

Java static keyword

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

The static can be:

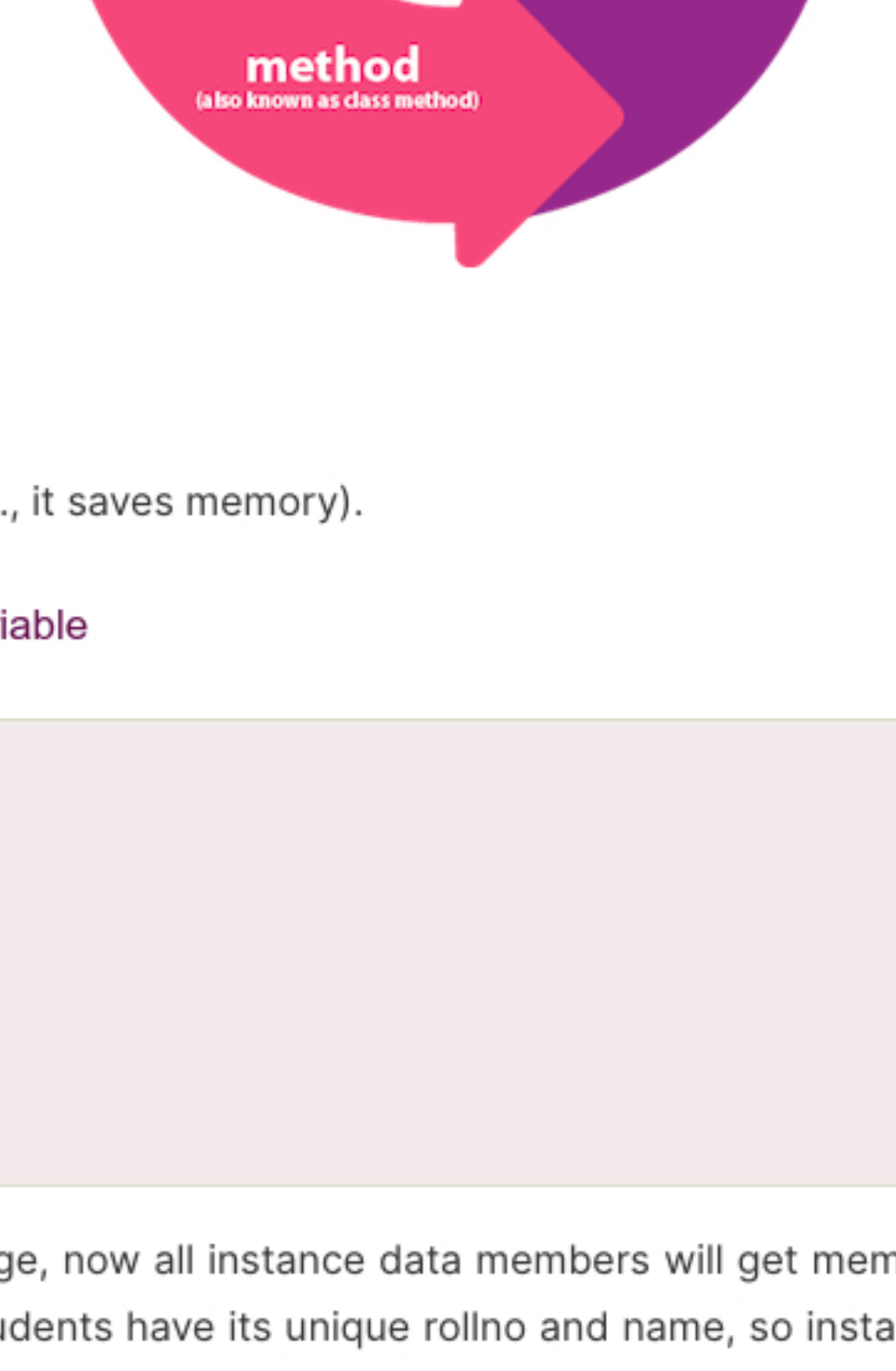
1. Variable (also known as a class variable)
2. Method (also known as a class method)
3. Block
4. Nested class

- » Static variable
- » Program of the counter without static variable
- » Program of the counter with static variable
- » Static method
- » Restrictions for the static method
- » Why is the main method static?
- » Static block
- » Can we execute a program without main method?

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.



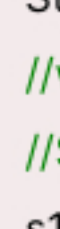
Advantages of static variable

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

Understanding the problem without static variable

```
class Student{
    int rollno;
    String name;
    String college="ITS";
}
```


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

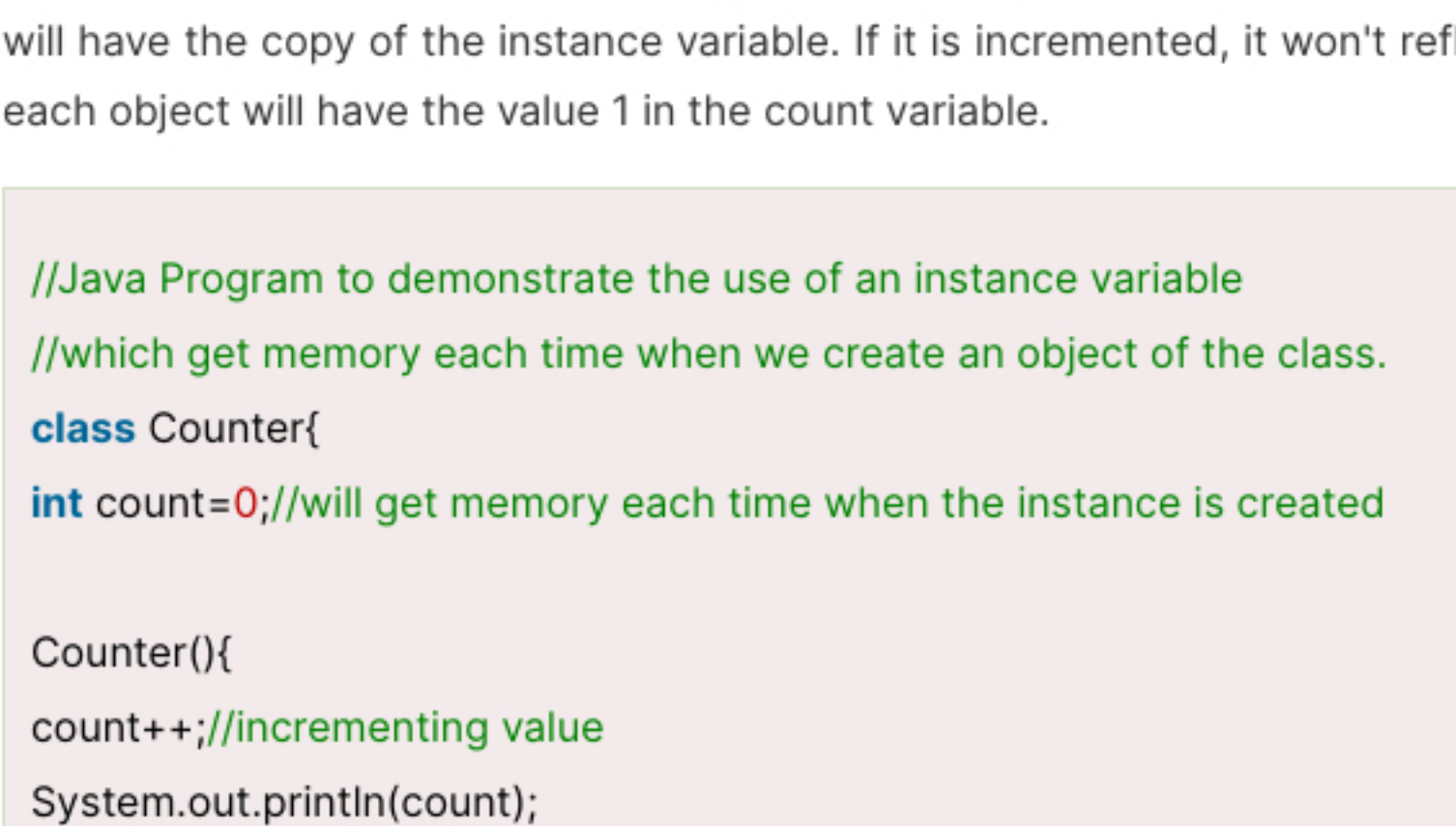
```
//Java Program to demonstrate the use of static variable
class Student{
    int rollno;//instance variable
    String name;
    static String college ="ITS";//static variable
    //constructor
    Student(int r, String n){
        rollno = r;
        name = n;
    }
    //method to display the values
    void display (){System.out.println(rollno+" "+name+" "+college);}
}

//Test class to show the values of objects
public class TestStaticVariable1{
    public static void main(String args[]){
        Student s1 = new Student(111,"Karan");
        Student s2 = new Student(222,"Aryan");
        //we can change the college of all objects by the single line of code
        //Student.college="BBDIT";
        s1.display();
        s2.display();
    }
}
```

 **Test It Now**

Output:

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



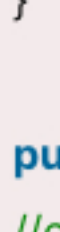
Program of the counter without static variable

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.

```
//Java Program to demonstrate the use of an instance variable
//which get memory each time when we create an object of the class.
class Counter{
    int count=0;//will get memory each time when the instance is created

    Counter(){
        count++;//incrementing value
        System.out.println(count);
    }

    public static void main(String args[]){
        //Creating objects
        Counter c1=new Counter();
        Counter c2=new Counter();
        Counter c3=new Counter();
    }
}
```

 **Test It Now**

Output:

```
1
1
1
```

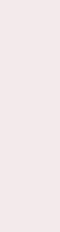
Program of counter by static variable

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.

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

    Counter2(){
        count++;//incrementing the value of static variable
        System.out.println(count);
    }

    public static void main(String args[]){
        //creating objects
        Counter2 c1=new Counter2();
        Counter2 c2=new Counter2();
        Counter2 c3=new Counter2();
    }
}
```

 **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

```
//Java Program to demonstrate the use of a static method.
class Student{
    int rollno;
    String name;
    static String college = "ITS";
    //static method to change the value of static variable
    static void change(){
        college = "BBDIT";
    }
    //constructor to initialize the variable
    Student(int r, String n){
        rollno = r;
        name = n;
    }
    //method to display values
    void display(){System.out.println(rollno+" "+name+" "+college);}
}

//Test class to create and display the values of object
public class TestStaticMethod{
    public static void main(String args[]){
        Student.change();//calling change method
        //creating objects
        Student s1 = new Student(111,"Karan");
        Student s2 = new Student(222,"Aryan");
        Student s3 = new Student(333,"Sonoo");
        //calling display method
        s1.display();
        s2.display();
        s3.display();
    }
}
```


 **Test It Now**

Output:111 Karan BBDIT
222 Aryan BBDIT
333 Sonoo BBDIT

Another example of a static method that performs a normal calculation

```
//Java Program to get the cube of a given number using the static method
class Calculate{
    static int cube(int x){
        return x*x*x;
    }

    public static void main(String args[]){
        int result=Calculate.cube(5);
        System.out.println(result);
    }
}
```

 **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.

```
class A{
    int a=40;//non static

    public static void main(String args[]){
        System.out.println(a);
    }
}
```

 **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

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

 Test It Now

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**.

```
class A3{
    static{
        System.out.println("static block is invoked");
        System.exit(0);
    }
}
```

 Test It Now

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
```