Java ClassLoader loads a java class file into java virtual machine. It is as simple as that. It is not a huge complicated concept to learn and every java developer must know about the java class loaders and how it works.

Like [NullPointerException](https://javapapers.com/core-java/nullpointerexception-null-bad-good-and-ugly/" \o "Java null and NullPointerException), one exception that is very popular is ClassNotFoundException. At least in your beginner stage you might have got umpteen number of ClassNotFoundException. Java class loader is the culprit that is causing this [exception](https://javapapers.com/core-java/java-exception/explain-type-of-exceptions-or-checked-vs-unchecked-exceptions-in-java/).

Types (Hierarchy) of Java Class Loaders

Java class loaders can be broadly classified into below categories:

* Bootstrap Class Loader  
  Bootstrap class loader loads java’s core classes like java.lang, java.util etc. These are classes that are part of java runtime environment. Bootstrap class loader is native implementation and so they may differ across different JVMs.
* Extensions Class Loader  
  JAVA\_HOME/jre/lib/ext contains jar packages that are extensions of standard core java classes. Extensions class loader loads classes from this ext folder. Using the system environment propery java.ext.dirs you can add ‘ext’ folders and jar files to be loaded using extensions class loader.
* System Class Loader  
  Java classes that are available in the java classpath are loaded using System class loader.

You can see more class loaders like java.net.URLClassLoader, java.security.SecureClassLoader etc. Those are all extended from java.lang.ClassLoader

These class loaders have a hierarchical relationship among them. Class loader can load classes from one level above its hierarchy. First level is bootstrap class loader, second level is extensions class loader and third level is system class loader.

## Class Self Reference

When a java source file is compiled to a [binary class](https://javapapers.com/core-java/java-binary/), compiler inserts a field into java class file. It is a public [static](https://javapapers.com/core-java/explain-the-java-static-modifier/) [final field](https://javapapers.com/core-java/explain-the-final-keyword-in-java/) named ‘class’ of type java.lang.Class

So for all java classes you can access it as java.lang.Class classObj = ClassName.class;  
  
Significance of this Class object is it contains a method getClassLoader() which returns the class loader for the class. It will return null it it was loaded by bootstrap class loader.

## How a Java Class Loader Works?

When a class name is given, class loader first locates the class and then reads a class file of that name from the native file system. Therefore this loading process is platform dependent.

By default java.lang.ClassLoader is registered as a class loader that is capable of loading classes in parallel. But the subclasses needs to register as parallel or not at the time of instantiation.

Classes can also be loaded from network, constructed on runtime and loaded. ClassLoader class has a method name defineClass which takes input as byte array and loads a class.

### Class Loader Parent

All class loaders except bootstrap class loader has a parent class loader. This parent is not as in parent-child [relationship](https://javapapers.com/oops/association-aggregation-composition-abstraction-generalization-realization-dependency/) of inheritance. Every class loader instance is associated with a parent class loader.

When a class loader is entrusted with the responsibility of loading a class, as a first step it delegates this work to the associated parent class loader. Then this parent class loader gets the instruction and sequentially it delegates the call to its parent class loader. In this chain of hierarchy the bootstrap class loader is at the top.

When a class loader instance is created, using its constructor the parent classloader can be associated with it.

## Class Loader Rule 1

A class is loaded only once into the JVM.

In this rule, what is “a class”? Uniqueness of a class is identified along with the ClassLoader instance that loaded this class into the [JVM](https://javapapers.com/core-java/differentiate-jvm-jre-jdk-jit/). A class is always identified using its fully qualified name (package.classname). So when a class is loaded into JVM, you have an entry as (package, classname, classloader). Therefore the same class can be loaded twice by two different ClassLoader instances.

I will be writing some more articles on custom class loaders, jar hell and internals of class loading like loading-linking.