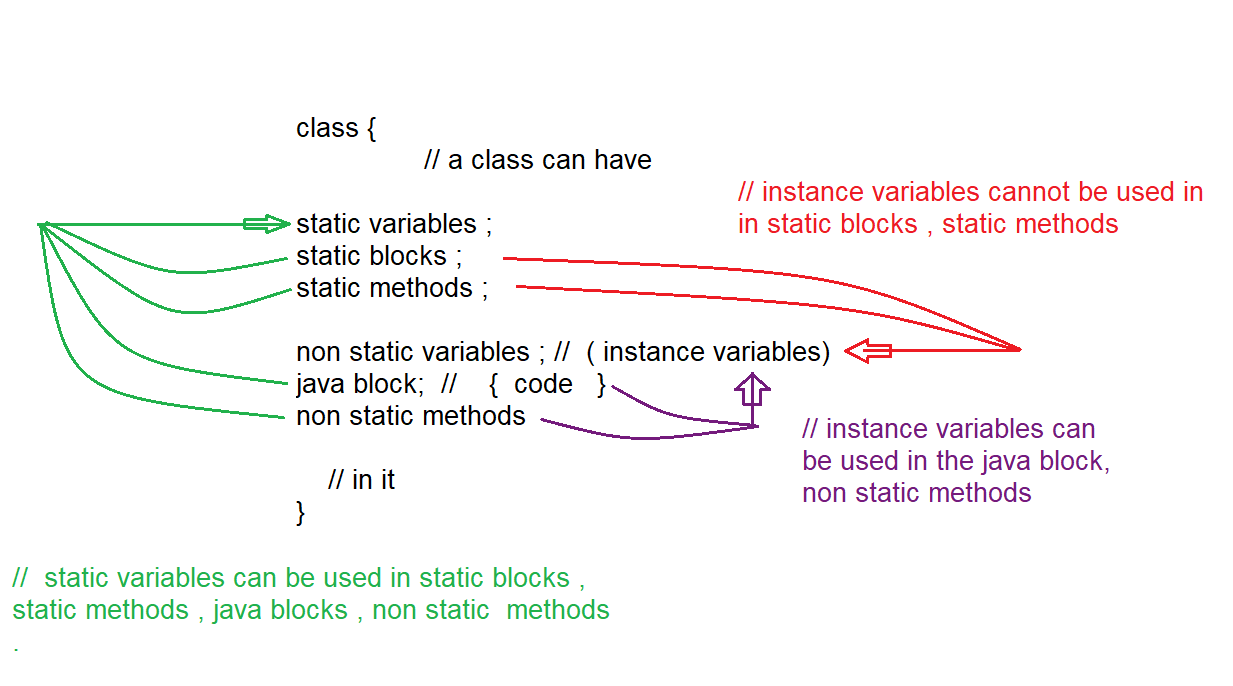
Static keyword:



Note: main() is not the first one to execute in the program , jvm will decide whether to execute the program or not , if program has main() it is executed .

Order of execution of java program

1. Static variable
2. Static block
3. Static method (main method is also static)
4. Non static variables
5. non static block
6. constructor
7. non static method

jvm evaluates the program in the above order.

but you can invoke them in main() in different order of your choice .

Eg: Order\_Of\_Execution

Note: when object is created 3 things will happen

1. memory for instance variables is allocated
2. java block is executed
3. constructor is executed.

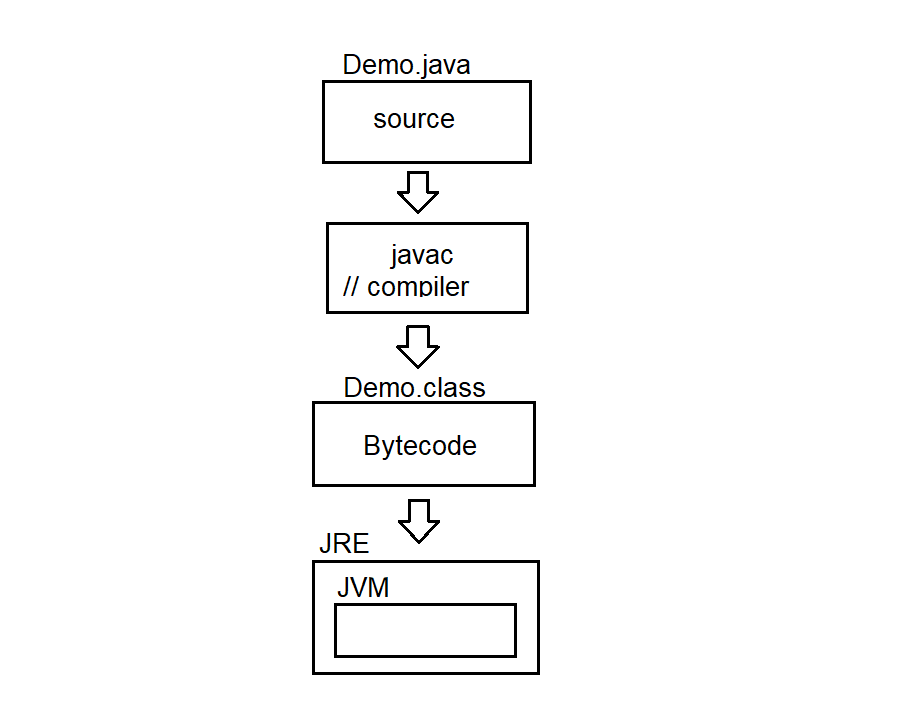
Note: static block will have no name.

Syntax: static {

// Code

}

Program is executed as follows:



JRE (java runtime environment) internally has JVM (java virtual machine)

Byte code is given to JRE

JRE gives the byte code to the JVM

JVM takes that bytecode, and process it internally

Note: static variables and static blocks are executed during class loading, main method is executed by jvm by default

Note: if static method and main method is in same class you can call static method directly with method\_name , class\_name.method\_name , with obj\_ref.method\_name also

Eg: Static\_Methods

If static method and main method is in different class you should call static method with class\_name.method, obj\_ref.method\_name (here object reference should be of class type in which static method is present)

Eg: Static\_Method\_1

Static variables, static block , static method ( only main method ) is executed automatically

Eg: Automatic\_Execution

// go through the code

Static method (user defined), constructor, normal method is not executed automatically, it is executed if and only if we call them.

Eg: Manual\_Execution

// go through the code

Note: we can call method inside a method, but cannot create new method inside a method.

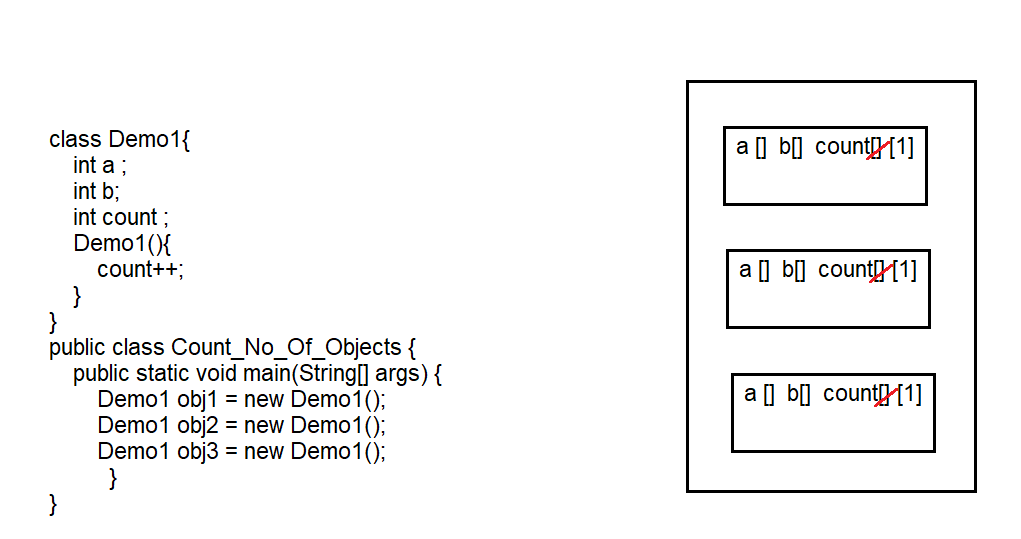
Note: non static variables (instance variables) cannot be used in static block and static method, because memory for instance variables is allocated only when object is created.

But static blocks, and static methods are executed (at class loading) even before memory of instance variables is allocated. so instance variables are not allowed in static blocks and static methods.

Note: you can use static variables in all the static blocks, static methods, java block, constructor, non-static method. since their memory is allocated at first (during class loading itself)

Need of static

Eg: Count\_No\_Of\_Objects



Here count is instance variable , memory is allocated every time the object is created . so cannot calculate the objects . the possible way is make count as static . then memory is allocated once and can be incremented every time when object is created .

Eg: Count\_No\_Of\_Objects\_Using\_Static\_Variable\_In\_Constructor

// here static variable count is written in constructor , if the variable is not written in one onn the constructors we cannot get the correct no of objects. And also size of program increases there by decreasing performance .

To overcome that

Note : which ever content is common is all the constructors include that in the non static block.

If it is kept in a static block it is executed only once . but java block will be executed no of times whenever object is created.

Eg: Count\_No\_Of\_Objects\_Using\_Non\_Static\_Block

Explanation

Whenever constructor is executed , it will treat non static block as its body first executes it , and then executes the actual body.

Static variables memory is created once and can be used multiple times .

Non static block is executed even before constructor’s body . this() (or) super() should be the first one to be executed in constructor actual body . but even before that java block is executed first as a constructor body and then actual constructor body is executed .

Eg: java\_block\_Execution

Static variable :

1. static keyword is used
2. executed during class loading
3. memory is allocated in heap area
4. memory is allocated only once in heap .
5. one copy is used by all the objects
6. can be called using class name.
7. also called as class variables
8. object independent
9. can be used in static elements and non static elements

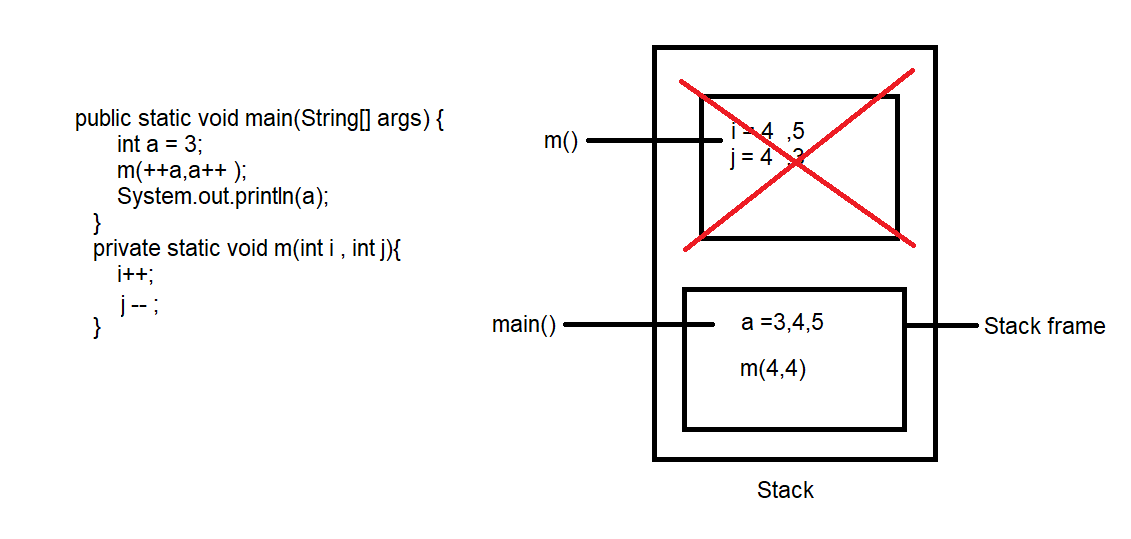
Note :

1. if something needs to be executed even before main method and during class loading , write them in static block , generally we use to initialize static variables .

Eg: Static\_Block\_Execution

1. if something need to be executed during object creation write them in constructor (or) non static block. And if there is some common statements in all the constructors write them in java block .

Eg: Static\_Method



Here first main method is brought to stack frame and executed , method m() is called in main method , so another stack frame will be created for method m(), after the execution of method m() , it is removed from stack area , and local variables i,j has scope only until its execution completes . so a value is fetched from main method which is 5 .

Note : Pass by value : whenever you pass the data w.r.t variable , those changes will not be reflected in the memory (example above program there we passed values to method and there are not reflected in the memory )

Pass by reference :

Eg: Fizz here you are passing object references , whatever changes are made they are stored in the object and reflected in main method.

Note: whenever a common copy of data has to be accessed by all the objects then make that data member as static .

If the data is object specific then make that data member as instance variable .

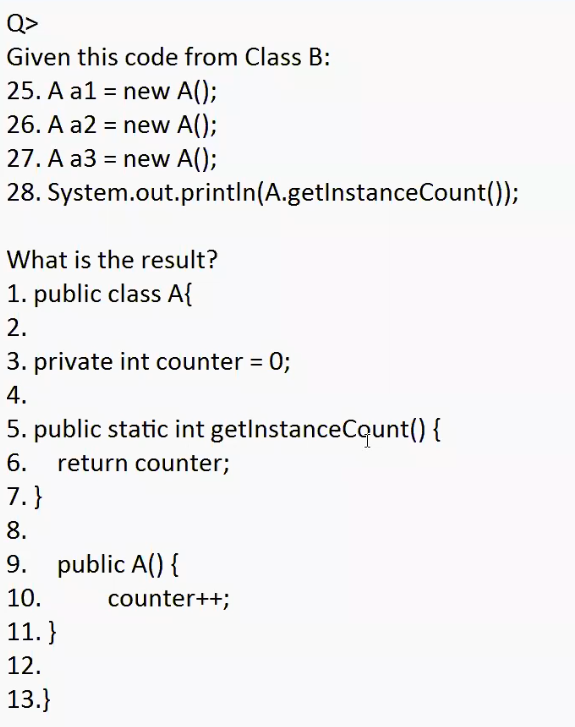
Eg: Farmer\_Loan

// here rate of interest is common to all the objects (farmers) so it is made static

// principal amount, time period (object specific ) is different for every farmer so used instance varaibles

Note: static methods are also known as generic methods since they are not object dependent.

Instance methods are also known as specific methods since they are object specific (object dependent) .



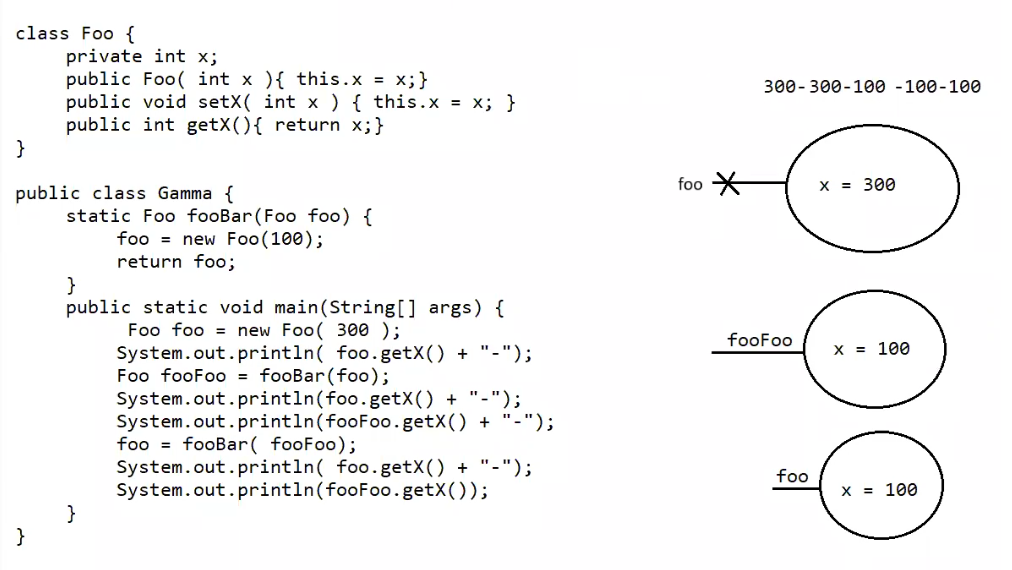
Ans : Instance variable counter cant be accessed in the static method getInstanceCount() , which leads to compilation error.

Eg: Static\_Method\_Eg2

// go through the code

Eg: Gamma

// go through the code



Eg: Static\_Block\_Eg5

