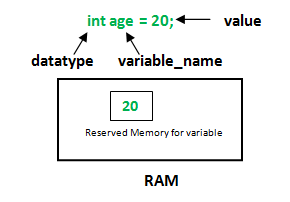
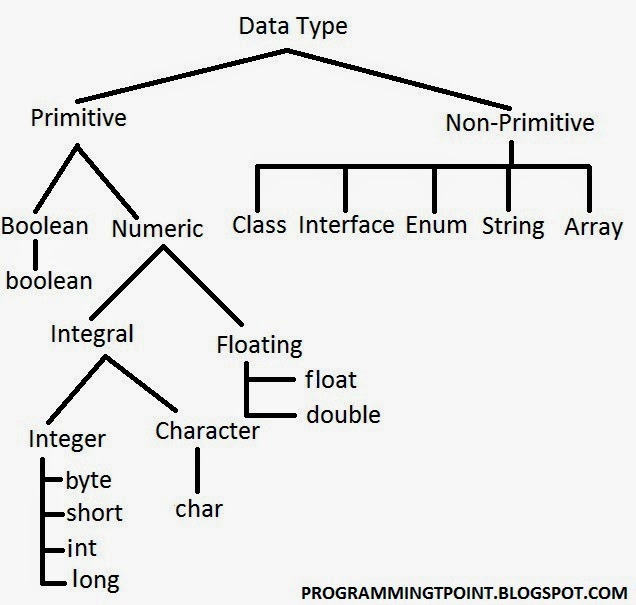
1. Variable in Java?

A **Java variable** is a piece of memory that can contain a data value.

A **variable** thus has a data type. Data types are covered in more detail in the text on **Java** data types.

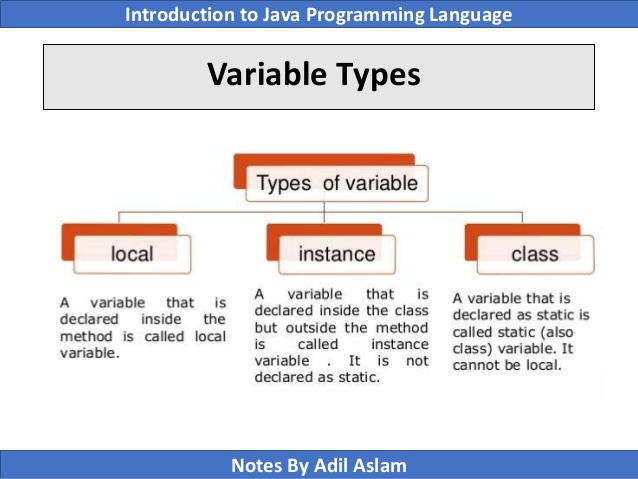
**Variables** are typically used to store information which your **Java** program needs to do its job.





**Here is a list of all primitives in Java:**

* byte (number, 1 byte)
* short (number, 2 bytes)
* int (number, 4 bytes)
* long (number, 8 bytes)
* float (float number, 4 bytes)
* double (float number, 8 bytes)
* char (a character, 2 bytes)
* boolean (true or false, 1 byte)



**Local variables**

Example

Here, *age* is a local variable. This is defined inside *pupAge()* method and its scope is limited to only this method.

public class Test {

public void pupAge() {

int age = 0;

age = age + 7;

System.out.println("Puppy age is : " + age);

}

public static void main(String args[]) {

Test test = new Test();

test.pupAge();

}

}

This will produce the following result −

### Output

Puppy age is: 7

**Instance variables**

Example

import java.io.\*;

public class Employee {

// this instance variable is visible for any child class.

public String name;

// salary variable is visible in Employee class only.

private double salary;

// The name variable is assigned in the constructor.

public Employee (String empName) {

name = empName;

}

// The salary variable is assigned a value.

public void setSalary(double empSal) {

salary = empSal;

}

// This method prints the employee details.

public void printEmp() {

System.out.println("name : " + name );

System.out.println("salary :" + salary);

}

public static void main(String args[]) {

Employee empOne = new Employee("Ransika");

empOne.setSalary(1000);

empOne.printEmp();

}

}

This will produce the following result −

Output

name : Ransika

salary :1000.0

**Class/Static variables**

Example

import java.io.\*;

public class Employee {

// salary variable is a private static variable

private static double salary;

// DEPARTMENT is a constant

public static final String DEPARTMENT = "Development ";

public static void main(String args[]) {

salary = 1000;

System.out.println(DEPARTMENT + "average salary:" + salary);

}

}

This will produce the following result −

Output

Development average salary:1000

## **Class/Static Variables**

* Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
* There would only be one copy of each class variable per class, regardless of how many objects are created from it.
* Static variables are rarely used other than being declared as constants. Constants are variables that are declared as public/private, final, and static. Constant variables never change from their initial value.
* Static variables are stored in the static memory. It is rare to use static variables other than declared final and used as either public or private constants.
* Static variables are created when the program starts and destroyed when the program stops.
* Visibility is similar to instance variables. However, most static variables are declared public since they must be available for users of the class.
* Default values are same as instance variables. For numbers, the default value is 0; for Booleans, it is false; and for object references, it is null. Values can be assigned during the declaration or within the constructor. Additionally, values can be assigned in special static initializer blocks.
* Static variables can be accessed by calling with the class name *ClassName.VariableName*.
* When declaring class variables as public static final, then variable names (constants) are all in upper case. If the static variables are not public and final, the naming syntax is the same as instance and local variables.

**What is Methods in Java??**

A **Java method** is a collection of statements that are grouped together to perform an operation. ... Now you will learn how to create your own **methods** with or without return values, invoke a **method** with or without parameters, and apply **method** abstraction in the program design.

****

**Basic Java method**  **types are:**

* Primitive types: int, short, long, double, float, boolean, byte, char.
* Wrapper classes for primitive types: java.lang.Integer, java.lang.Short, java.lang.Long, java.lang.Double, java.lang.Float, java.lang.Boolean, java.lang.Byte, java.lang.Character.
* String type: java.lang.String.

Object

A **Java object** is a combination of data and procedures working on the available data. An **object**has a state and behavior. The state of an **object** is stored in fields (variables), while methods (functions) display the **object's** behavior. **Objects** are created from templates known as classes.