**Java**

History: Java was developed by James Gosling in 1995 , who is

known as the father of Java. Initially developed at Sun

Microsystems and later on Oracle Purchased the Sun

Microsystem in 2009.

Principles: Robust, Portable, Write Once Run Anywhere , Platform

Independent , Secured .

JVM: Java Virtual Machine (JVM) is a specification that provides

run time environment in which java bytecode can be executed.

Tasks Performed by JVM

* Loads the code.
* Verifies the code.
* Executes the code.
* Provides the runtime environment.

JRE: Java Run Time Environment(JRE) . It is the implementation

of JVM. It contains the set of libraries and other files that JVM

uses at runtime.

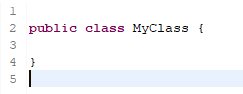
JDK: Java Development Kit. It contains JRE + Development tools.

Class: A class is the blueprint from which objects are created.

First letter of a class name is always capital and afterwards

use camel casing .

Syntax of Class:



Object: Object is the instance of class . It has states and behaviour.

Object is created through a new keyword.

Data Types: Data types specify the different sizes and values that can stored in a variable.

Types:

1. Primitive: A primitive data type specifies the size and type of

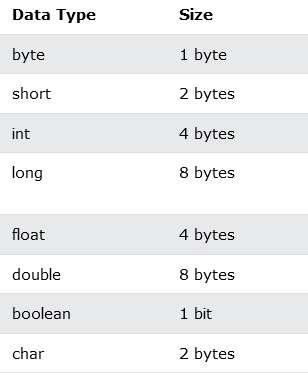
variables values and it has no additional methods.

A: Number : Integer(byte , short ,int , long)

Rational Number (float,double)

B: Character : Single line character (char).

C: Boolean: True or False.



1. Non Primitive: Non-primitive data types are called **reference types** because they refer to objects.

The main difference between **primitive** and **non-primitive** data types are:

* Primitive types are predefined (already defined) in Java. Non-primitive types are created by the programmer and is not defined by Java (except for String).
* Non-primitive types can be used to call methods to perform certain operations, while primitive types cannot.
* A primitive type has always a value, while non-primitive types can be null.
* A primitive type starts with a lowercase letter, while non-primitive types starts with an uppercase letter.
* The size of a primitive type depends on the data type, while non-primitive types have all the same size.

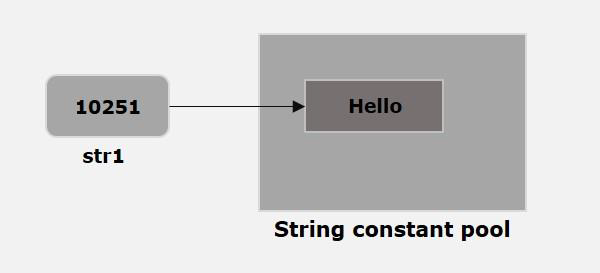
Examples of non-primitive types are [Strings](https://www.w3schools.com/java/java_strings.asp), [Arrays](https://www.w3schools.com/java/java_arrays.asp), [Classes,](https://www.w3schools.com/java/java_classes.asp) [Interface](https://www.w3schools.com/java/java_interface.asp), etc. You will learn more about these in a later chapter

Q: What is the difference between String s1 = "Hello" and String s1= new String("Hello") in java?

When you store a String as

String str1 = "Hello";

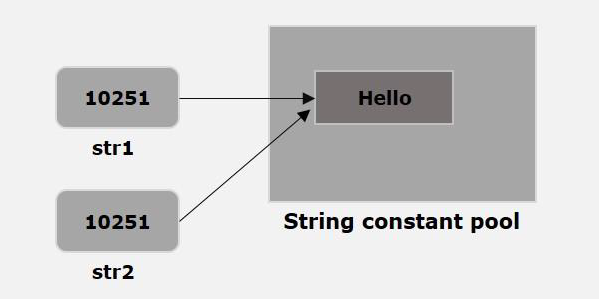
directly, then JVM creates a String object with the given value in a separate block of memory known as *String constant pool*.



And whenever we try to create another String as

String str2 = "Hello";

JVM verifies whether any String object with the same value exists in the String constant pool, if so, instead of creating a new object JVM assigns the reference of the existing object to the new variable.



And when we store String as

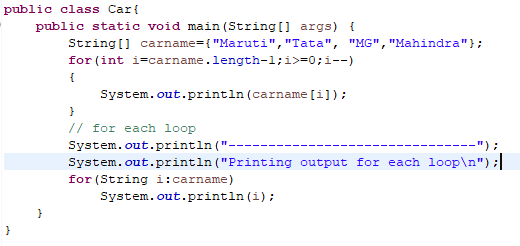
String str = new String("Hello");

using the new keyword, a new object with the given value is created irrespective of the contents of the String constant pool.

A: String: String has methods and string belongs to a class .

A string variable contains a collection of characters.

B: Arrays: Arrays are use to store multiple values of similar data type in a single variable.

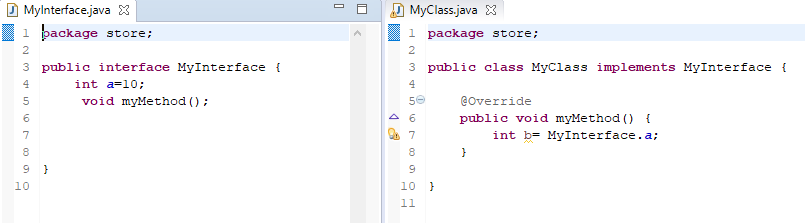


**Classes:** A [class in Java](https://www.edureka.co/blog/java-tutorial/#obj) is a blueprint which includes all your data. A class contains fields(variables) and methods to describe the behavior of an object.

**Interface:** Like a class, an *interface* can have methods and variables, but the methods declared in [*interface*](https://www.edureka.co/blog/java-collections/#interface) are by default abstract (only method signature, no body).

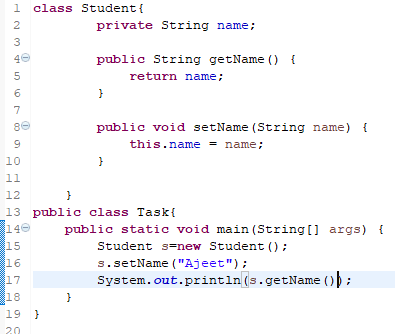
It provides the qualified template for a class to be eligible or

declared as belonging to a type.



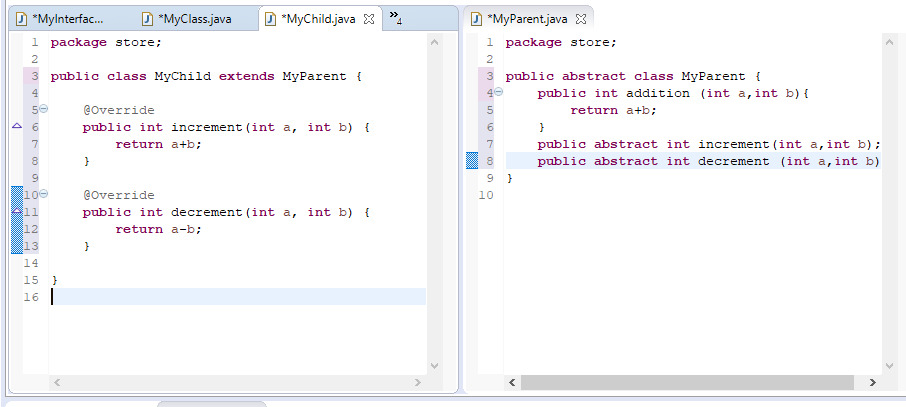
OOPS majorly based on 4 principles:

1. Encapsulation: data binding -> there is a strong bond between class and its data(variables).



1. Abstraction: operation/functional hiding-> We will be presenting the user which is required for him.

A class which is declared with the abstract keyword is known as abstract class. It have abstract and non abstract methods.It can have constructors and static methods also.



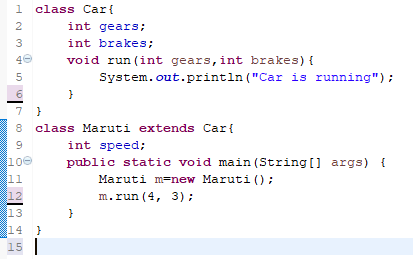
ATM Machine

1. Inheritance: Child and Parent relation . Child will acquire the properties

(data and members) that are available in parent.

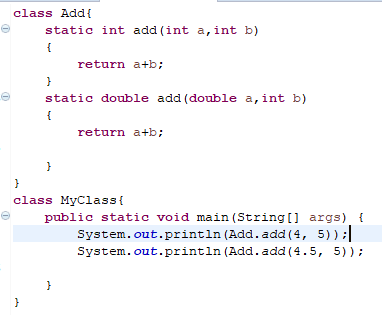
Inheritance is achieved by extends keywords.

Example:



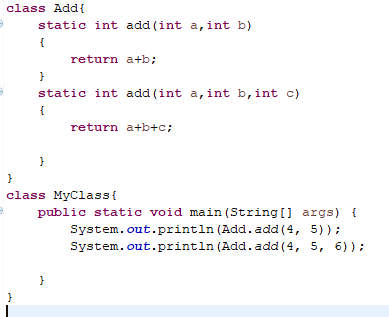
1. Polymorphism: Data enhancement / modification

a.(RunTime Polymorphism) Overriding: If we have same method signature Cannot be implemented in the same class.



b. (Compile Time)Overloading: If the method signature is changed.

Implemented in same class.



NOTE:-> If a programming language supports those 4 principles then that programming language is known as Object Oriented Programming Language.

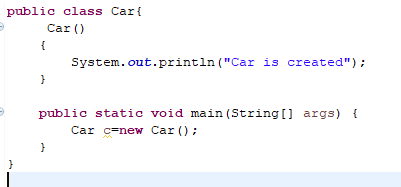
Constructor: A constructor is a block of codes similar to the method. It is called when an instance of the [class](https://www.javatpoint.com/object-and-class-in-java) is created. At the time of calling the constructor, memory for the object is allocated in the memory.

Rules:

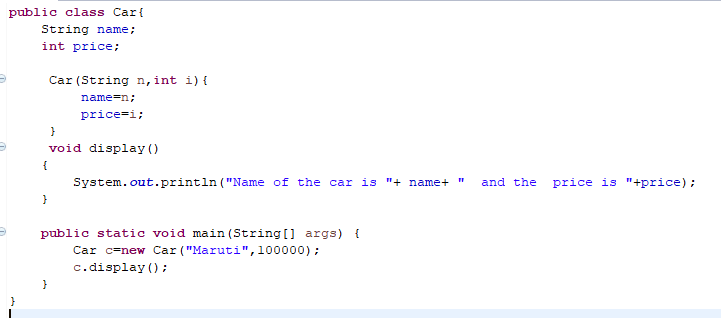
1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. A Java constructor cannot be abstract, static, final, and synchronized

Types:

1. Default Constructor : A constructor is called "Default Constructor" when it doesn't have any parameter.



1. Parameterized Constructor: A constructor which has a specific number of parameters is called a parameterized constructor.



Java Memory Model

The **major difference between Stack memory and heap memory** is that the stack is used to store the order of method execution and local variables while the heap memory stores the objects.

Image synchronization.

Java Memory +

ArrayList: It provides us with dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed. This class is found in java.util package.