What is a Variable?

When we want to store any information, we store it in an address of the computer. Instead of remembering the complex address where we have stored our information, we name that address.The naming of an address is known as variable. Variable is the name of memory location.

In other words, variable is a name which is used to store a value of any type during program execution.

To declare the variable in Java, we can use following syntax

datatype variableName;

Java Programming language defines mainly three kind of variables.

1. Instance Variables(Object level)
2. Static Variables (Class Variables)
3. Local Variables

## **Instance variables in Java**

If the value of a variable is varied from object to object such type of variables are called as instance variables

For every object a separate copy of instance variables will be created

Instance variables also known as object level variables or attributes.

Instance variables should be declared within the class directly but outside of any method or block or constructor.

Instance variable will be created at the time of object creation and destroyed at the time of object destruction hence the scope of instance variable is a exactly same as scope of object

Instance variable will be stored in the HEAP memory as the part of object

We can not access instance variables directly from static area but we can access By using object reference

But we can access instance variable directly from instance area

EXAMPLE:

class Student

{

int a=10;

public static void main(String args[])

{

Student obj1=new Student();

System.out.println("obj1 a="+obj1.a);

}

void Display()

{

System.out.println("a="+a);

}

}

Instance variables are variables that are declare inside a class but outside any method

They are referred as object variable. Each object has its own copy of each variable and thus, it doesn't effect the instance variable if one object changes the value of the variable.

class Student

{

String name;

int age;

}

Copy

For instance variable JVM will always provide default values and we are not require to perform initialization explicitly.

class Student

{

int a;

double b;

String c;

Boolean d;

public static void main(String args[])

{

Student t1=new Student();

System.out.println(t1.a);//0

System.out.println(t1.b);//0.0

System.out.println(t1.c);//null

System.out.println(t1.d);//false

}

}

Here **name** and **age** are instance variable of Student class.

STATIC VARIABLES:(class level variables or fields)

1:If the value of a variable is not varied from object to object then its not recommended to declare variable as instance variable

We have to declare such type of variable at class level by using static modifier.

2:In the case of instance variables for every object a separate copy will be created in the case of static variables a single copy will be created at class level and shared by every object of the class.

3:Static variables should be declared within the class directly but outside of any method or block or constructor.

4:static variables will be created at the time of class loading and destroyed at the time of class unloading hence scope of static variable is exactly same as scope of .class file.

JVAM STARTS:

1:start jvm

2:create and start main thread

3:Loacate Test.class file

4:Load Test.class ------------🡪static variable creation

5:Execute main() method

6:Unload test.class-----------🡪static variable destruction

7:Terminate main Thread

8:Shutdown JVM.

->static variables will be stored in method area

->we can access static variables either by object reference or by class name but recommended to use class name

->within the same class it is not required to use class name and we can access directly .

class Student

{

static int a;

public static void main(String args[])

{

Student t1=new Student();

System.out.println(t1.a);

System.out.println(Student.b);

System.out.println(c);

}

}

->we can access static variables directly from both instance and static areas because static variable got created at beginning.

class Student

{

int a;

double b;

String c;

Boolean d;

public static void main(String args[])

{

Student t1=new Student();

System.out.println(t1.a);//0

System.out.println(t1.b);//0.0

System.out.println(t1.c);//null

System.out.println(t1.d);//false

}

}

class Student

{

static int x=10;

int y=20;

public static void main(String args[])

{

Student t1=new Student();

t1.x=888;

t1.y=999;

Student t2=new Student();

System.out.println("t2.x="+t2.x);

System.out.println("t2.y="+t2.y);

}

}

### Variable Scope in Java

Scope of a variable decides its accessibility throughout the program. As we have seen variables are different types so they have their own scope.

**Local variable**: Scope of local variable is limited to the block in which it is declared. For example, a variables declared inside a function will be accessible only within this function.

**Instance variable**: scope of instance variable depends on the access-modifiers **(public, private, default)**. If variable is declared as **private** then it is accessible within class only.

If variable is declared as **public** then it is accessible for all and throughout the application.

If variable is declared as **default** the it is accessible with in the same package.