

Java has very effective memory management. Switching between these threads take very little time as they share the same area of memory. Also, they operate independent of each other, so it does not affect the other if one thread faces exception. This is especially useful for heavy applications like games and animation.

**Java vs Core java**

****

**Tokens in java**

The Java compiler breaks the line of code into text (words) is called **Java tokens**. These are the smallest element of the [Java program](https://www.javatpoint.com/java-programs). The Java compiler identified these words as tokens. These tokens are separated by the delimiters. It is useful for compilers to detect errors. Remember that the delimiters are not part of the Java tokens.

<https://www.javatpoint.com/java-tokens>

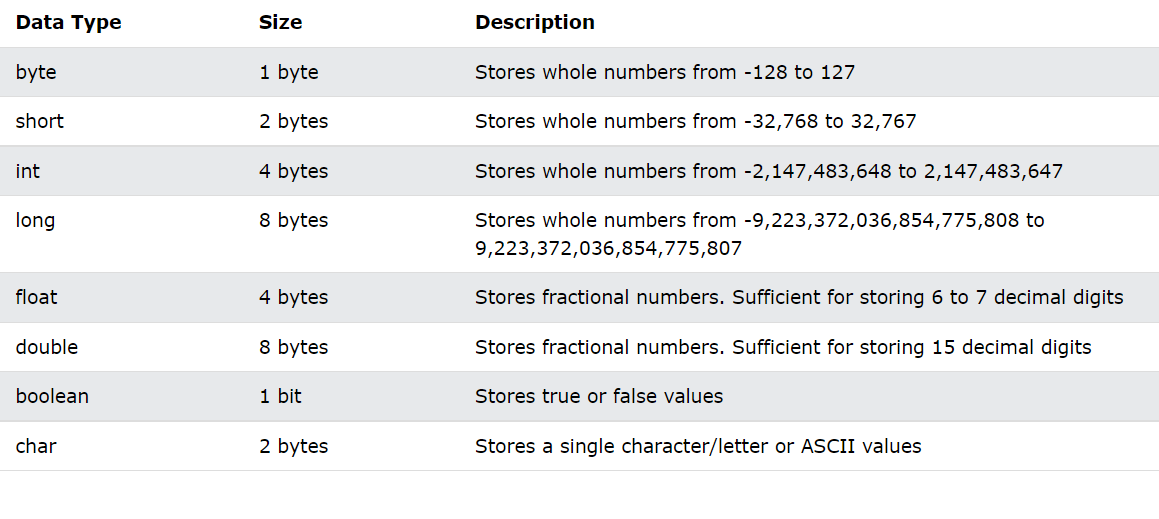
**Java Statements**

<https://javabeginnerstutorial.com/core-java-tutorial/java-statements-tutorial-for-beginners/>

**Data types in java**

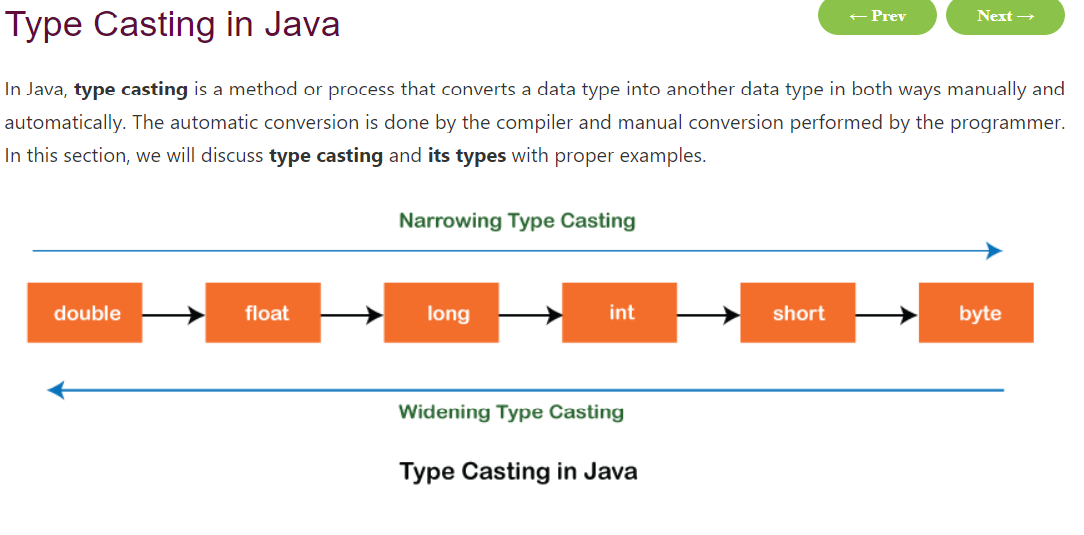
Data types are divided into two groups:

* Primitive data types includes byte, short, int, long, float, double, boolean and char
* Non-primitive data types - such as [String](https://www.w3schools.com/java/java_strings.asp), [Arrays](https://www.w3schools.com/java/java_arrays.asp) and [Classes](https://www.w3schools.com/java/java_classes.asp)



<https://www.w3schools.com/java/java_data_types.asp>

**Type Casting in java**



**Oops (Object Oriented Programing )**

Java classes & Objects

* **Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.
* **Class** − A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.

The basic Object-oriented programming concepts are:

**Inheritance**

 Inheritance can be defined as the process where one (parent/super) class acquires the properties (methods and fields) of another (child/sub). With the use of inheritance, the information is made manageable in a hierarchical order.

**Polymorphism**

Polymorphism is the ability of an object to perform different actions (or, exhibit different behaviors) based on the context.

**Abstraction**

Abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

In Java, abstraction is achieved using Abstract classes and interfaces.

**Encapsulation**

Encapsulation in Java is a mechanism for wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes and can be accessed only through the methods of their current class. Therefore, it is also known as**data hiding**. To achieve encapsulation in Java −

1. Declare the variables of a class as private.
2. Provide public setter and getter methods to modify and view the variables values.

**Java Packages**

A **java package** is a group of similar types of classes, interfaces and sub-packages.

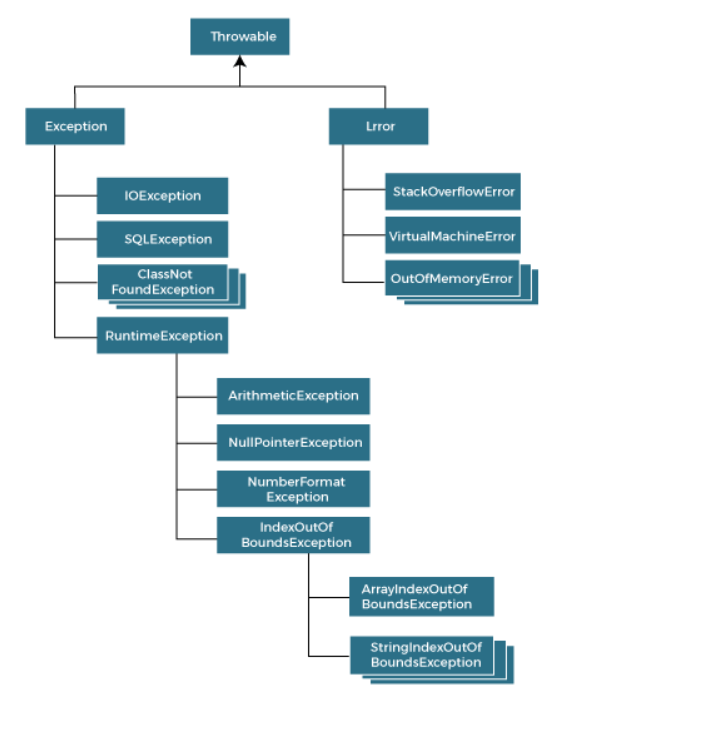
Package in java can be categorized in two form, built-in package and user-defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Here, we will have the detailed learning of creating and using user-defined packages.

**Exception Handling**

Link : <https://www.javatpoint.com/exception-handling-in-java>



**Multi-threading**

is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

Java Multithreading is mostly used in games, animation, etc.

Link : [https://www.javatpoint.com/multithreading-in-java#:~:text=Multithreading%20in%20java%20is%20a%20process%20of%20executing,and%20multithreading%2C%20both%20are%20used%20to%20achieve%20multitasking.](https://www.javatpoint.com/multithreading-in-java" \l ":~:text=Multithreading%20in%20java%20is%20a%20process%20of%20executing,and%20multithreading%2C%20both%20are%20used%20to%20achieve%20multitasking.)

**Collection Frame Work**

**What is a Java Collection Framework?**

A Java collection framework provides an architecture to store and manipulate a group of objects. A Java collection framework includes the following:

* Interfaces
* Classes
* Algorithm

Let’s learn about them in detail:

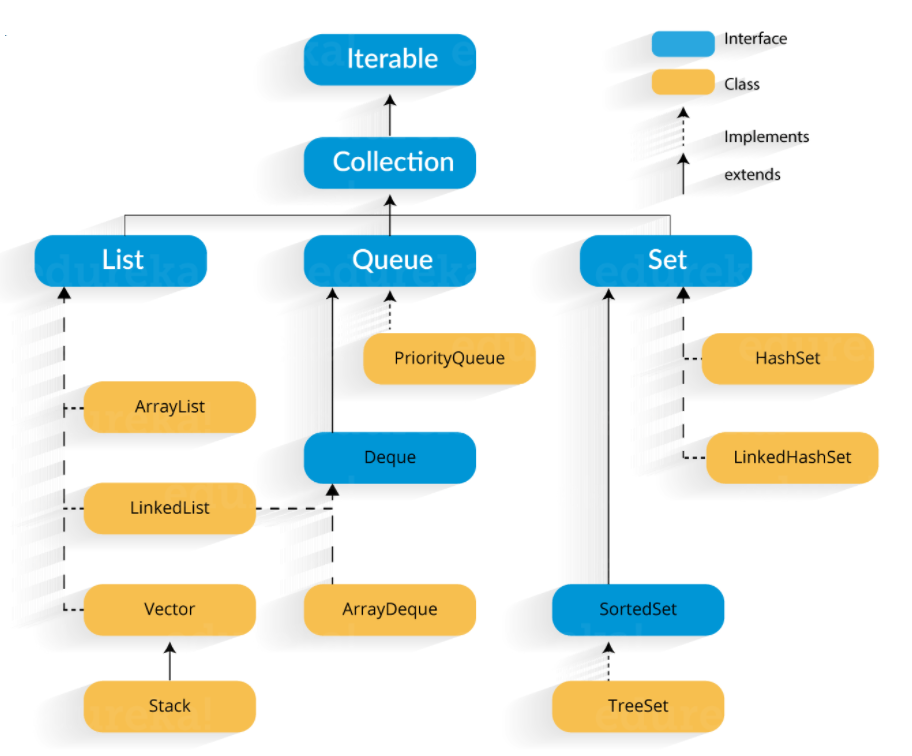
**Interfaces**: Interface in Java refers to the abstract data types. They allow Java collections to be manipulated independently from the details of their representation. Also, they form a hierarchy in object-oriented programming languages.

**Classes:**Classes in Java are the implementation of the collection interface. It basically refers to the data structures that are used again and again.

**Algorithm:** Algorithm refers to the methods which are used to perform operations such as searching and sorting, on objects that implement collection interfaces. Algorithms are polymorphic in nature as the same method can be used to take many forms or you can say perform different implementations of the Java collection interface.

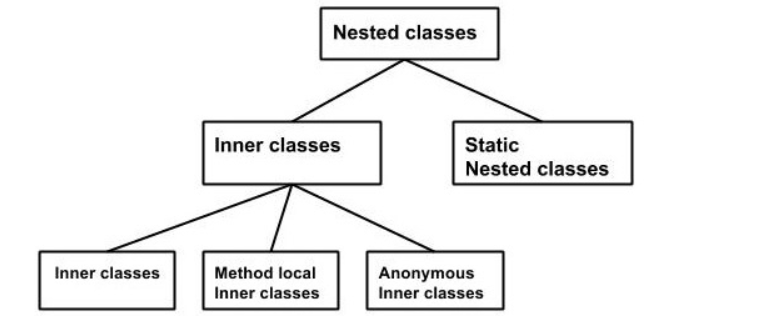
So why do you think we need Java collections? The Java collection framework provides the developers to access prepackaged data structures as well as algorithms to manipulate data. Next, let us move to the Java collections framework hierarchy and see where these interfaces and classes resides.

<tutorialspoint.com/java/java_collections.htm>



**Inner Classes in Java**

<https://www.tutorialspoint.com/java/java_innerclasses.htm>



**Swing (JFC)**

[https://www.javatpoint.com/java-swing#:~:text=Java%20Swing%20tutorial%20is%20a%20part%20of%20Java,AWT%2C%20Java%20Swing%20provides%20platform-independent%20and%20lightweight%20components.](https://www.javatpoint.com/java-swing%23:~:text=Java%20Swing%20tutorial%20is%20a%20part%20of%20Java,AWT%2C%20Java%20Swing%20provides%20platform-independent%20and%20lightweight%20components.)

**Java AWT**

Abstract Window Toolkit) is *an API to develop Graphical User Interface (GUI) or windows-based applications* in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

<https://www.javatpoint.com/java-awt>

