Java static keyword

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 instance of the class.

The static can be:

1. variable (also known as class variable)
2. method (also known as class method)
3. block
4. nested class

---

1) Java static variable

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

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

Advantage of static variable

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

Understanding problem without static variable

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

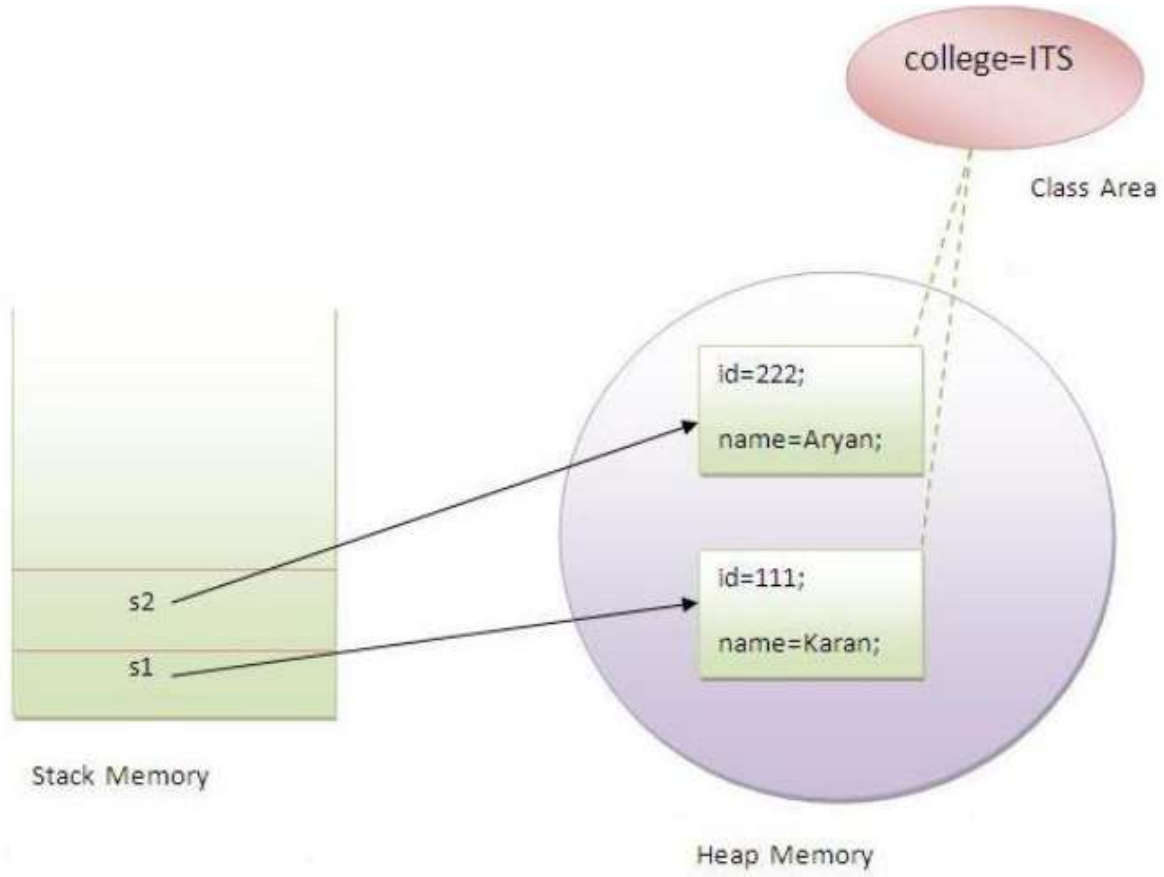
Suppose there are 500 students in my college, now all instance data members will get memory each time when object is created. All student have its unique rollno and name so instance data member is good. Here, college refers to the common property of all objects. If we make it static, this field will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

Java static property is shared to all objects.

Example of static variable

```
1. //Program of static variable
2.
3. class Student8{
4.    int rollno;
5.    String name;
6.    static String college ="ITS";
7.
8.    Student8(int r,String n){
9.         rollno = r;
10.          name = n;
11.          }
12.         void display (){System.out.println(rollno+" "+name+" "+college);
    }
13.
14.         public static void main(String args[]){
15.         Student8 s1 = new Student8(111,"Karan");
16.         Student8 s2 = new Student8(222,"Aryan");
17.
18.         s1.display();
19.         s2.display();
20.         }
21.        }
```

```
Output:111 Karan ITS
       222 Aryan ITS
```



Program of 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 to other objects. So each objects will have the value 1 in the count variable.

```
1. class Counter{
2. int count=0;//will get memory when instance is created
3.
4. Counter(){
5. count++;
6. System.out.println(count);
7. }
8.
9. public static void main(String args[]){
10.
11.      Counter c1=new Counter();
12.      Counter c2=new Counter();
13.      Counter c3=new Counter();
```

```
14.
15.       }
16.      }
```

```
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. class Counter2{
2. static int count=0;//will get memory only once and retain its value
3.
4. Counter2(){
5. count++;
6. System.out.println(count);
7. }
8.
9. public static void main(String args[]){
10.
11.      Counter2 c1=new Counter2();
12.      Counter2 c2=new Counter2();
13.      Counter2 c3=new Counter2();
14.
15.       }
16.      }
```

```
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 object of a class.

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

Example of static method

```
1. //Program of changing the common property of all objects(static field).
2.
3. class Student9{
4.     int rollno;
5.     String name;
6.     static String college = "ITS";
7.
8.     static void change(){
9.     college = "BBDIT";
10.         }
11.
12.         Student9(int r, String n){
13.         rollno = r;
14.         name = n;
15.         }
16.
17.         void display (){System.out.println(rollno+" "+name+" "+colleg
    e);}
18.
19.         public static void main(String args[]){
20.         Student9.change();
21.
22.         Student9 s1 = new Student9 (111,"Karan");
23.         Student9 s2 = new Student9 (222,"Aryan");
24.         Student9 s3 = new Student9 (333,"Sonoo");
25.
26.         s1.display();
27.         s2.display();
28.         s3.display();
29.         }
30.     }
```

```
Output:111 Karan BBDIT
```

```
222 Aryan BBDIT
333 Sonoo BBDIT
```

Another example of static method that performs normal calculation

```
1. //Program to get cube of a given number by 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.       }
```

```
Output:125
```

Restrictions for static method

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

1. The static method cannot 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. }
```

```
Output:Compile Time Error
```

Q) why java main method is static?

Ans) because object is not required to call static method if it were non-static method, jvm create 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 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) Yes, one of the way is static block but in previous version of JDK not in JDK 1.7.

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

```
Output:static block is invoked (if not JDK7)
```

In JDK7 and above, output will be:

```
Output:Error: Main method not found in class A3, please define
the main method as:
public static void main(String[] args)
```

Static class in Java

Can a class be static in Java ?

The answer is YES, we can have static class in java. In java, we have static instance variables as well as static methods and also static block. Classes can also be made static in Java.
Java allows us to define a class within another class. Such a class is called a nested class. The class which enclosed nested class is known as Outer class. In java, we can't make Top level class static. ***Only nested classes can be static***.

What are the differences between static and non-static nested classes? Following are major differences between static nested class and non-static nested class. Non-static nested class is also called Inner Class.
1) Nested static class doesn't need reference of Outer class, but Non-static nested class or Inner class requires Outer class reference.
2) Inner class(or non-static nested class) can access both static and non-static members of Outer class. A static class cannot access non-static members of the Outer class. It can access only static members of Outer class.
3) An instance of Inner class cannot be created without an instance of outer class and an Inner class can reference data and methods defined in Outer class in which it nests, so we don't need to pass reference of an object to the constructor of the Inner class. For this reason Inner classes can make program simple and concise.

```
/* Java program to demonstrate how to implement static and non-static
   classes in a java program. */
class OuterClass{
   private static String msg = "GeeksForGeeks";

   // Static nested class
   public static class NestedStaticClass{

        // Only static members of Outer class is directly accessible in nested
        // static class
        public void printMessage() {

          // Try making 'message' a non-static variable, there will be
          // compiler error
          System.out.println("Message from nested static class: " + msg);
        }
    }

    // non-static nested class - also called Inner class
    public class InnerClass{

        // Both static and non-static members of Outer class are accessible in
        // this Inner class
        public void display(){
           System.out.println("Message from non-static nested class: "+ msg);
        }
    }
```

```
}
class Main
{
    // How to create instance of static and non static nested class?
    public static void main(String args[]){

        // create instance of nested Static class
        OuterClass.NestedStaticClass printer = new
OuterClass.NestedStaticClass();

        // call non static method of nested static class
        printer.printMessage();

        // In order to create instance of Inner class we need an Outer class
        // instance. Let us create Outer class instance for creating
        // non-static nested class
        OuterClass outer = new OuterClass();
        OuterClass.InnerClass inner  = outer.new InnerClass();

        // calling non-static method of Inner class
        inner.display();

        // we can also combine above steps in one step to create instance of
        // Inner class
        OuterClass.InnerClass innerObject = new OuterClass().new InnerClass();

        // similarly we can now call Inner class method
        innerObject.display();
    }
}
```

Run on IDE

Output:

```
Message from nested static class: GeeksForGeeks

Message from non-static nested class: GeeksForGeeks

Message from non-static nested class: GeeksForGeeks
```

Reference Book:
Introduction To Java Programming by Y. DANIEL LIANG