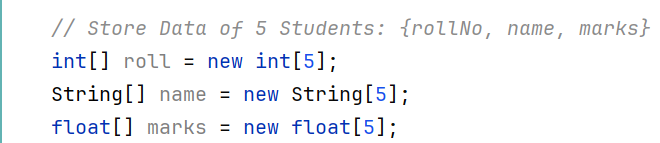
Oops Introduction

CLASS

A Class is a named group of properties and functions.



These 3 properties, if I want to combine these 3 properties into a single entity, i.e. create our own datatype, we can use class.

A screenshot of a computer program

Description automatically generated

Example:

A group of white rectangular signs with text

Description automatically generated

* A Class is a template of an object.
* Class helps us in defining a datatype.
* Class does not exist physically; it is a logical construct. It does not occupy space in memory.
* Here, CAR is a class.

OBJECT

* An object is an instance of a class.
* Objects exist physically.
* Here, all the 3 cars, Maruti, Ferrari and Audi are objects that are instance of CAR class.
* It occupies space in memory.

Three essential properties of objects:

1. State: Value from its datatype.
2. Identity: Whether one object is different from another.
3. Behavior: The behavior or operations of an object is its predefined functions. For example, a T.V. can show pictures, change channels, tune for a channel etc.

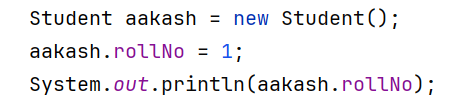
A close-up of a black background

Description automatically generated

How to access the properties of student1, student2 and student3

Using the dot(.) operator: The dot operator basically links the reference variable (object, i.e. student1, student2) with the name of instance variable.

Example:



This will print 99.

Creating Objects

A white cylinder with black text

Description automatically generated

Here, student1 is declared only and student1(reference variable) is stored in the stack memory.

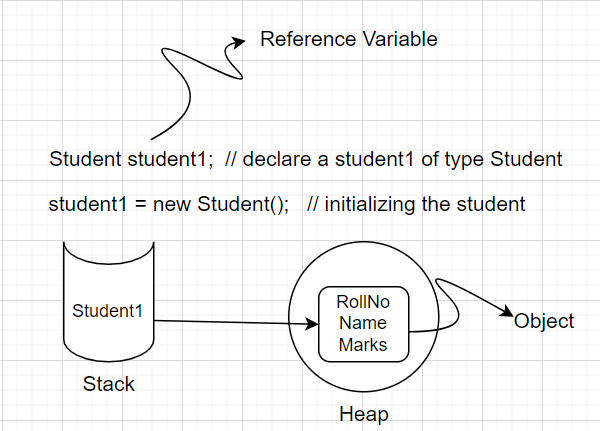
It is currently pointing to null.

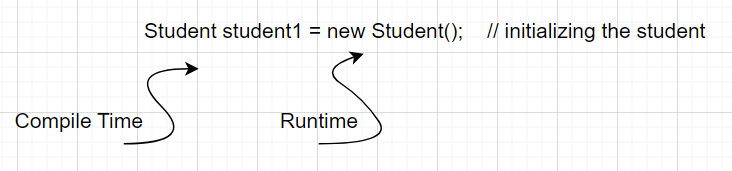
* IN JAVA, EVERYTING IS PASSED BY VALUE.

NEW Keyword

The new keyword dynamically allots (allocates memory at runtime) and returns a reference variable to it.



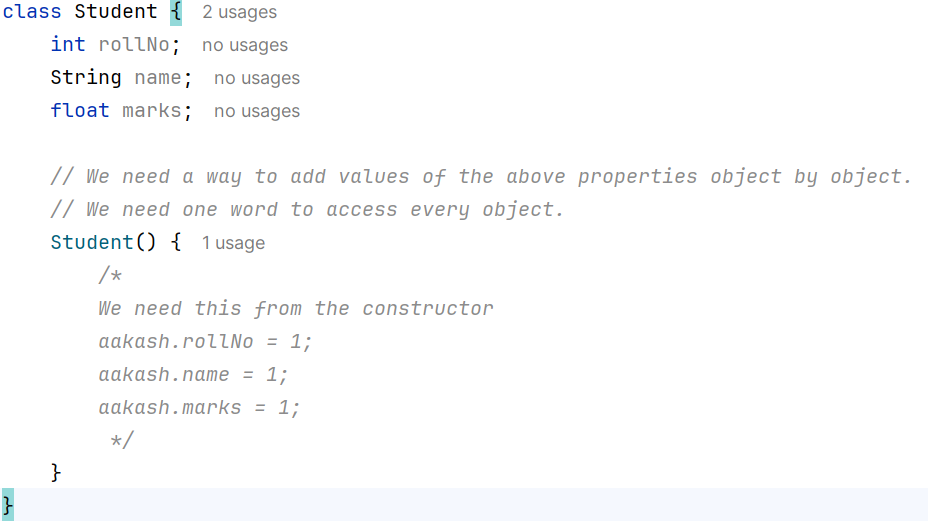




Java Constructor

A constructor is a special function in the class that defines what will happen when the object is created. It runs when the object is created, and it allocates some variables.

It has NO RETURN TYPE.



This Keyword

This keyword is used to access objects inside the class template. Internally `this` will be replaced by the reference variable.

A white background with black text

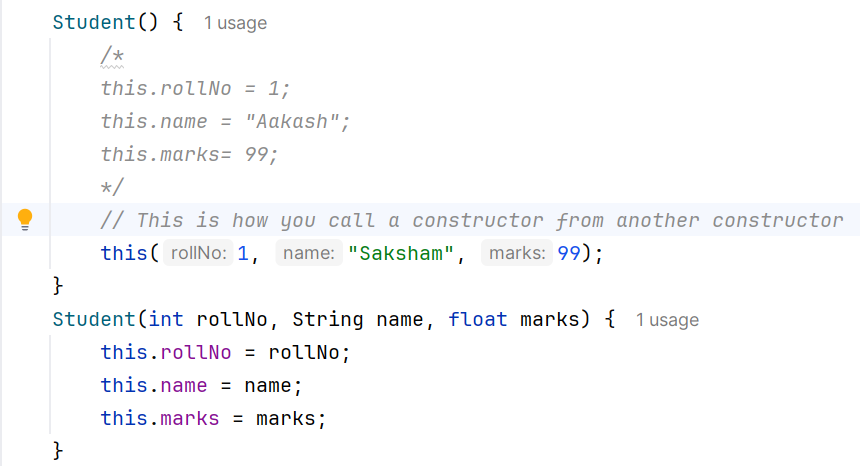
Description automatically generated

Constructor Overloading

A screenshot of a computer code

Description automatically generated

Call a constructor from another constructor!



Why don’t we use NEW KEYWORD for creating primitive datatypes?

* In JAVA, the primitive datatypes are implemented as Objects.
* We know that objects are stored in heap memory, and primitives are not objects.
* Therefore, primitives are stored in the STACK MEMORY.

Wrapper Classes

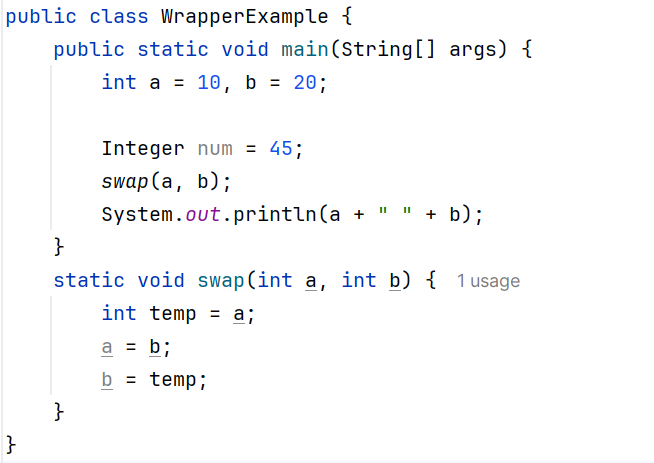
A Wrapper class in Java is a class whose object wraps or contains primitive data types. When we create an object to a wrapper class, it contains a field and, in this field, we can store primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

Need of Wrapper Classes

There are certain needs for using the Wrapper class in Java as mentioned below:

* They convert primitive data types into objects. Objects are needed if we wish to modify the arguments passed into a method (because primitive types are passed by value).
* The classes in `java.util` package handles only objects and hence wrapper classes help in this case also.
* Data structures in the Collection framework, such as Array List and Vector, store only objects (reference types) and not primitive types.
* An object is needed to support synchronization in multithreading.

Example: a and b are primitives; they are not considered as Objects while num is an Object.



Here, a and b are not swapped because everything in JAVA is passed by value. Therefore, the scope of a and b is inside the swap function only.

Final Keyword

Final is a keyword by which we can prevent the content from being modified.

They are written in UPPERCASE.

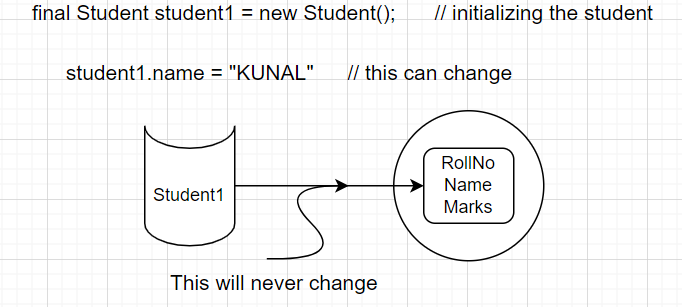
A screenshot of a computer

Description automatically generated

The final variable needs to be initialized. It cannot be left declared, because later we cannot modify or change its value.

**It guarantees that this immutability or you cannot change it, is only when the instance variables are primitive datatypes, not the reference type of object.**

**If the instance variable of reference type has FINAL, the reference to the object will never change but value of the object can change.**

****

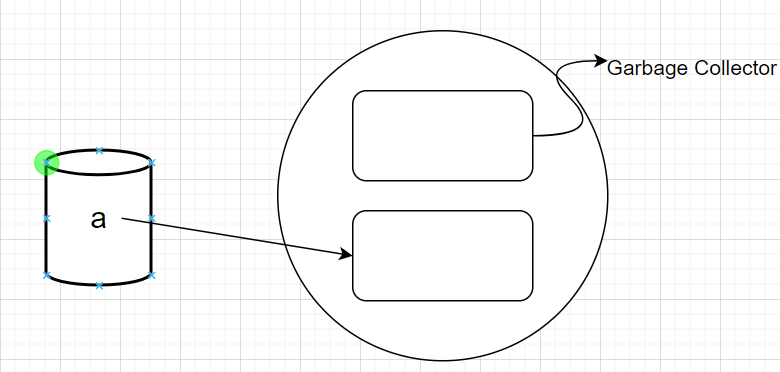
This will give error. We cannot point it to another object.

**A black and white text

Description automatically generated**

Garbage Collection

Java garbage collection is the process by which Java programs perform automatic memory management. Java programs compile to bytecode that can be run on a Java Virtual Machine, or JVM for short. When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program.



In JAVA, garbage collection is performed by JVM on its own when necessary. We cannot tell JVM manually to perform GC when we want.

But we can tell JAVA what to do when GC is performed using the **finalize** keyword.

Finalize Keyword

The finalize() method in Java is a method of the Object class used to perform cleanup activity before destroying any object. Garbage collector calls it before destroying the objects from memory.

Example:

A screenshot of a computer program

Description automatically generated

Creating numerous objects so that JVM call garbage collector and finalize() method runs.

A screenshot of a computer program

Description automatically generated