**Reference:** [**https://www.javabykiran.com/interview/corejava/final-keyword-interview-questions.php**](https://www.javabykiran.com/interview/corejava/final-keyword-interview-questions.php)

* **What is the use of final keyword in java?**

• By using final keyword we can make  
• Final class  
• Final method  
• Final variables  
• If we declare any class as final we can not extend that class  
• If we declare any method as final it can not be overridden in sub class  
• If we declare any variable as final its value unchangeable once assigned

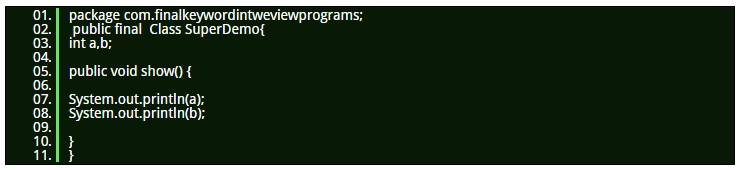
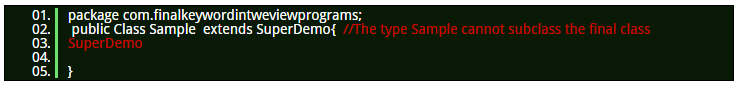
* **What is the main difference between abstract method and final method?**

Abstract methods must be overridden in sub class where as final methods can not be overridden in sub class

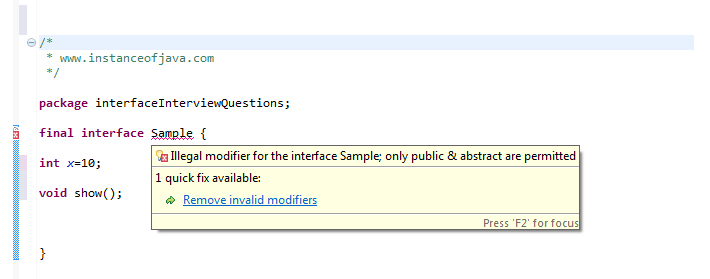
* **What is the actual use of final class in java?**

• If a class needs some security and it should not participate in inheritance in this scenario we need to use final class.   
• We can not extend final class.

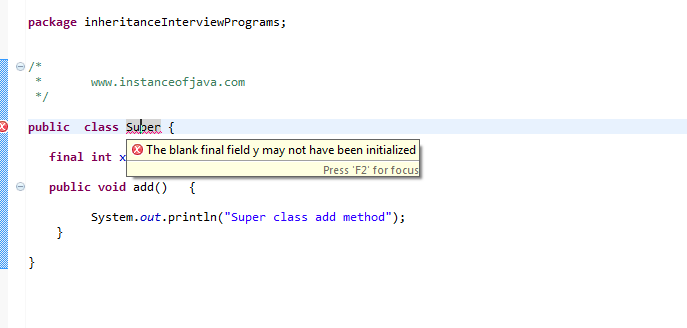
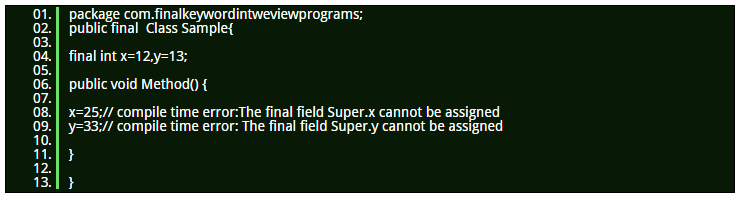
* **What will happen if we try to extend final class in java?**

• Compile time error will come.   
  
  


* **Can we declare interface as final?**

No We can not declare interface as final because interface should be implemented by some class so its not possible to declare interface as final.   


* **Is it possible to declare final variables without initialization?**

• No. Its not possible to declare a final variable without initial value assigned.  
• While declaring itself we need to initialize some value and that value can not be change at any time.   
  
  


* **Can we declare constructor as final?**

No.Constructors can not be final.

* **What will happen if we try to override final methods in sub classes?**

Compile time error will come :Cannot override the final method from Super class

* **Can we create object for final class?**

Yes we can create object for final class.

* **What is the most common predefined final class object you used in your code?**

String (for example)

* **What is the use of final keyword in java?**

final keyword in java is used to make any class or a method or a field as unchangeable. You can’t extend a final class, you can’t override a final method and you can’t change the value of a final field. final keyword is used to achieve high level of security while coding.

* **What is the blank final field?**

Uninitialized final field is called blank final field.

* **When do you override hashcode and equals() ?**

Whenever necessary especially if you want to do equality check or want to use your object as key in HashMap.

* **Can we change the state of an object to which a final reference variable is pointing?**

Yes, we can change the state of an object to which a final reference variable is pointing, but we can’t re-assign a new object to this final reference variable.

* **What is the main difference between abstract methods and final methods?**

Abstract methods must be overridden in the sub classes and final methods are not at all eligible for overriding.

* **What is the use of final class?**

A final class is very useful when you want a high level of security in your application. If you don’t want inheritance of a particular class, due to security reasons, then you can declare that class as a final..

* **Can we change the value of an interface field? If not, why?**

No, we can’t change the value of an interface field. Because interface fields, by default, are final and static. They remain constant for whole execution of a program..

* **Where all we can initialize a final non-static global variable if it is not initialized at the time of declaration?**

In all constructors or in any one of instance initialization blocks.

* **What are final class, final method and final variable?**

final class —> can not be extended.   
final method —> can not be overridden in the sub class.   
final variable —> can not change it’s value once it is initialized.

* **Where all we can initialize a final static global variable if it is not initialized at the time of declaration?**

In any one of static initialization blocks.

* **Can we use non-final local variables inside a local inner class?**

No. Only final local variables can be used inside a local inner class.

* **Can we declare constructors as final?**

No, constructors can not be final.

### Which access specifier can be used with a class ?

For an outer class we can only use 'public' and 'default' access specifiers. We can use 'private' access specifier for an inner class.

### Can we reduce the accessibility of an inherited or overridden method ?

No, we cannot reduce the accessibility of an inherited or overridden method.

### what is the difference between public, private, default and protected access specifiers ?

**• Private -** The code defined within a 'private' access specifier is not accessible outside the object scope.   
  
**• Public -**The code defined within a 'public' access specifier is accessible from anywhere.  
  
**• Default -**The code defined within a 'default' access specifier is accessible from anywhere within the same package.  
  
**• Protected -**The code defined within a 'default' access specifier is accessible by object and the sub class objects.

### Can we reduce the accessibility of the overridden method ?

No, we cannot reduce the accessibility of the overridden method.

### What will happen if we make the constructor 'private' ?

We won't be able to create the objects directly by invoking a new operator by making a constructor private.

### Can we instantiate the object of derived class if parent constructor is 'protected' ?

No, we cannot instantiate the object of derived class if parent constructor is protected.

### Can we declare an abstract method 'private' ?

No. An abstract method can only be declared 'protected' or 'public'.

### What is an access specifier ?

It is used to explicitly mention the way how the data (variables and methods of a class) will be available outside the scope.  
  
An access specifier is something which mentions the way how the member of a class will be made available to anything outside the class.  
  
An access specifier cannot be used with the local variables(i.e. present inside method/scope).  
  
Access specifiers are the keywords using which we can control the accessibility of the members of a class.

<https://www.javabykiran.com/interview/corejava/annotation-interview-questions.php>

* **What is ArrayStoreException in java? When will this exception occur ?**

ArrayStoreException is a runtime exception which occurs when you try to store non-compatible element in an array object. These type of the elements must be compatible with the type of array object. Say for instance, you can store only string elements in an array of strings. If we try to insert an integer element in an array of strings, we will get ArrayStoreException at runtime.   
  
public class MainClass  
{  
public static void main(String[] args)  
{  
Object[] stringArray = new String[5]; //No compile time error : String[] is auto-upcasted to Object[]  
stringArray[1] = "JAVA";  
stringArray[2] = 100; //No compile time error will occur, but this statement will throw java.lang.ArrayStoreException at runtime  
//This is because we are inserting an integer element into an array of string  
}  
}.

* **What are the drawbacks of the arrays in java?**

The main drawback of the arrays is that arrays are of fixed size. We cannot change the size of an array once it has been created. Therefore, we must know how many elements we want in an array before creating it. You cannot insert or delete the elements once an array has been created. Only the value of the elements can be changed.

* **What is an anonymous array? Give example.**

Anonymous array is an array without a reference. For example:  
  
public class MainClass  
{  
public static void main(String[] args)  
{  
//Creating anonymous arrays  
System.out.println(new int[]{1, 2, 3, 4, 5}.length); //Output : 5  
System.out.println(new int[]{21, 14, 65, 24, 21}[1]); //Output : 14  
}  
}

* **Can we pass a negative number as an array size ?**

No. We cannot pass a negative integer as an array size. If we do so, there will be no compile time error but NegativeArraySizeException will be prompted at runtime.   
  
public class MainClass  
{  
public static void main(String[] args)  
{  
int[] array = new int[-5]; //No compile time error  
//but you will get java.lang.NegativeArraySizeException at run time  
}  
}

* **Can you change the size of an array once it has been defined? OR Can you insert or delete the elements after creating an array?**

No. We cannot change the size of an array once it has been defined. We cannot insert or delete the elements after creating an array. Only the value of the elements can be changed.

* **What is the difference between int[] a and int a[ ] ?**

Both are the legal methods to declare the arrays in java.

* **“int a[] = new int[3]{1, 2, 3}” – is it a legal way of defining the arrays in java?**

No. We cannot mention the size of an array when we are providing the array contents.

* **There are two array objects of int type. One contains 100 elements and another one contains 10 elements. Can you assign array of 100 elements to an array of 10 elements?**

Yes, you can assign array of 100 elements to an array of 10 elements provided they should be of same type. While assigning, compiler only checks the type of the array and not the size.   
  
public class MainClass  
{  
public static void main(String[] args)  
{  
int[] a = new int[10];  
int[] b = new int[100];  
a = b; //Compiler checks only type, not the size  
}  
}

* **Differenciate between Array and ArrayList in java.**

|  |  |
| --- | --- |
| **ARRAY** | **ARRAYLIST** |
| **Arrays are of fixed length.** | ArrayList is of variable length. |
| **You cannot change the size of an  array once it has been created.** | Size of the ArrayList grows and shrinks  as you add or remove the elements. |
| **Array does not support generics.** | ArrayList supports generics. |
| **You can use arrays to store both  primitive types as well as reference types.** | You can store only reference types  in an ArrayList. |

* **How do you check the equality of two arrays in java? OR How can we compare the two arrays in java?**

You can use Arrays.equals() method to compare one dimensional arrays and to compare multi-dimensional arrays, use Arrays.deepEquals() method.

* **What are the different ways of copying an array into another array?**

There are four methods available in java to copy an array.  
  
1) Using 'for' loop.  
2) Using Arrays.copyOf() method.  
3) Using System.arraycopy() method.  
4) Using clone() method.

* **What are jagged arrays in java?**

Jagged arrays in java are the arrays containing arrays of different length. Jagged arrays are also multi-dimensional arrays also called as ragged arrays.

* **What is ArrayIndexOutOfBoundsException in java and when does it occur?**

ArrayIndexOutOfBoundsException is a runtime exception which occurs when your program tries to access invalid index of an array i.e negative index or index higher than the size of the array.

* **How can you sort the array elements?**

You can sort the array elements using Arrays.sort() method. This method internally uses quick sort algorithm to sort the array elements.   
  
import java.util.Arrays;  
public class MainClass  
{  
public static void main(String[] args)  
{  
int[] a = new int[]{45, 12, 78, 34, 89, 21};  
Arrays.sort(a);  
System.out.println(Arrays.toString(a));  
//Output : [12, 21, 34, 45, 78, 89]  
}  
}

* **How can you find the intersection of two arrays in java ?**

This is one of the most common java interview question asked to freshers as well as to experienced java professionals of 1 or 2 years. In this post, we will discuss couple of methods to find common elements between two arrays.   
  
1) Using iterative Method:   
In this method, we iterate both the given arrays and compare each element of one array with elements of other array. If the elements are found to be equal, we will add that element into HashSet. This method also works for those arrays which contain duplicate elements.   
  
2) Using retainAll() method :  
This is one of the easiest methods to find the common elements from two arrays. In this method, we create two HashSets using given two arrays and then use reatinAll() method of HashSet to retain only common elements from the two sets.

* **How do you find duplicate elements in an array ?**

This is one of the most asked java interview program for freshers. In this post, I have discussed two methods of finding duplicates in an array.   
  
1) Using Brute Force Method :  
In this method, we compare each element of an array with other elements. The performance of this method is very low if an array contains lots of elements. Therefore, this method is not recommended in real time.   
  
2) Using HashSet :  
HashSet contains only unique elements. HashSet never allows duplicate elements. We use this property of HashSet to find duplicates in an array. We try to add each element of an array into HashSet using add() method. This method will return true if an element is added successfully otherwise it returns false.   
  
Here is the java program which uses both the methods to find duplicate elements in an array.

* **What are the different ways of declaring multidimensional arrays in java?**

* **While creating the multidimensional arrays, can you specify an array dimension after an empty dimension?**

No. You cannot specify an array dimension after an empty dimension while creating multi-dimensional arrays. It gives compile time error.   
  
int[][][] a = new int[][5][]; //Compile time error   
  
int[][][] b = new int[5][][5]; //Compile time error   
  
int[][][] c = new int[][5][5]; //Compile time error

* **How do you search an array for a specific element ?**

You can search an array to check whether it contains the given element or not using Arrays.binarySearch() method. This method internally uses binary search algorithm to search for an element in an array.

* **What value does array elements get, if they are not initialized ?**

They get default values.

* **What are the different ways to iterate over an array in java ?**

1) Using normal 'for' loop  
public class MainClass  
{  
public static void main(String[] args)  
{  
int[] a = new int[]{45, 12, 78, 34, 89, 21};  
//Iterating over an array using normal for loop  
for (int i = 0; i < a.length; i++)  
{  
System.out.println(a[i]);  
}  
}  
}  
  
2) Using extended 'for' loop  
public class MainClass  
{  
public static void main(String[] args)  
{  
int[] a = new int[]{45, 12, 78, 34, 89, 21};  
//Iterating over an array using extended for loop  
for (int i : a)  
{  
System.out.println(i);  
}  
}  
}

* **How do you find second largest element in an array of integers ?**

This is also one of the java interview program asked many times to java freshers to check the candidate's logical ability and understanding of the language's fundamentals. While writing the program you should not use any sorting methods or any collection types. You should find the second largest number in the given array by iterating the array only once. These are the likely conditions interviewer may ask you to follow. In this article, we will try to find second largest number in an integer array using iterative method.

* **How do you find all pairs of elements in an array whose sum is equal to a given number?**

Given an array of integers, you have to find all pairs of elements in this array such that whose sum must be equal to a given number. For instance if {4, 5, 7, 11, 9, 13, 8, 12} is an array and 20 is the given number, then you have to find all pairs of elements in this array whose sum must be 20. In this example, (9, 11), (7, 13) and (8, 12) are such pairs whose sum is 20.

* **How do you separate zeroes from non-zeroes in an integer array ?**

Given an integer array, you have to separate all zero elements from non-zero elements. You have to move zeroes either to end of the array or bring them to beginning of the array. For example, if {14, 0, 5, 2, 0, 3, 0} is the given array, then moving zeros to end of the array will result {14, 5, 2, 3, 0, 0, 0} and bringing zeroes to front will result {0, 0, 0, 14, 5, 2, 3}. In this post, we will see both as how to move zeros to end and front of an array.

* **How do you find continuous sub array whose sum is equal to a given number?**

You have been given an integer array and a number. You need to find the continuous sub array of the given array whose sum is equal to given number. For example, If {12, 5, 31, 9, 21, 8} is the given array and 45 is the given number, then you have to find continuous sub array in this array such that whose elements add up to 45. In this case, {5, 31, 9} is such sub array whose elements add up to 45.

* **What is difference between fail-fast and fail-safe?**

Fail-safe iterators are just opposite to fail-fast.  
They never fail if you modify the underlying collection on which they are iterating, because they work on  
clone of Collection instead of original collection and that’s why they are called as fail-safe iterator.  
Iterator of CopyOnWriteArrayList is an example of fail-safe Iterator also iterator written by ConcurrentHashMap keySet is also fail-safe iterator and never throw ConcurrentModificationException..

* **Can we use any class as Map key?**

We can use any class as Map Key, however following points should be considered before using them.   
  
If the class overrides equals() method, it should also override hashCode() method. The class should follow the rules associated with equals() and hashCode() for all instances.   
  
If a class field is not used in equals(), you should not use it in hashCode() method. Best practice for user defined key class is to make it immutable, so that hashCode() value can be cached for fast performance.   
  
Also immutable classes make sure that hashCode() and equals() will not change in future that will solve any issue with mutability.   
  
For example,

let’s say I have a class MyKey that I am using for HashMap key. //MyKey name argument passed is used for equals() and hashCode()

MyKey key = new MyKey("Pankaj"); //assume hashCode=1234

myHashMap.put(key, "Value");

// Below code will change the key hashCode() and equals()

// but it's location is not changed.

key.setName("Amit"); //assume new hashCode=7890

//below will return null, because HashMap will try to look for key

//in the same index as it was stored but since key is mutated,

//there will be no match and it will return null.

myHashMap.get(new MyKey("Pankaj"));

This is the reason why String and Integer are mostly used as HashMap keys.

* **What are the differences between Queue and Stack in java?**

* **How can we make HashMap synchronized?**
* **What are IdentityHashMap and WeakHashMap?**

IdentityHashMap : IdentityHashMap is similar to HashMap except that it uses reference equality when comparing elements.  
IdentityHashMap class is not a widely used Map implementation.  
While this class implements the Map interface, it intentionally violates Map’s general contract, which

* **What is NavigableMap in Java ? What are its benefits over Map?**

NavigableMap Map was added in Java 1.6, it adds navigation capability to Map data structure. It provides methods like lowerKey() to get keys which is less than specified key, floorKey() to return keys which is less than or equal to specified key, ceilingKey() to get keys which is greater than or equal to specified key and higherKey() to return keys which is greater specified key from a Map. It also provide similar methods to get entries e.g. lowerEntry(), floorEntry(), ceilingEntry() and higherEntry().   
  
Apart from navigation methods, it also provides utilities to create sub-Map e.g. creating a Map from entries of an exsiting Map like tailMap, headMap and subMap. headMap() method returns a NavigableMap whose keys are less than specified, tailMap() returns a NavigableMap whose keys are greater than the specified and subMap() gives a NavigableMap between a range, specified by toKey to fromKey.

* **What is the Difference between the Iterator and ListIterator?**

Iterator : Iterator Can Only get Data From forward Direction .   
  
ListIterator : An iterator for lists that allows one to traverse the list in either direction.modify the list during iteration, and obtain the iterator’s current position in the list.   
  
A ListIterator has no current element. its cursor position always lies between the element that would be returned by a call to previous() and the element that would be returned by a call to next(). In a list of length n, there are n+1 valid index values, from 0 to n, inclusive.

* **Difference between Vector and ArrayList?**

Vector & ArrayList both classes are implemented using dynamically resizable arrays, providing fast random access and fast traversal. ArrayList and Vector class both implement the List interface.   
  
1) Synchronization - ArrayList is not thread-safe whereas Vector is thread-safe. In Vector class each method like add(), get(int i) is surrounded with a synchronized block and thus making Vector class thread-safe.   
  
2) Data growth - Internally, both the ArrayList and Vector hold onto their contents using an Array. When an element is inserted into an ArrayList or a Vector, the object will need to expand its internal array if it runs out of room. A Vector defaults to doubling the size of its array, while the ArrayList increases its array size by 50 percent.

* **Which two method you need to implement for key Object in HashMap ?**

In order to use any object as Key in HashMap, it must implements equals and hashcode method in Java. Read How HashMap works in Java for detailed explanation on how equals and hashcode method is used to put and get object from HashMap.

* **How to make a collection read only?**

Use following methods:   
  
Collections.unmodifiableList(list);  
Collections.unmodifiableSet(set);  
Collections.unmodifiableMap(map);  
These methods takes collection parameter and return a new read-only collection with same elements as in original collection.

* **What is BlockingQueue?**

A Queue that additionally supports operations that wait for the queue to become non-empty when  
retrieving an element, and wait for space to become available in the queue when storing an element.  
BlockingQueue methods come in four forms: one throws an exception, the second returns a special value  
(either null or false, depending on the operation), the third blocks the current thread indefinitely until the  
operation can succeed, and the fourth blocks for only a given maximum time limit before giving up.

* **When do you use ConcurrentHashMap in Java?**

This is another advanced level collection interview questions in Java which normally asked to check whether interviewer is familiar with optimization done on ConcurrentHashMap or not. ConcurrentHashMap is better suited for situation where you have multiple readers and one Writer or fewer writers since Map gets locked only during write operation. If you have equal number of reader and writer than ConcurrentHashMap will perform in line of Hashtable or synchronized HashMap.

* **Which implementation of the List interface provides for the fastest insertion of a new element into the middle of the list?**

We have three implementation of List : Vector, ArrayList, LinkedList. ArrayList and Vector both use an array to store the elements of the list. When an element is inserted into the middle of the list the elements that follow the insertion point must be

* **Which collection classes provide random access of it’s elements?**

ArrayList, HashMap, TreeMap, Hashtable classes provide random access to it’s elements.

* **What will happen if we put a key object in a HashMap which is already there ?**

This tricky Java questions is part of How HashMap works in Java, which is also a popular topic to create confusing and tricky question in Java. well if you put the same key again than it will replace the old mapping because HashMap doesn't allow duplicate keys.

* **What will be the problem if you don't override hashcode() method ?**
* **How to convert a string array to arraylist?**

new ArrayList(Arrays.asList(myArray));

* **How can you suppress unchecked warning in Java ?**

javac compiler for Java 5 generates unchecked warnings if you use combine raw types and generics types. You can be suppress those warnings by using @SuppressWarnings("unchecked") annotation.

* **What is the Properties class?**

The properties class is a subclass of Hashtable that can be read from or written to a stream. It also provides the capability to specify a set of default values to be used.

https://www.javabykiran.com/interview/corejava/collections-interview-questions-2.php