

The ? Operator /Java Ternary Operator

Java Ternary operator is used as one liner replacement for if-then-else statement and used a lot in java programming. It is the only conditional operator which takes three operands.

Java Ternary Operator Example

```
class OperatorExample{  
public static void main(String args[]){  
int a=2;  
int b=5;  
int min=(a<b)?a:b;  
System.out.println(min);  
}}
```

Output:

2

Another Example:

```
class OperatorExample{  
public static void main(String args[]){  
int a=10;  
int b=5;  
int min=(a<b)?a:b;  
System.out.println(min);  
}}
```

Output:

5

Break and Continue Statement in Java

Java Break Statement

When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

The Java *break* is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

We can use Java break statement in all types of loops such as for loop, while loop and do-while loop.

Syntax:

```
jump-statement;  
break;
```

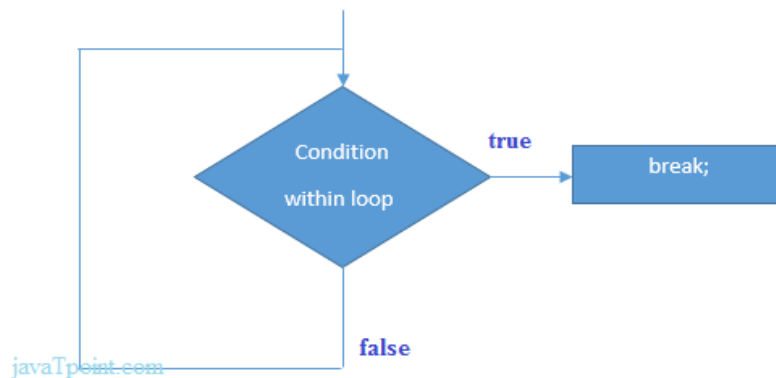


Figure: Flowchart of break statement

Java Break Statement with Loop

Example:

```
//Java Program to demonstrate the use of break statement  
//inside the for loop.  
public class BreakExample {  
  public static void main(String[] args) {  
    //using for loop  
    for(int i=1;i<=10;i++){  
      if(i==5){  
        //breaking the loop  
        break;  
      }  
      System.out.println(i);  
    }  
  }  
}
```

Output:

```
1  
2  
3  
4
```

```
//Java Program to illustrate the use of break statement
//inside an inner loop
public class BreakExample2 {
    public static void main(String[] args) {
        //outer loop
        for(int i=1;i<=3;i++){
            //inner loop
            for(int j=1;j<=3;j++){
                if(i==2&&j==2){
                    //using break statement inside the inner loop
                    break;
                }
                System.out.println(i+" "+j);
            }
        }
    }
}
```

Output:

```
1 1
1 2
1 3
2 1
3 1
3 2
3 3
```

Java Break Statement in while loop

Example:

```
//Java Program to demonstrate the use of break statement
//inside the while loop.
public class BreakWhileExample {
    public static void main(String[] args) {
        //while loop
        int i=1;
        while(i<=10){
            if(i==5){
                //using break statement
                i++;
                break;//it will break the loop
            }
            System.out.println(i);
            i++;
        }
    }
}
```

Output:

```
1
2
3
4
```

Java Break Statement in while loop

Example:

```
//Java Program to demonstrate the use of break statement
//inside the while loop.
public class BreakWhileExample {
    public static void main(String[] args) {
        //while loop
        int i=1;
        while(i<=10){
            if(i==5){
                //using break statement
                i++;
                break;//it will break the loop
            }
            System.out.println(i);
            i++;
        }
    }
}
```

Output:

```
1
2
3
4
```

Java Continue Statement

The continue statement is used in loop control structure when you need to jump to the next iteration of the loop immediately. It can be used with for loop or while loop.

The Java *continue statement* is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition. In case of an inner loop, it continues the inner loop only.

We can use Java continue statement in all types of loops such as for loop, while loop and do-while loop.

Syntax:

```
jump-statement;  
continue;
```

Java Continue Statement Example

Example:

```
//Java Program to demonstrate the use of continue statement  
//inside the for loop.  
public class ContinueExample {  
  public static void main(String[] args) {  
    //for loop  
    for(int i=1;i<=10;i++){  
      if(i==5){  
        //using continue statement  
        continue;//it will skip the rest statement  
      }  
      System.out.println(i);  
    }  
  }  
}
```

Output:

```
1  
2  
3  
4  
6  
7  
8  
9  
10
```

Java Continue Statement with Inner Loop

It continues inner loop only if you use the continue statement inside the inner loop.

Example:

```
//Java Program to illustrate the use of continue statement
//inside an inner loop
public class ContinueExample2 {
    public static void main(String[] args) {
        //outer loop
        for(int i=1;i<=3;i++){
            //inner loop
            for(int j=1;j<=3;j++){
                if(i==2&&j==2){
                    //using continue statement inside inner loop
                    continue;
                }
                System.out.println(i+" "+j);
            }
        }
    }
}
```

Output:

```
1 1
1 2
1 3
2 1
2 3
3 1
3 2
3 3
```

Java Continue Statement in while loop

Example:

```
//Java Program to demonstrate the use of continue statement
//inside the while loop.
public class ContinueWhileExample {
    public static void main(String[] args) {
        //while loop
        int i=1;
        while(i<=10){
            if(i==5){
                //using continue statement
                i++;
                continue;//it will skip the rest statement
            }
            System.out.println(i);
            i++;
        }
    }
}
```

Output:

```
1
2
3
4
6
7
8
9
10
```

Java Continue Statement in do-while loop

Example:

```
//Java Program to demonstrate the use of continue statement
//inside the Java do-while loop.
public class ContinueDoWhileExample {
    public static void main(String[] args) {
        //declaring variable
        int i=1;
        //do-while loop
        do{
            if(i==5){
                //using continue statement
                i++;
                continue;//it will skip the rest statement
            }
            System.out.println(i);
            i++;
        }while(i<=10);
    }
}
```

Output:

```
1
2
3
4
6
7
8
9
10
```


Final Keyword In Java

The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

1. variable
2. method
3. class

The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only. The blank final variable can be static also which will be initialized in the static block only. We will have detailed learning of these. Let's first learn the basics of final keyword.

1) Java final variable

If you make any variable as final, you cannot change the value of final variable(It will be constant).

Java Final Keyword

- ⇒ Stop Value Change
- ⇒ Stop Method Overriding
- ⇒ Stop Inheritance

javatpoint.com

Example of final variable

There is a final variable speedlimit, we are going to change the value of this variable, but It can't be changed because final variable once assigned a value can never be changed.

```
class Bike9{  
    final int speedlimit=90;//final variable  
    void run(){  
        speedlimit=400;  
    }  
    public static void main(String args[]){  
        Bike9 obj=new Bike9();  
        obj.run();  
    }  
}//end of class
```

✓ Test it Now

Output:Compile Time Error

2) Java final method

If you make any method as final, you cannot override it.

Example of final method

```
class Bike{
    final void run(){System.out.println("running");}
}

class Honda extends Bike{
    void run(){System.out.println("running safely with 100kmph");}

    public static void main(String args[]){
        Honda honda= new Honda();
        honda.run();
    }
}
```

✓ Test it Now

Output:Compile Time Error

2) Java final method

If you make any method as final, you cannot override it.

Example of final method

```
class Bike{
    final void run(){System.out.println("running");}
}

class Honda extends Bike{
    void run(){System.out.println("running safely with 100kmph");}

    public static void main(String args[]){
        Honda honda= new Honda();
        honda.run();
    }
}
```

✓ Test it Now

Output:Compile Time Error

Q) Is final method inherited?

Ans) Yes, final method is inherited but you cannot override it. For Example:

```
class Bike{
    final void run(){System.out.println("running...");}
}
class Honda2 extends Bike{
    public static void main(String args[]){
        new Honda2().run();
    }
}
```

 Test it Now

Output:running...

Q) What is blank or uninitialized final variable?

Que) Can we initialize blank final variable?

Yes, but only in constructor. For example:

```
class Bike10{
    final int speedlimit;//blank final variable

    Bike10(){
        speedlimit=70;
        System.out.println(speedlimit);
    }

    public static void main(String args[]){
        new Bike10();
    }
}
```

 Test it Now

Output: 70

Java Garbage Collection

In java, garbage means unreferenced objects.

Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.

To do so, we were using `free()` function in C language and `delete()` in C++. But, in java it is performed automatically. So, java provides better memory management.

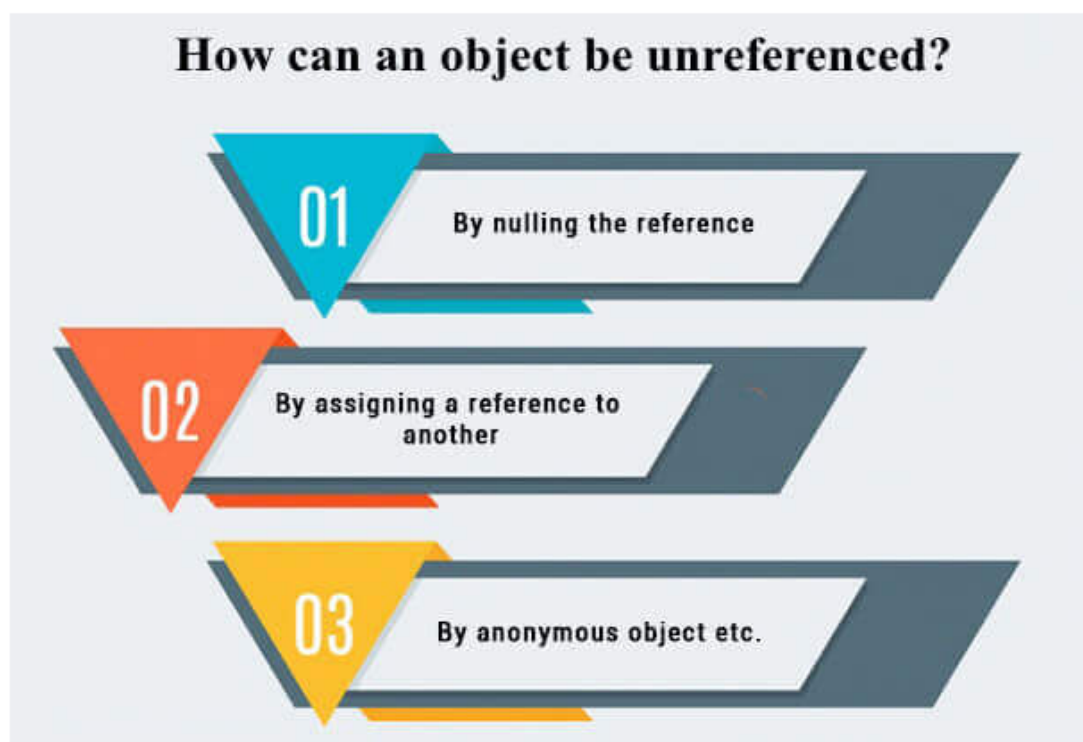
Advantage of Garbage Collection

- It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.
 - It is **automatically done** by the garbage collector(a part of JVM) so we don't need to make extra efforts.
-

How can an object be unreferenced?

There are many ways:

- By nulling the reference
- By assigning a reference to another
- By anonymous object etc.



1) By nulling a reference:

```
Employee e=new Employee();  
e=null;
```

2) By assigning a reference to another:

```
Employee e1=new Employee();  
Employee e2=new Employee();  
e1=e2;//now the first object referred by e1 is available for garbage collection
```

3) By anonymous object:

```
new Employee();
```

finalize() method

The finalize() method is invoked each time before the object is garbage collected. This method can be used to perform cleanup processing. This method is defined in Object class as:

```
protected void finalize(){}
```



Note: The Garbage collector of JVM collects only those objects that are created by new keyword. So if you have created any object without new, you can use finalize method to perform cleanup processing (destroying remaining objects).

gc() method

The gc() method is used to invoke the garbage collector to perform cleanup processing. The gc() is found in System and Runtime classes.

```
public static void gc(){}
```



Note: Garbage collection is performed by a daemon thread called Garbage Collector(GC). This thread calls the finalize() method before object is garbage collected.

Simple Example of garbage collection in java

```
public class TestGarbage1{  
    public void finalize(){System.out.println("object is garbage collected");}  
    public static void main(String args[]){  
        TestGarbage1 s1=new TestGarbage1();  
        TestGarbage1 s2=new TestGarbage1();  
        s1=null;  
        s2=null;  
        System.gc();  
    }  
}
```

 Test it Now

```
object is garbage collected  
object is garbage collected
```