Static methods vs Instance methods in Java

Instance Method

Instance method are methods which require an object of its class to be created before it can be called.
To invoke a instance method, we have to create an Object of the class in within which it defined.

```
public void geek(String name)
{
 // code to be executed....
}

// Return type can be int, float String or user defined data type.
```

Memory allocation: These methods themselves are stored in Permanent Generation space of heap
but the parameters (arguments passed to them) and their local variables and the value to be returned are allocated in stack.

They can be called within the same class in which they reside or from the different classes
defined either in the same package or other packages depend on the **access type** provided to the desired instance method.

Important Points:

- Instance method(s) belong to the Object of the class not to the class i.e. they can be called after creating the Object of the class
  .
- Every individual Object created from the class has its own copy of the instance method(s) of that class.

- Instance methods are not stored on a per-instance basis, even with virtual methods. They're stored in a single memory location, and they only "know" which object they belong to because the this pointer is passed when you call them.

- They can be overridden since they are resolved using **dynamic binding** at run time.

```
// Example to illustrate accessing the instance method .
importjava.io.*;

classFoo{

    String name = "";

    // Instance method to be called within the same class or
    // from a another class defined in the same package
    // or in different package.
    publicvoidgeek(String name){

        this.name = name;
    }
}

classGFG {
    publicstaticvoidmain (String[] args) {

        // create an instance of the class.
        Foo ob = newFoo();

        // calling an instance method in the class 'Foo'.
        ob.geek("GeeksforGeeks");
        System.out.println(ob.name);
    }
}
```

Output :

```
GeeksforGeeks
```

Static Method

Static methods are the methods in Java that can be called without creating an object of class. They are referenced by the **class name itself** or reference to the Object of that class.

```
public static void geek(String name)
{
 // code to be executed....
```

```
}

// Must have static modifier in their declaration.
// Return type can be int, float, String or user defined data type.
```

Memory Allocation: They are stored in Permanent Generation space of heap as they are associated to the class in which they reside not to the objects of that class.
But their local variables and the passed argument(s) to them are stored in the stack.

Since they belong to the class so they can be called to without creating the object of the class.

Important Points:

- Static method(s) are associated to the class in which they reside i.e. they can be called even without creating an instance of the class i.eClassName.methodName(args).

- They are designed with aim to be shared among all Objects created from the same class.

- Static methods can not be overridden. But can be overloaded since they are resolved using **static binding** by compiler at compile time.

```
// Example to illustrate Accessing the Static method(s) of the class.
importjava.io.*;

classGeek{

    publicstaticString geekName = "";

    publicstaticvoidgeek(String name){

        geekName = name;
    }
}

classGFG {
    publicstaticvoidmain (String[] args) {

        // Accessing the static method geek() and
        // field by class name itself.
```

```
        Geek.geek("vaibhav");
        System.out.println(Geek.geekName);

        // Accessing the static method geek() by using Object's reference.
        Geek obj = newGeek();
        obj.geek("mohit");
        System.out.println(obj.geekName);


    }
}
```

Output:

```
vaibhav
mohit
```

Note: Static variables and their values (primitives or references) defined in the class are stored in **PermGen** space of memory.

What if static variable refers to an Object ?

```
staticinti = 1;
static Object obj = new Object();
```

In first line, the value 1 would be stored in PermGen section. In second line, the reference obj would be stored in PermGen section and the Object it refers to would be stored in heap section.

When to use static methods ??

- When you have code that can be shared across all instances of the same class, put that portion of code into static method.
- They are basically used to access static field(s) of the class.

Instance method vs Static method

- Instance method can access the instance methods and instance variables directly.
- Instance method can access static variables and static methods directly.
- Static methods can access the static variables and static methods directly.
- Static methods can't access instance methods and instance variables directly. They must use reference to object. And static method can't use this keyword as there is no instance for 'this' to refer to.

References

Java static keyword

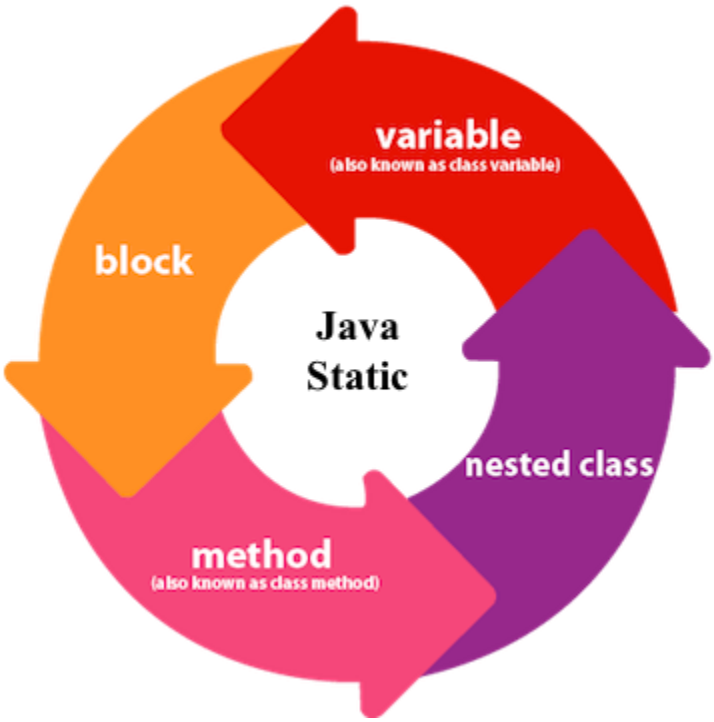
1. Static variable
2. Program of the counter without static variable

3. Program of the counter with static variable
4. Static method
5. Restrictions for the static method
6. Why is the main method static?
7. Static block
8. Can we execute a program without main method?

The **static keyword** in Java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than an instance of the class.

The static can be:

1. Variable (also known as a class variable)
2. Method (also known as a class method)
3. Block
4. Nested class



1) Java static variable

If you declare any variable as static, it is known as a static variable.

- The static variable can be used to refer to the common property of all objects (which is not unique for each object), for example, the company name of employees, college name of students, etc.
- The static variable gets memory only once in the class area at the time of class loading.

Advantages of static variable

It makes your program **memory efficient** (i.e., it saves memory).

Understanding the problem without static variable

```
1. class Student{
2.     int rollno;
3.     String name;
4.     String college="ITS";
5. }
```

Suppose there are 500 students in my college, now all instance data members will get memory each time when the object is created. All students have its unique rollno and name, so instance data member is good in such case. Here, "college" refers to the common property of all objects. If we make it static, this field will get the memory only once.

> *Java static property is shared to all objects.*

Example of static variable

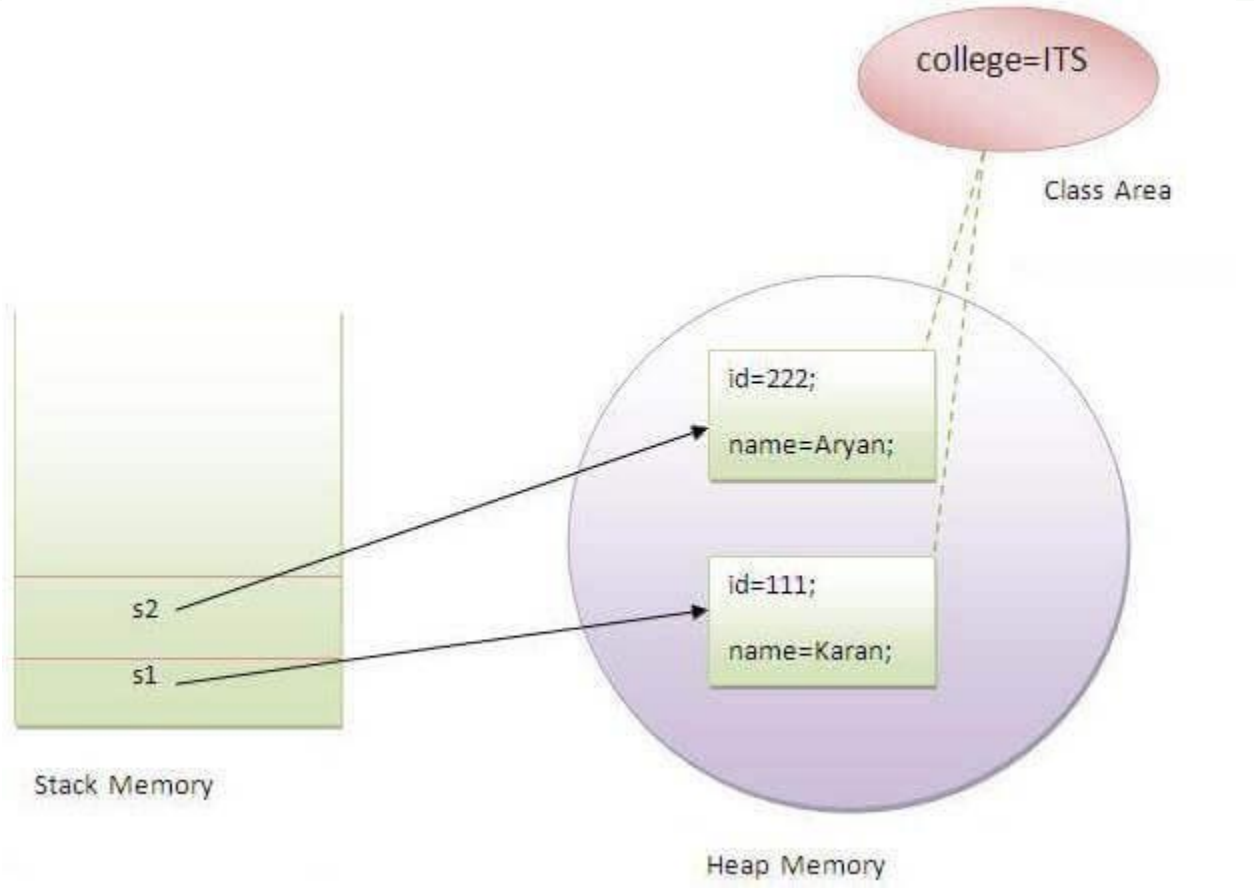
```
1. //Java Program to demonstrate the use of static variable
2. class Student{
3.    int rollno;//instance variable
4.    String name;
5.    static String college ="ITS";//static variable
6.    //constructor
7.    Student(int r, String n){
8.    rollno = r;
9.    name = n;
10.   }
11.   //method to display the values
12.   void display (){System.out.println(rollno+" "+name+" "+college);}
13. }
14. //Test class to show the values of objects
15. public class TestStaticVariable1{
```

```
16. public static void main(String args[]){
17. Student s1 = new Student(111,"Karan");
18. Student s2 = new Student(222,"Aryan");
19. //we can change the college of all objects by the single line of code
20. //Student.college="BBDIT";
21. s1.display();
22. s2.display();
23. }
24. }
```

Test it Now

Output:

```
111 Karan ITS
222 Aryan ITS
```



Program of the counter without static variable

In this example, we have created an instance variable named count which is incremented in the constructor. Since instance variable gets the memory at the time of object creation, each object will have the copy of the instance variable. If it is incremented, it won't reflect other objects. So each object will have the value 1 in the count variable.

```
1. //Java Program to demonstrate the use of an instance variable
2. //which get memory each time when we create an object of the class.
3. class Counter{
4. int count=0;//will get memory each time when the instance is created
5.
6. Counter(){
7. count++;//incrementing value
8. System.out.println(count);
9. }
10.
11. public static void main(String args[]){
12. //Creating objects
13. Counter c1=new Counter();
14. Counter c2=new Counter();
15. Counter c3=new Counter();
16. }
17. }
```

Test it Now

Output:

```
1
1
1
```

Program of counter by static variable

As we have mentioned above, static variable will get the memory only once, if any object changes the value of the static variable, it will retain its value.

```
1. //Java Program to illustrate the use of static variable which
2. //is shared with all objects.
3. class Counter2{
4. static int count=0;//will get memory only once and retain its value
5.
```

```
6.  Counter2(){
7.  count++;//incrementing the value of static variable
8.  System.out.println(count);
9.  }
10.
11. public static void main(String args[]){
12. //creating objects
13. Counter2 c1=new Counter2();
14. Counter2 c2=new Counter2();
15. Counter2 c3=new Counter2();
16. }
17. }
```

Test it Now

Output:

```
1
2
3
```

---

2) Java static method

If you apply static keyword with any method, it is known as static method.

- A static method belongs to the class rather than the object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- A static method can access static data member and can change the value of it.

Example of static method

```
1.  //Java Program to demonstrate the use of a static method.
2.  class Student{
3.      int rollno;
4.      String name;
5.      static String college = "ITS";
6.      //static method to change the value of static variable
7.      static void change(){
8.      college = "BBDIT";
9.      }
10.     //constructor to initialize the variable
11.     Student(int r, String n){
12.     rollno = r;
```

```
13.     name = n;
14.     }
15.     //method to display values
16.     void display(){System.out.println(rollno+" "+name+" "+college);}
17. }
18. //Test class to create and display the values of object
19. public class TestStaticMethod{
20.    public static void main(String args[]){
21.    Student.change();//calling change method
22.    //creating objects
23.    Student s1 = new Student(111,"Karan");
24.    Student s2 = new Student(222,"Aryan");
25.    Student s3 = new Student(333,"Sonoo");
26.    //calling display method
27.    s1.display();
28.    s2.display();
29.    s3.display();
30.    }
31. }
```

Test it Now

```
Output:111 Karan BBDIT
       222 Aryan BBDIT
       333 Sonoo BBDIT
```

Another example of a static method that performs a normal calculation

```
1.  //Java Program to get the cube of a given number using the static method
2.
3.  class Calculate{
4.    static int cube(int x){
5.    return x*x*x;
6.    }
7.
8.    public static void main(String args[]){
9.    int result=Calculate.cube(5);
10.   System.out.println(result);
11.   }
12. }
```

Test it Now

```
Output:125
```

Restrictions for the static method

There are two main restrictions for the static method. They are:

1. The static method can not use non static data member or call non-static method directly.
2. this and super cannot be used in static context.

```
1. class A{
2.  int a=40;//non static
3.
4.  public static void main(String args[]){
5.   System.out.println(a);
6.  }
7. }
```

Test it Now

```
Output:Compile Time Error
```

Q) Why is the Java main method static?

Ans) It is because the object is not required to call a static method. If it were a non-static method, JVM creates an object first then call main() method that will lead the problem of extra memory allocation.

---

3) Java static block

- Is used to initialize the static data member.
- It is executed before the main method at the time of classloading.

Example of static block

```
1. class A2{
2.   static{System.out.println("static block is invoked");}
3.   public static void main(String args[]){
4.    System.out.println("Hello main");
5.   }
6. }
```

```
Output: static block is invoked
        Hello main
```

Q) Can we execute a program without main() method?

Ans) No, one of the ways was the static block, but it was possible till JDK 1.6. Since JDK 1.7, it is not possible to execute a java class without the main method.

```
1. class A3{
2.   static{
3.   System.out.println("static block is invoked");
4.   System.exit(0);
5.   }
6. }
```

Output:

```
static block is invoked
```

Since JDK 1.7 and above, output would be:

```
Error: Main method not found in class A3, please define the main method as:
public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
```