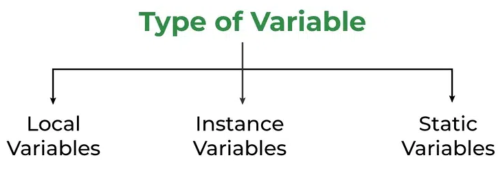
VARIABLES ADVANCED

Types of Variables

Now let us discuss different types of variables which are listed as follows:

- Local Variables
- Instance Variables
- Static Variables



Local Variables

A variable defined within a block or method or constructor is called a local variable.

- The Local variable is created at the time of declaration and destroyed after exiting from the block or when the call returns from the function.
- The scope of these variables exists only within the block in which the variables are declared, i.e., we can access these variables only within that block.
- Initialization of the local variable is mandatory before using it in the defined scope.

Instance Variables

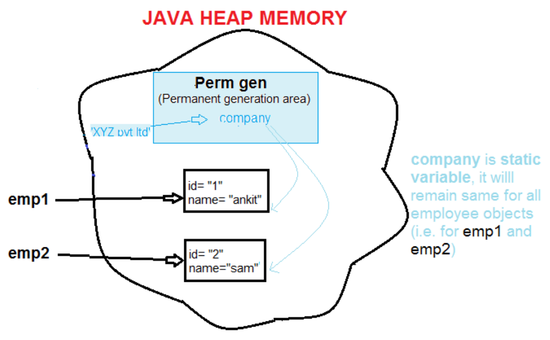
Instance variables are non-static variables and are declared in a class outside of any method, constructor, or block.

- As instance variables are declared in a class, these variables are created when an object of the class is created and destroyed when the object is destroyed.
- Unlike local variables, we may use access specifiers for instance variables. If we do not specify any access specifier, then the default access specifier will be used.
- Initialization of an instance variable is not mandatory. Its default value is dependent on the data type of variable. For String it is null, for float it is 0.0f, for int it is 0, for Wrapper classes like Integer it is null, etc.
- Instance variables can be accessed only by creating objects.
- We initialize instance variables using constructors while creating an object. We can also use instance blocks to initialize the instance variables.

Static Variable

Static variables are also known as class variables. These variables are declared similarly to instance variables. The difference is that static variables are declared using the static keyword within a class outside of any method, constructor, or block.

Unlike instance variables, we can only have one copy of a static variable per class, irrespective of how many objects we create. The static variables are stored in PermGen(Meta Space).


JAVA HEAP MEMORY
Perm gen
(Permanent generation area)
company
'XYZ pvt ltd'
emp1
id= "1"
name= "ankit"
emp2
id= "2"
name="sam"
company is static variable, it willl remain same for all employee objects (i.e. for emp1 and emp2)

Static variables are created at the start of program execution and destroyed automatically when execution ends.
Initialization of a static variable is not mandatory. Its default value is dependent on the data type of variable. For *String* it is *null*, for *float* it is *0.0f*, for *int* it is *0*, for *Wrapper classes* like *Integer* it is *null,* etc.

- If we access a static variable like an instance variable (through an object), the compiler will show a warning message, which won't halt the program. The compiler will replace the object name with the class name automatically.
- If we access a static variable without the class name, the compiler will automatically append the class name. But for accessing the static variable of a different class, we must mention the class name as 2 different classes might have a static variable with the same name.
- Static variables cannot be declared locally inside an instance method. Instance methods cannot be called directly inside a static method, without creating an object. Instance method can be called inside a Instance method. As we know somehow the method will be called from static method and thus object should have been created for that.
- Static blocks can be used to initialize static variables. Static Block executes just after programme is executed, the main function is run.
- Static variable can call by directly with the help of class only, we do not need to create object for the class in this.
- 'this' inside a static context don't replaces this with the class name, instead it will give an error.

Suppose there are 25 students in the Production Engineering department of NIT Agartala. All students have its unique enrolment number, registration number, and name. So instance data member is good in such a case. Now all instance data members will get memory each time when the object is created. Here, "department" refers to the common property of all the objects. If we make it static, this field will get the memory only once.
Thus static variables can be used to refer to the common property of all objects (which is not unique for each object), for example, college name of students, company name of employees, CEO of a company, etc. It makes the program memory efficient (i.e., it saves memory).