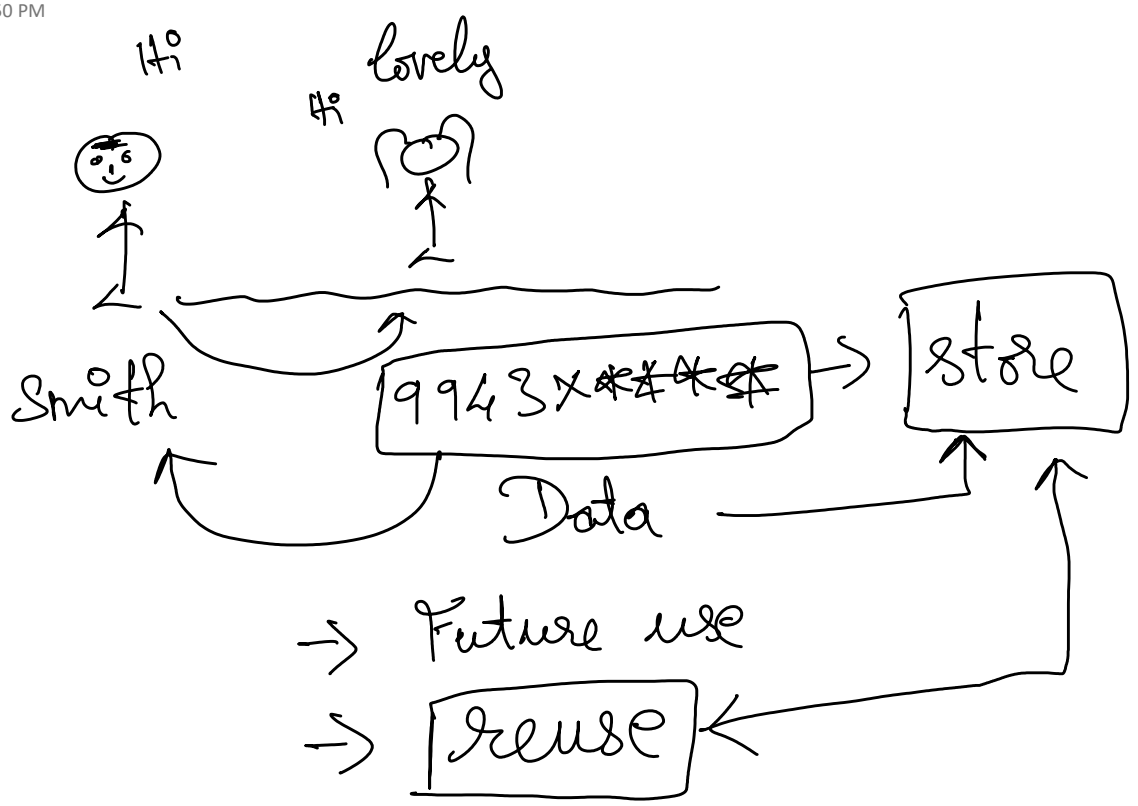
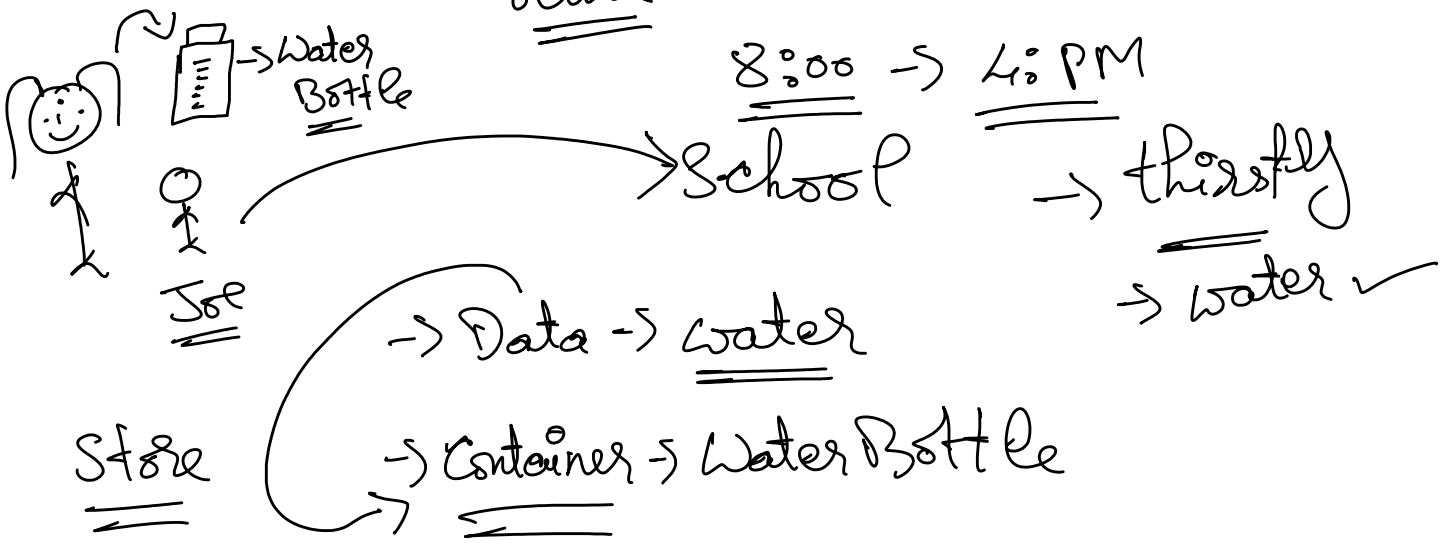
Wednesday, April 15, 2020 12:50 PM



Data → store → use

reuse



Water → lunch Box

lunch ⟶ WaterBottle

Data ⟶ (to store data) container ····› variables

specific Data ⟶ specific container

In Java, to store Data:

step 1: create container ✓

step 2: We need to specify the type container

How do I specify the type?

data types

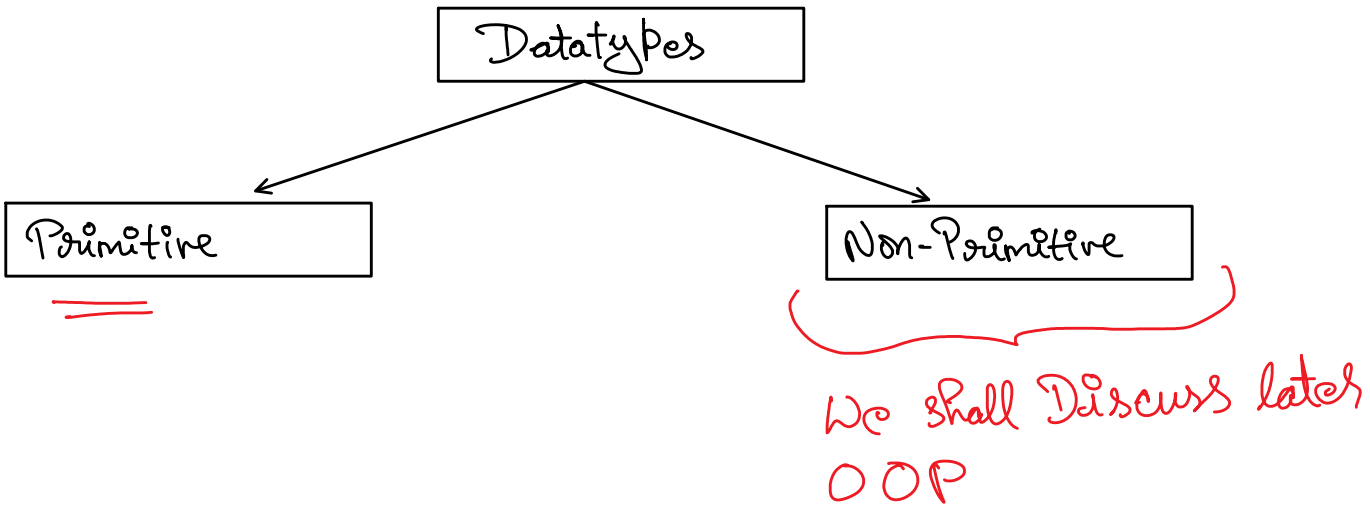
Wednesday, April 15, 2020 1:07 PM

Datatypes :

Datatypes helps to create containers to store the data.
or
Datatypes are used to create variables.

In java Datatypes are classified into Two types :

1. primitive datatypes
2. non-primitive datatypes / user defined / Derived



Primitive Datatypes :

Primitive datatypes are used to store all the types of literals(data) except String literal.

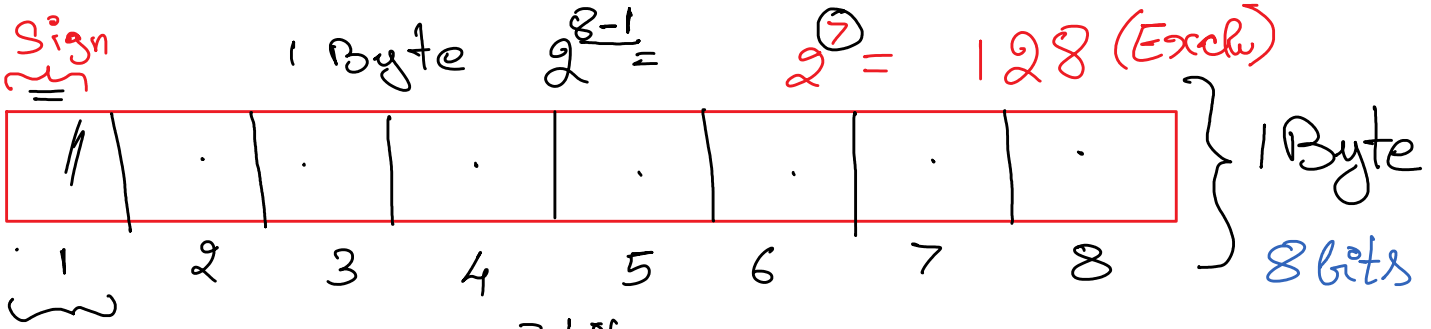
Therefore, with the help of primitive datatypes we can store :

- numbers (integers and decimals)
- characters
- Boolean

Note:
all primitives datatypes are keywords of java.

Literal	Data type	default value	size
Integer	1.byte	0	1 byte (8 bits)

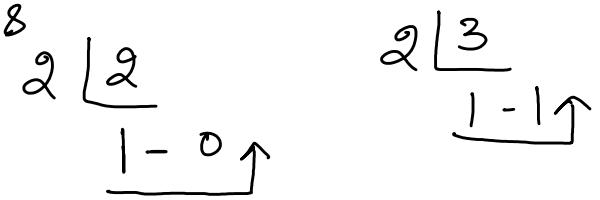
Integer	2.short	0	2 bytes (16 bits)
Integer	3.int	0	4 bytes (32 bits)
Integer	4.long	0l / 0L	8 bytes (64 bits)
Decimal	1.float	0.0f / 0.0F	4 bytes
Decimal	2.double	0.0d / 0.0D	8 bytes
Characters	1.char	/u0000	2 bytes
Boolean	1.boolean	false	1 bit

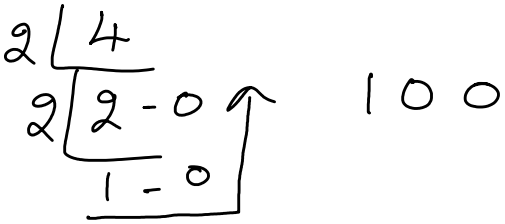


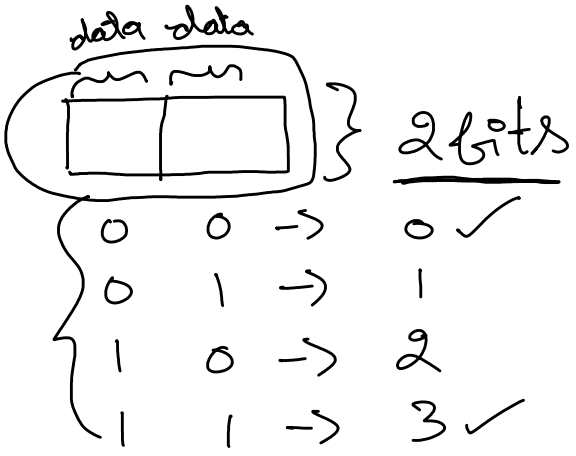
1 bit

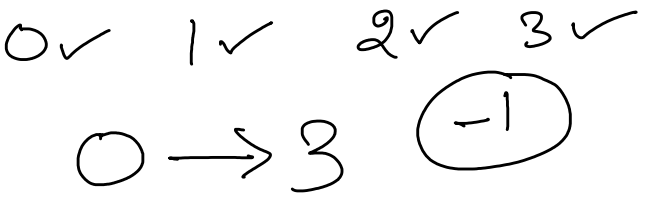
8 bits

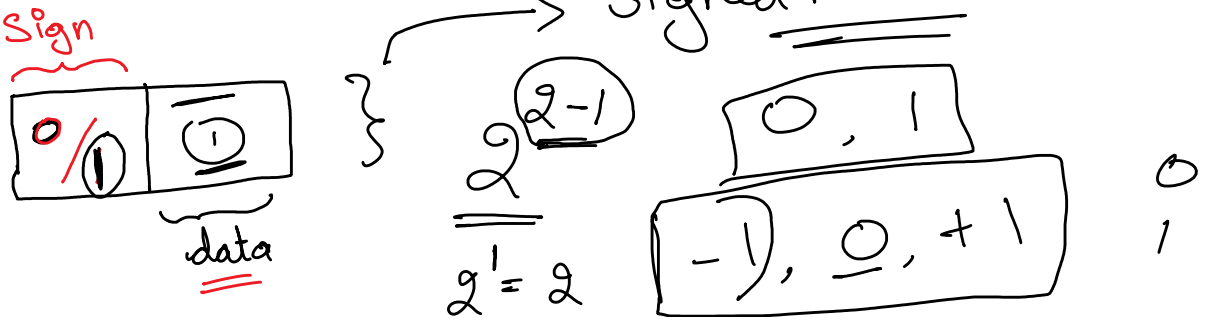
(0) ---- > 0
(1) -----> 1
(2) -----> 10 (2 bits)
(3) -----> 11 (2 bits)
(4) -----> 100 (3 bits)

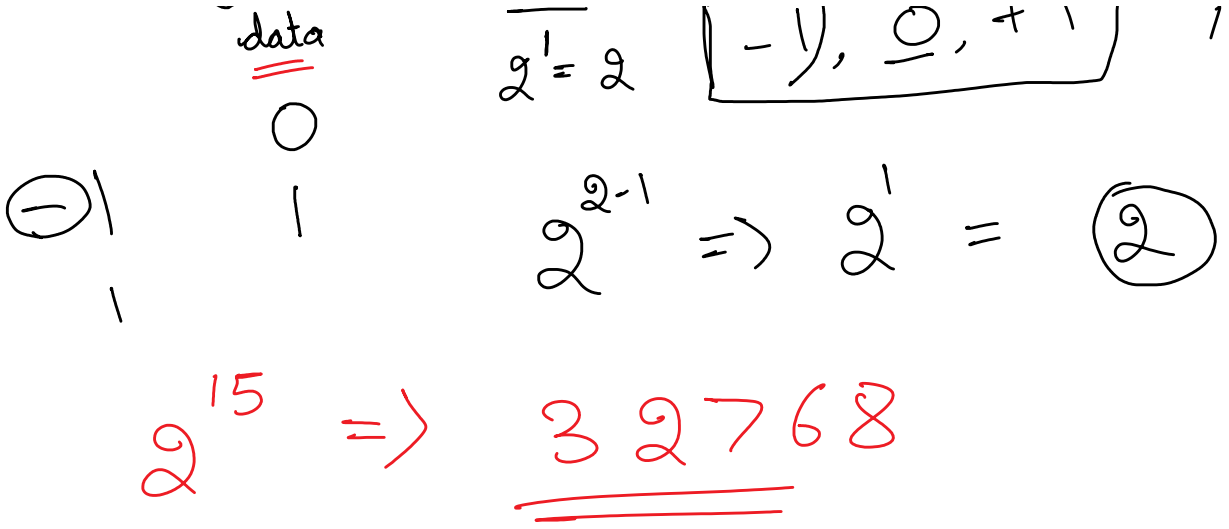






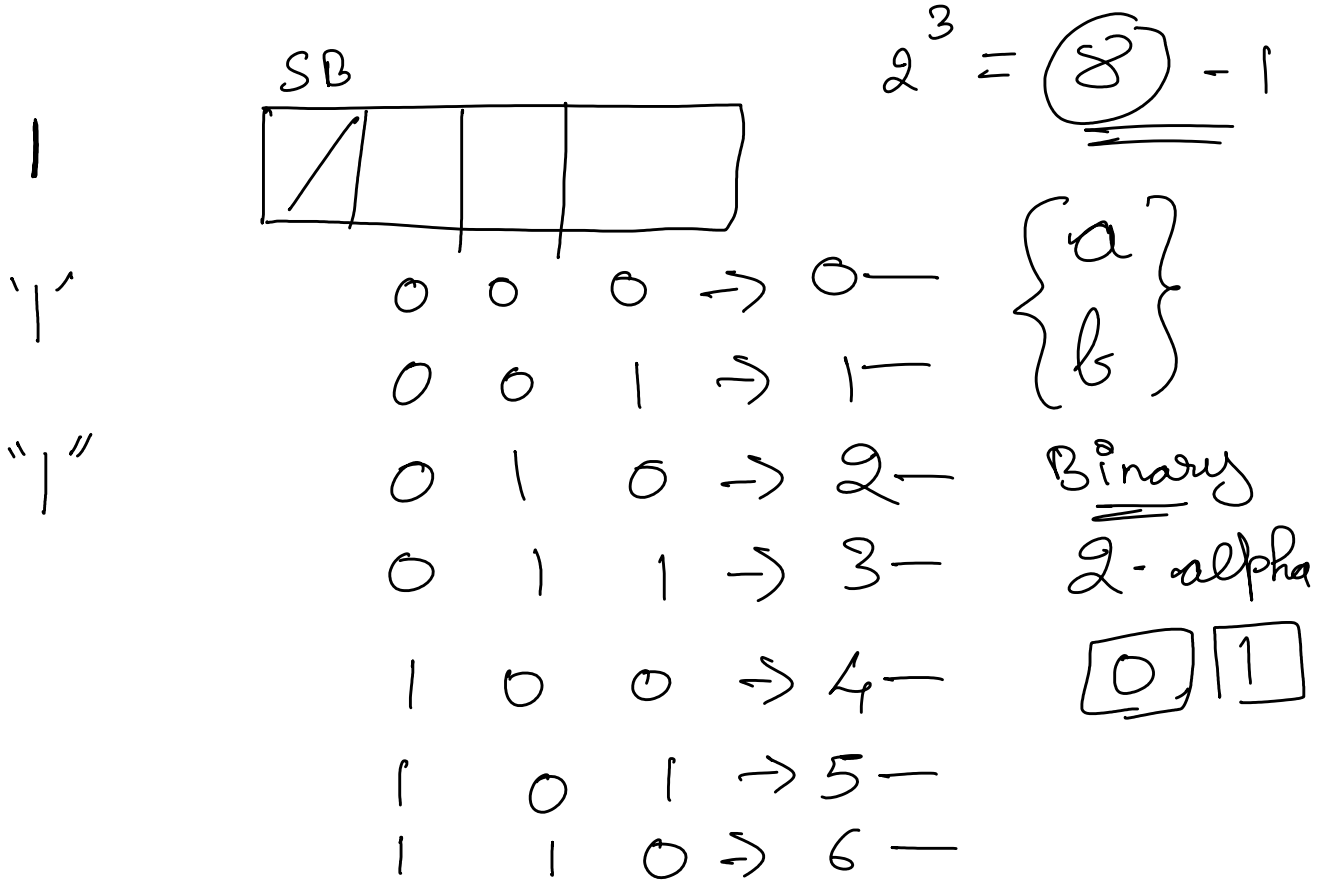






Number Datatypes in the increasing Order of their Capacity :

byte < short < int < long < float < double



Variable :

A variable is a container which is used to store data.
A variable is created with the help of **datatype,** the statement which is used to create a variable is known as **declaration statement**.

Syntax For Variable Declaration statement :

```
datatype   varaible_name1 ,  variable_name2 , … ;
```

Assignment Operator (=) :
it is used to store literal(data) into the variable.

Syntax for assignment statement :

```
variable =  literal / expression  ;
```

Task1 :
WAJP to store your age and print it.

Assignment1:
WAJP to store your date of birth as dd, mm, yyyy and print it in the following format dd / mm / yyyy

Rule (Semantic rule) :

1. the type of literal(data) should be same as the type of the variable, if not compiler tries to convert the literal into the type of the variable, If that is not possible then we get **Compile time error**.

   For examples, refer : **jp/app2/P4.java,P5.java,P6.java**

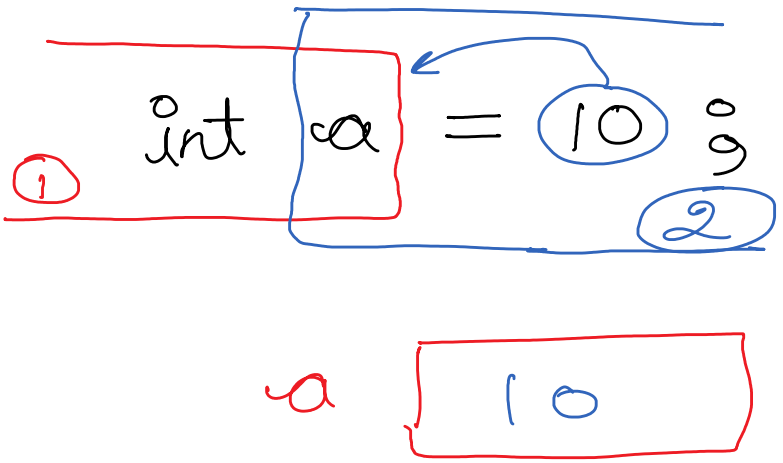
Declare & Initialization statement :

This stmt will perform 2 activities,

1. create a variable
2. assign the given value.

Syntax:

```
datatype  variable1 = literal/exp , variable2 , … ;
```



For example refer : **jp/app2/P7.java**

Assignment2:

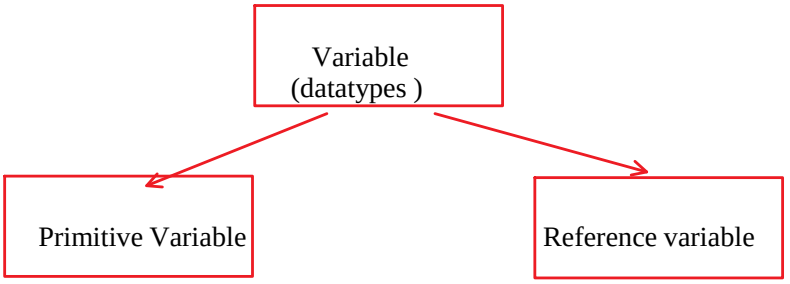
1. Write the syntax for declaration stmt, and write 5 java programs with one variable created in each of different data type.
2. Write the syntax for declare and initialization stmt and write 5 example programs.
3. Write a java program to demonstrate the Semantic rule.

Types Of Variables :

We can classify variables based on datatypes as well as the scope & life span of the variable.

- Based on datatypes :

1. primitive variables
2. non-primitive / reference variables



1. Primitive variables :

The variables created for primitive datatype is known as primitive variables.

ex: boolean x ; // primitive variable of boolean type
char y ; // primitive variable of char type

Note :

- primitive variables are used to store the values (primitive data)

2. Non-primitive / reference variables :

The variables created for non-primitive datatype, such for a class, for an array , for an interface are known as non-primitive/reference variables.

ex : String a ; // reference variable of String type

Task1 :

WAJP to store and print your name

```
class PrintName
{
	public static void main(String[] args)
	{
		String  name  =  "Dixith S N"  ;
		System.out.println( name ) ;
	}
}
```

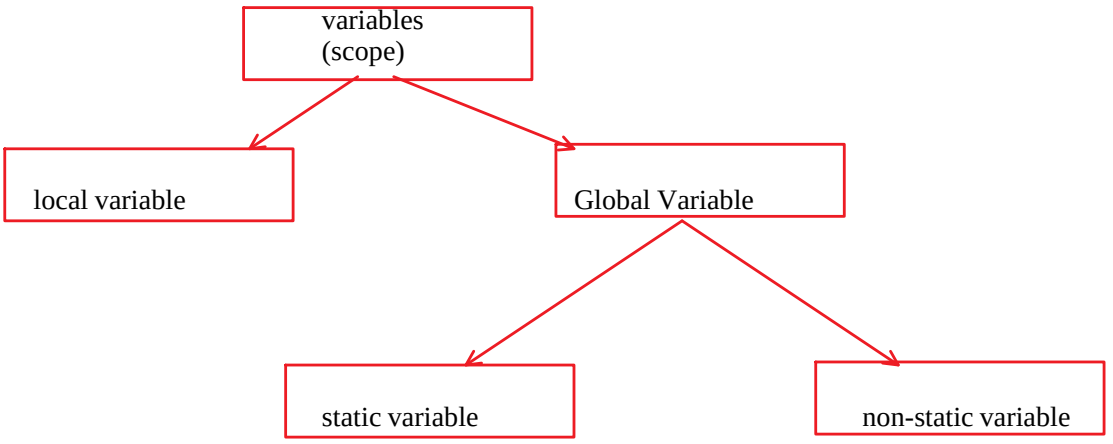
OutPut ;

```
D:\QCCA3\jp\app2>javac PrintName.java

D:\QCCA3\jp\app2>java PrintName
Dixith S N
```

- Based on Scope/(visibility) of a variable :

1. Local Variables
2. Global Variables




```
class    class_name
{  // class block begins

                // Global Area

    public static void main(String[] args )
    { // method block begins
```

```
                // local Area

    }// method block ends

            // Global Area

}// class block end
```

Note :

1. Class block area is considered as global area
2. the remaining block area, such as method block area is considered as local area.

1. Global Variable :

   A variable declared in class block is known as global variable. (we shall discuss later…)

2. Local Variable :

   A variable declared in a block other than class block is known as local variable.

Note :

1. local variables will not be assigned with default values.
2. We cannot use local variables without initialization. (if tried to use before initialization we get CTE)
   example refer, **jp/app2/P14.java**
3. a local variable can be used only within the block where it is created, it cannot be used outside the block.
   (Scope(Visibility) of local variable is within the block where it is created)
   for example refer,
   jp/app2/P15.java , P16.java, P17.java, P18.java