* Install JDK and Eclipse

**https://www.youtube.com/watch?v=Md3dIQre7PQ**

* A list of the most important features of the Java language is given below.

Simple

Java is very easy to learn, and its syntax is simple, clean, and easy to understand. According to Sun Microsystem, Java language is a simple programming language because:

* Java syntax is based on C++ (so easier for programmers to learn it after C++).
* Java has removed many complicated and rarely used features, for example, explicit pointers, operator overloading, etc.
* There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

### Object-oriented

Java is an [object-oriented](https://www.javatpoint.com/java-oops-concepts) programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behaviour.

### **Platform Independent**



Java is platform independent because it is different from other languages like [C](https://www.javatpoint.com/c-programming-language-tutorial), [C++](https://www.javatpoint.com/cpp-tutorial), etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. 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 a software-based platform.

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

Runtime Environment

API(Application Programming Interface)

Java code can be executed on multiple platforms, for example, 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 best known for its security. With Java, we can develop virus-free systems. Java is secured because:

No explicit pointer

Java Programs run inside a virtual machine sandbox



* Classloader: Classloader in Java is a part of the Java Runtime Environment (JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.
* Bytecode Verifier: It checks the code fragments for illegal code that can violate access rights to objects.
* Security Manager: It determines what resources a class can access such as reading and writing to the local disk.

Java language provides these securities by default. Some security can also be provided by an application developer explicitly through SSL, JAAS, Cryptography, etc.

### Robust

The English mining of Robust is strong. Java is robust because:

* It uses strong memory management.
* There is a lack of pointers that avoids security problems.
* Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
* There are exception handling and the type checking mechanism in Java. All these points make Java robust.

### Architecture-neutral

Java is architecture neutral because there are no implementation dependent features, for example, the 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. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

### Portable

Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

### High-performance

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

### Distributed

Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.

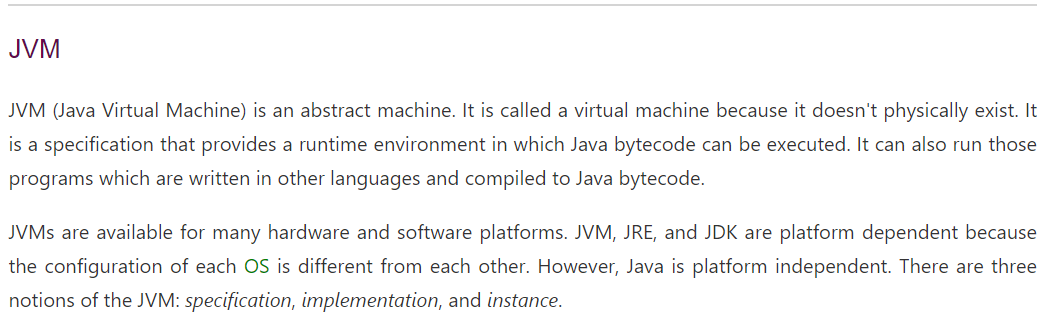
### Multi-threaded

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.

### Dynamic

Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

* JDK and JRE



Graphical user interface, text, application, email

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Diagram

Description automatically generated

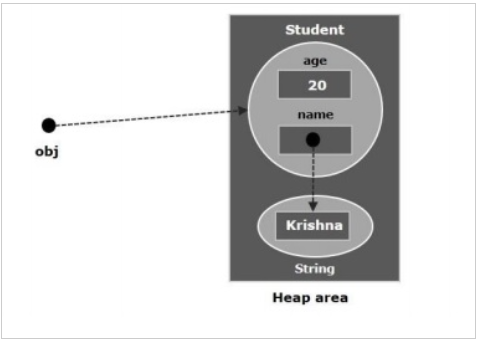
* Difference between object and reference variable

When you create an object of a class as,

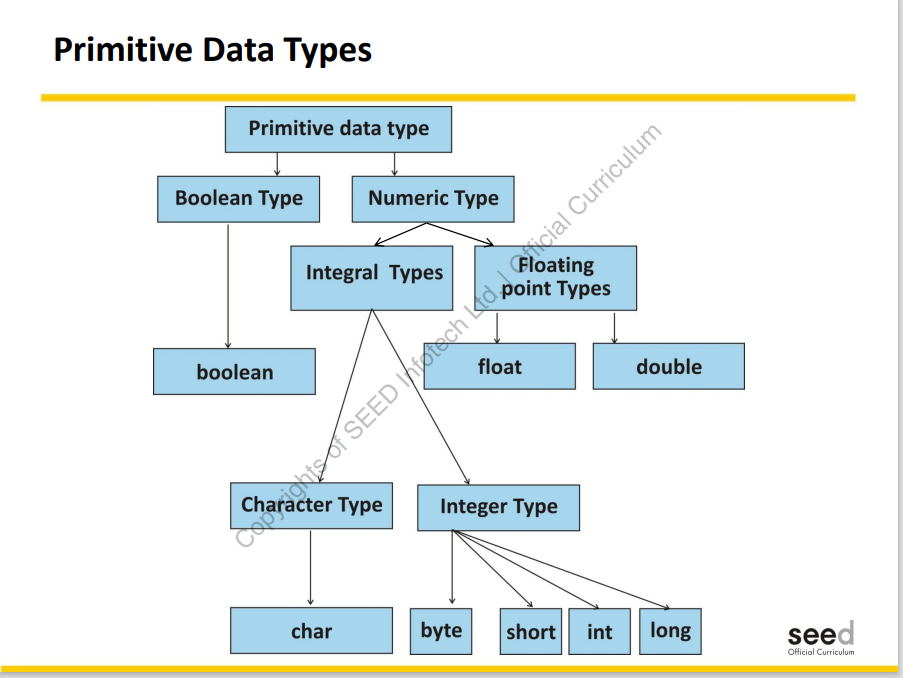
Student obj = new Student ();

Here Student obj is reference variable and using new keyword, object of class Student is created in the memory.

The objects are created in the heap area and, the reference obj just points out to the object of the student class in the heap, i.e. it just holds the memory address of the object (in the heap).

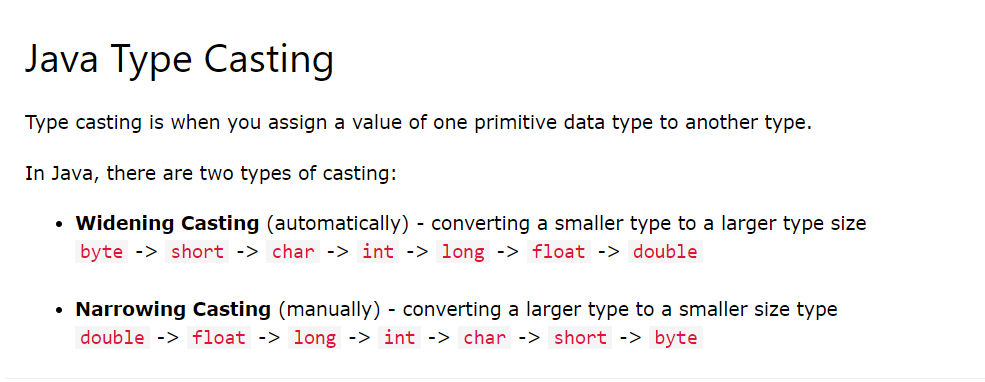


In short, object is an instance of a class and reference (variable) points out to the object created in the heap area.



Table

Description automatically generated



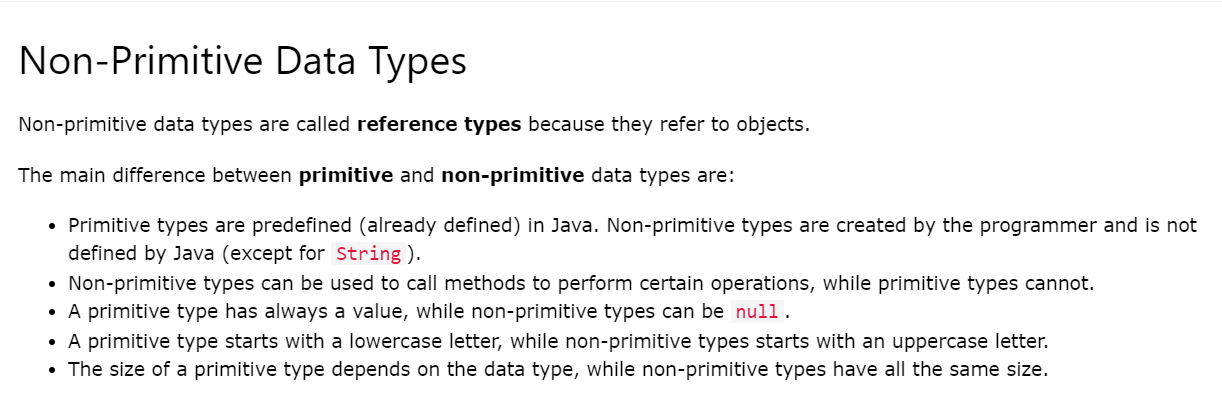


Graphical user interface, text, application, email

Description automatically generated

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)



* public static void main (String[] args)

In Java programs, the point from where the program starts its execution or simply the entry point of Java programs is the main () method. Hence, it is one of the most important methods of Java and having a proper understanding of it is very important.

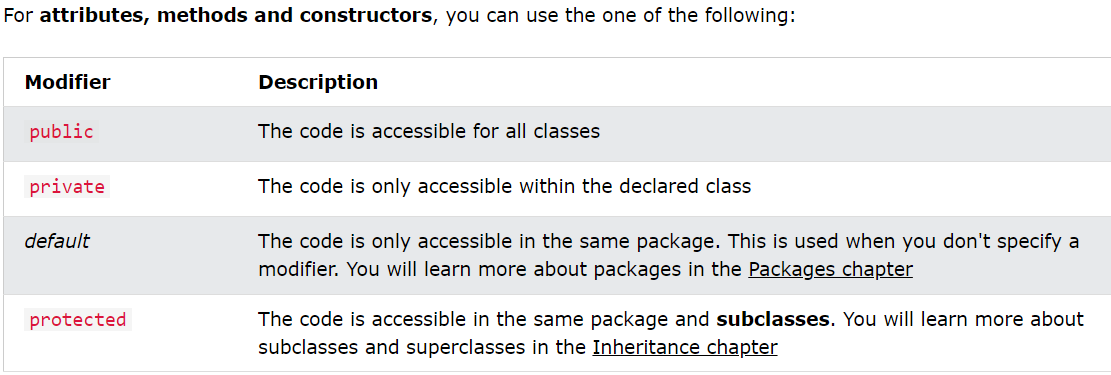
The Java compiler or JVM looks for the main method when it starts executing a Java program. The signature of the main method needs to be in a specific way for the JVM to recognize that method as its entry point.

* Access Specifiers

We divide modifiers into two groups:

* **Access Modifiers** - controls the access level
* **Non-Access Modifiers** - do not control access level, but provides other functionality

Graphical user interface, text, application

Description automatically generated

Refer link for more details,

[Java Access Modifiers (With Examples) (programiz.com)](https://www.programiz.com/java-programming/access-modifiers)

* Accessor and Mutator Methods in Java

Refer below links

[Accessors And Mutators In Java (codegym.cc)](https://codegym.cc/groups/posts/accessors-and-mutators-in-java)

[Advantages of getter and setter Over Public Fields in Java with Examples - GeeksforGeeks](https://www.geeksforgeeks.org/advantages-of-getter-and-setter-over-public-fields-in-java-with-examples/)

* Constructors

In [Java](https://www.javatpoint.com/java-tutorial), a constructor is a block of codes similar to the method. It is called when an instance of the [class](https://www.javatpoint.com/object-and-class-in-java) is created.

Every time an object is created using the new () keyword, at least one constructor is called.

It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default.

**Note:** It is called constructor because it constructs the values at the time of object creation. It is not necessary to write a constructor for a class. It is because java compiler creates a default constructor if your class doesn't have any.

There are two rules defined for the constructor.

1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. A Java constructor cannot be abstract, static, final, and synchronized

There are two types of constructors in Java:

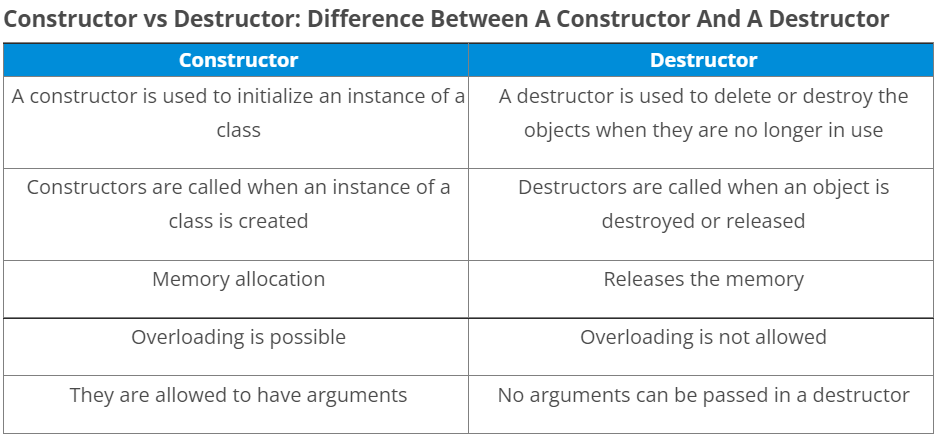
1. Default constructor (no-arg constructor)
2. Parameterized constructor

* Destructors

A destructor is a special [method](https://www.edureka.co/blog/java-methods/) that gets called automatically as soon as the life cycle of an object is finished. A destructor is called to de-allocate and free memory.

### Garbage Collector

A garbage collector is a program that runs on the [Java virtual machine](https://www.edureka.co/blog/java-virtual-machine/) to recover the memory by deleting the objects which are no longer in use or have finished their life-cycle. An object is said to be eligible for garbage collection if and only if the object is unreachable.



## Finalize () Method

## It becomes difficult for any developer to force the execution of a garbage collector, but there is an alternative to this. We can use the object.[finalize()](https://www.edureka.co/blog/final-finally-and-finalize-in-java/) method.

## It is not a destructor but is used to make sure or provide additional security to ensure the use of external resources like closing the file, etc before the program shuts down.

## The use of finalize() method is highly not recommended since it can be very unsafe and in some cases used incorrectly.

* Method Overloading

Method overloading is also known as Compile-time Polymorphism, Static Polymorphism, or Early binding in Java.

If a [class](https://www.javatpoint.com/object-and-class-in-java) has multiple methods having same name but different in parameters/data types, it is known as **Method Overloading**.

* This

This keyword refers to the current object in a method or constructor.

The most common use of this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter).