**1. How many types of Modifiers in Java?**

There are two types of Modifiers in Java:

         a) Access Modifiers

         b) Non- access Modifiers  
   
**2. What are the access modifiers in java?**  
Access Modifiers are the modifiers which are used to restrict the visibility of a class, field, method, and constructor.  
Java supports four access modifiers:  
**a) Private:** Private members of a class can be accessible only within the class. It cannot be accessed from outside the class.  
**b) Default:** Default members of a class are accessible within the same package due to visible only within the package. They cannot be accessed from outside the package.  
**c) Protected:** Protected members of a class are visible within the package. Therefore, we can only access within the package but can be accessed to the subclasses outside the package through the inheritance only.  
**d) Public:** Public members are visible anywhere. So, we can access anywhere within or outside the package.

**3. Can we have a private constructor in Java?**  
Yes, we can have a private constructor in Java.   
1. The private constructor is used when you do not want to create the object of that class.  
2. We cannot create a subclass of that class.  
3. It is also used in singleton design pattern, Factory method design pattern.  
  
**4. Which access specifier can be used with a class?**  
Public and Default access specifiers can be used with a class.  
  
**5. Can we instantiate the object of derived class if the parent constructor is protected?**  
No, we cannot instantiate the object of derived class if the parent constructor is protected.

**6. What are non-access modifiers in Java?**  
There are four non-access modifiers in Java. They are as follows:  
**a. Static:** This modifier is used to check that a member is a class member or instance member. If you declare a class as static, this class will be executed first.  
**b. Final:** Final is a keyword which is used to restrict the users. In other words, It is used to restrict further modification of a class, field or method. If a class is declared as ‘final’, the class cannot be subclassed.   
**c. Abstract:** Abstract is a keyword that is used with a class or a method. An abstract class or abstract method is used for further modification. If a class is declared as ‘abstract’, the class cannot be instantiated.  
**d. Synchronization:** It is used to achieve the thread safeness. Only one thread can enter in a synchronized method or block at a given time.

**7. Can we declare a top-level class as private?**  
No, we cannot declare a top-level or outer class as private. It can have either “public” or no modifier. If you declare a top-level class as a private, the compiler will complain that the “modifier private is not allowed here” but an inner class can be private. Inner class means class as a member of another class. Same is the case with protected.  
  
**8. Can we declare an abstract method as private?**  
No, an abstract method cannot be private. They must be declared as public, protected or default so that they can be further modified.  
  
**9. Can we declare a top-level class as protected?**  
No, we cannot declare a class as protected. An inner class can be protected but not outer class.  
  
**10. Can a method or a class be final and abstract at the same time?**  
No, it is not possible. A class or a method cannot be final or abstract at the same time because the final method or final class cannot be further modified whereas an abstract class or an abstract method must be modified further.  
  
**11. Why are access modifiers used?**  
The access modifiers are used to restrict the access of a class and its members. Access modifiers are used to reduce the visibility of the members of a class.  
  
**12. Which is the default access modifier?**  
‘Internal’ is the default access modifier if no access modifier is mentioned with a class or its members. Internal is a keyword which is used for declaration.  
  
**13. What is the default access modifier for Interface?**  
The public is the default access modifier for the interface. No other access modifier is allowed for them  
  
**14. Can we define struct members as protected?**  
No, we cannot define struct members as protected because struct does not support inheritance.  
  
**15. What is the default access specifier for a class, an interface, and struct declared directly with a namespace?**

Internal  
  
**16. What is the access modifier for enumeration?**  
Enumeration members are always public. No other access modifiers are allowed.  
  
**17. What is the role of private constructor in Java?**  
If you declare any constructor of a class as a private then you cannot create the object of a class from outside the class. In other words, we cannot create the subclass of that class.

**18. Which is the least restrictive access modifier in Java?**  
Public  
  
**19. Which is the most restrictive access modifier in Java?**  
Private  
  
**20. Which access modifier is also known as Universal access modifier?**  
Public  
  
**21. Explain visibility control in Java.**  
Visibility control in Java is implemented by the access specifier.

**Final words**  
We hope that this article has covered almost all the important ***Access modifiers interview questions*** and will help you in an interview. You always remember that when your concepts are clear then you may answer any questions.

# Java Access Modifiers - Private, Protected, Public, Static, Final - Interview Questions

Interview Questions on Access Modifiers can be very tricky. It is very important to prepare well. This article gives a quick refresher on all the important interview questions related to all important Access Modifiers.

## Interview Questions

1. [What is default class modifier?](http://www.javainterview.in/p/access-modifier-interview-questions.html#question1)
2. [What are the different method access modifiers?](http://www.javainterview.in/p/access-modifier-interview-questions.html#question2)
3. [What is the use of a final modifier on a class?](http://www.javainterview.in/p/access-modifier-interview-questions.html#question3)
4. [What is the use of a final modifier on a method?](http://www.javainterview.in/p/access-modifier-interview-questions.html#question4)
5. [What is a Final variable?](http://www.javainterview.in/p/access-modifier-interview-questions.html#question5)
6. [What is a final argument?](http://www.javainterview.in/p/access-modifier-interview-questions.html#question6)
7. [What happens when a variable is marked as volatile?](http://www.javainterview.in/p/access-modifier-interview-questions.html#question7)
8. [What is a Static Variable?](http://www.javainterview.in/p/access-modifier-interview-questions.html#question8)

### What is default class modifier?

* A class is called a Default Class is when there is no access modifier specified on a class.
* Default classes are visible inside the same package only.
* Default access is also called Package access.

##### **Example**

package com.rithus.classmodifiers.defaultaccess.a;

/\* No public before class. So this class has default access\*/

class DefaultAccessClass {

//Default access is also called package access

}

##### **Another Class in Same Package: Has access to default class**

package com.rithus.classmodifiers.defaultaccess.a;

public class AnotherClassInSamePackage {

//DefaultAccessClass and AnotherClassInSamePackage

//are in same package.

//So, DefaultAccessClass is visible.

//An instance of the class can be created.

DefaultAccessClass defaultAccess;

}

##### **Class in Different Package: NO access to default class**

package com.rithus.classmodifiers.defaultaccess.b;

public class ClassInDifferentPackage {

//Class DefaultAccessClass and Class ClassInDifferentPackage

//are in different packages (\*.a and \*.b)

//So, DefaultAccessClass is not visible to ClassInDifferentPackage

//Below line of code will cause compilation error if uncommented

//DefaultAccessClass defaultAccess; //COMPILE ERROR!!

}

### What are the different method access modifiers?

Let’s discuss about access modifiers in order of increasing access.

##### **private**

a. Private variables and methods can be accessed only in the class they are declared.

b. Private variables and methods from SuperClass are NOT available in SubClass.

##### **default or package**

a. Default variables and methods can be accessed in the same package Classes.

b. Default variables and methods from SuperClass are available only to SubClasses in same package.

##### **protected**

a. Protected variables and methods can be accessed in the same package Classes.

b. Protected variables and methods from SuperClass are available to SubClass in any package

##### **public**

a. Public variables and methods can be accessed from every other Java classes.

b. Public variables and methods from SuperClass are all available directly in the SubClass

### What is the use of a final modifier on a class?

Final class cannot be extended. Example of Final class in Java is the String class. Final is used very rarely as it prevents re-use of the class.Consider the class below which is declared as final.

final public class FinalClass {

}

Below class will not compile if uncommented. FinalClass cannot be extended.

/\*

class ExtendingFinalClass extends FinalClass{ //COMPILER ERROR

}

\*/

### What is the use of a final modifier on a method?

Final methods cannot be overridden. Consider the class FinalMemberModifiersExample with method finalMethod which is declared as final.

public class FinalMemberModifiersExample {

final void finalMethod(){

}

}

Any SubClass extending above class cannot override the finalMethod().

class SubClass extends FinalMemberModifiersExample {

//final method cannot be over-riddent

//Below method, uncommented, causes compilation Error

/\*

final void finalMethod(){

}

\*/

}

### What is a Final variable?

Once initialized, the value of a final variable cannot be changed.

final int finalValue = 5;

//finalValue = 10; //COMPILER ERROR

### What is a final argument?

Final arguments value cannot be modified. Consider the example below:

void testMethod(final int finalArgument){

//final argument cannot be modified

//Below line, uncommented, causes compilation Error

//finalArgument = 5;//COMPILER ERROR

}

### What happens when a variable is marked as volatile?

* Volatile can only be applied to instance variables.
* A volatile variable is one whose value is always written to and read from "main memory". Each thread has its own cache in Java. The volatile variable will not be stored on a Thread cache.

### What is a Static Variable?

Static variables and methods are class level variables and methods. There is only one copy of the static variable for the entire Class. Each instance of the Class (object) will NOT have a unique copy of a static variable. Let’s start with a real world example of a Class with static variable and methods.

##### **Static Variable/Method – Example**

count variable in Cricketer class is static. The method to get the count value getCount() is also a static method.

public class Cricketer {

private static int count;

public Cricketer() {

count++;

}

static int getCount() {

return count;

}

public static void main(String[] args) {

Cricketer cricketer1 = new Cricketer();

Cricketer cricketer2 = new Cricketer();

Cricketer cricketer3 = new Cricketer();

Cricketer cricketer4 = new Cricketer();

System.out.println(Cricketer.getCount());//4

}

}

4 instances of the Cricketer class are created. Variable count is incremented with every instance created in the constructor.

**What are the different types of Access Modifiers?**  
**Ans:** The following are the types of Access Modifiers:

* **Public:** Anything declared as public can be accessed from anywhere.
* **Private:** Anything declared as private can’t be seen outside of its class.
* **Protected:** Anything declared as protected can be accessed by classes in the same package and subclasses in the other packages.
* **Default Modifier:** Can be accessed only to classes in the same package.

**State the significance of public, private, protected, default modifiers both singly and in combination and state the effect of package relationships on declared items qualified by these modifiers.  
Ans:**

* **Public:** Public class is visible in other packages, the field is visible everywhere (class must be public too).
* **Private:** Private variables or methods may be used only by an instance of the same class that declares the variable or method; A private feature may only be accessed by the class that owns the feature.
* **Protected:** Is available to all classes in the same package and also available to all subclasses of the class that owns the protected feature. This access is provided even to subclasses that reside in a different package from the class that owns the protected feature.

**What you get by default, i.e. without any access modifier (i.e. public-private or protected).**  
**Ans:** It means that it is visible to all within a particular package.

**Can a top-level class be private or protected?**  
**Ans:** No. A top-level class cannot be private or protected. It can have either “public” or no modifier. If it does not have a modifier, it is supposed to have a default access. If a top-level class is declared as private, the compiler will complain that the “modifier private is not allowed here”. This means that a top-level class cannot be private. Same is the case with protected.

**What type of parameter passing does Java support?**  
**Ans:** In Java, the arguments are always passed by value.

**Primitive data types are passed by reference or pass by value?**  
**Ans:** Primitive data types are passed by value.

**Are objects passed by value or by reference?**  
**Ans:** Java only supports pass by value. With objects, the object reference itself is passed by value and so both the original reference and parameter copy both refer to the same object.

**Can a class be declared as protected?**  
**Ans:** A class can’t be declared as protected. Only methods can be declared as protected.

**What is the access scope of a protected method?**  
**Ans:** A protected method can be accessed by the classes within the same package or by the subclasses of the class in any package.

**If a variable is declared as private, where may the variable be accessed?**  
**Ans:** A private variable may only be accessed within the class in which it is declared.

**What do you understand by private, protected and public?**  
**Ans:** These are accessibility modifiers. Private is the most restrictive, while the public is the least restrictive. There is no real difference between protected and the default type (also known as package protected) within the context of the same package; however, the protected keyword allows visibility to a derived class in a different package.

**What modifiers may be used with an inner class that is a member of an outer class?**  
**Ans:** A (non-local) inner class may be declared as public, protected, private, static, final, or abstract.

**If a class is declared without any access modifiers, where may the class be accessed?**  
**Ans:** A class that is declared without any access modifiers is said to have package access. This means that the class can only be accessed by other classes and interfaces that are defined within the same package.

**If a method is declared as protected, where may the method be accessed?**  
**Ans:** A protected method may only be accessed by classes or[**interfaces**](https://www.softwaretestingo.com/interface-in-java/) of the same package or by subclasses of the class in which it is declared.

**What is the difference between a public and a non-public class?**  
**Ans:** A public class may be accessed outside of its package. A non-public class may not be accessed outside of its package.

**What modifiers may be used with a top-level class?**  
**Ans:** A top-level class may be public, abstract, or final.

**What modifiers may be used with an inner class that is a member of an outer class?**  
**Ans:** A (non-local) inner class may be declared as public, protected, private, static, final, or abstract.

**What modifiers may be used with a top-level class?**  
**Ans:** A top-level class may be public, abstract, or final.

If we have missed something, then you can inform us by commenting on the comment section.

**1) What is the use of access modifiers in Java?**

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse1)

**Answer :**  
Access Modifiers in Java are used to control the visibility of fields, methods, classes and constructors.

**2) Can you create a sub class to the following class?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | class A  {      private A()      {          //First Constructor      }        private A(int i)      {          //Second Constructor      }  } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse2)

**Answer :**  
No, you can’t create sub classes to that class which has only private constructors.

**3) Can you find out the error in the below code?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | private class A  {      private class B      {          //Inner class      }  } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse3)

**Answer :**  
Inner classes can be private, but outer classes can not be private.

**4) Does field ‘i’ of Class A be inherited to Class B in the below code?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class A  {      protected int i;  }    class B extends A  {    } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse4)

**Answer :**  
Yes, protected members of a class are inherited to any sub class.

**5) Is the below code written correctly?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | class A  {      private class B      {          //inner class      }  }    public class MainClass extends A  {      public static void main(String[] args)      {          B b = new B();      }  } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse5)

**Answer :**  
No. private inner Class B can not be instantiated outside the Class A.

**6) Is the below code written correctly?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package pack1;    class A  {    }    package pack2;    class B extends A  {    } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse6)

**Answer :**  
No. Class with default (no) access modifiers can not have subclass outside the package.

**7) Can we declare a class as protected?**

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse7)

**Answer :**  
Yes, but only inner class. Outer class can not be protected.

**8) Do you think the below program is written correctly? If yes, what will be the output?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | package pack1;    class X  {      protected int i = 1221;        void methodOfX()      {          System.out.println(i);      }  }    public class MainClass  {      public static void main(String[] args)      {          X x = new X();            System.out.println(x.i);            x.methodOfX();      }  } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse8)

**Answer :**  
Yes, it is written correctly. Output will be  
1221  
1221

**9) Why we can’t instantiate Class-A in the below code outside the package even though it has public constructor?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | package pack1;    class A  {      public A()      {          //public constructor      }  }    package pack2;    import pack1.\*;    class B  {      A a = new A();       //Compile Time Error  } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse9)

**Answer :**  
Because, Class-A itself has been defined with default access modifier. That means Class-A can be instantiated within the package in which it is defined. It can not be instantiated outside the package, even though it has public constructor.

**10) Can a protected field of a class be inherited to subclass outside the package?**

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse10)

**Answer :**  
Yes, protected members of a class are inherited to sub classes outside the package.

**11) Why Line 17 in the below code is throwing compile time error?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | package pack1;    public class A  {      protected A()      {          //protected constructor      }  }    package pack2;    import pack1.A;    class B  {      A a = new A();  } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse11)

**Answer :**  
Because, we can’t instantiate a class outside the package which has only protected constructors.

**12) Do you think the below code compiles successfully even though it is calling super class’s protected constructor outside the package?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | package pack1;    public class A  {      protected A(int i)      {          //protected constructor      }  }    package pack2;    import pack1.A;    class B extends A  {      public B()      {          super(10);     //calling super class's protected constructor      }  } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse12)

**Answer :**  
Yes, above code will compile successfully.

**13) Can we declare static methods as private?**

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse13)

**Answer :**  
Yes, static members of a class can be private.

**14) Is the below code written correctly? If yes, what will be the output?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | package pack1;    class A  {      protected static String s = "A";  }    class B extends A  {    }    class C extends B  {      static void methodOfC()      {          System.out.println(s);      }  }    public class MainClass  {      public static void main(String[] args)      {          C.methodOfC();      }  } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse14)

**Answer :**  
Yes, it is written correctly. Output will be A.

**15) Write the access modifiers in the increasing order of their visibility?**

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse15)

**Answer :**  
private —> default or no access modifiers —> protected —> public

**16) How many public classes a .java file can have?**

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse16)

**Answer :**  
Only one. A .java file can have maximum one public class.

**17) What will be the outcome of the below program?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | package pack1;    public class A  {      private int methodOne(int i)      {          return ++i;      }        public int methodTwo(int i)      {          return methodOne(++i);      }  }    package pack2;    import pack1.A;    class B extends A  {      int methodOne(int i)      {          return methodTwo(++i);      }  }    public class MainClass  {      public static void main(String[] args)      {          System.out.println(new B().methodOne(101));      }  } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse17)

**Answer :**  
104

**18) Can you find out the error in the following code snippet?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | class A  {      public void methodOfA()      {          System.out.println("Class A");      }  }    class B extends A  {      @Override      void methodOfA()      {          System.out.println("Class B");      }  } |

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse18)

**Answer :**  
The visibility of methodOfA() has been reduced to default while overriding it in the class B. You can’t reduce the visibility of a method while overriding it.

**19) private method can be overridden as public method. True or False?**

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse19)

**Answer :**  
False. private methods are not at all inherited.

**20) A method of super class with default access modifier can be overridden as protected or public but not as private. True or false?**

[View Answer](https://javaconceptoftheday.com/java-practice-questions-on-access-modifiers/#collapse20)

**Answer :**  
True.

**21) Monu has written the code like below but it is showing compile time error. Can you help him to remove the error?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | private class A  {      private class B      {          private class C          {            }      }  } |

**1) How many types of modifiers are there in Java.?**

Two types of modifiers are there in java. They are,

a) Access Modifiers

b) Non-access Modifiers

**2) What are access modifiers in java.?**

These are the modifiers which are used to restrict the visibility of a class or a field or a method or a constructor. Java supports 4 access modifiers.

**a) private** : private fields or methods or constructors are visible within the class in which they are defined.

**b) protected** : Protected members of a class are visible within the package but they can be inherited to sub classes outside the package.

**c) public :** public members are visible everywhere.

**d) default or No-access modifiers :** Members of a class which are defined with no access modifiers are visible within the package in which they are defined.

(For more info on access modifiers, [click here](https://javaconceptoftheday.com/access-modifiers-in-java/).)

**3) What are non-access modifiers in java.?**

These are the modifiers which are used to achieve other functionalities like,

**a) static :** This modifier is used to specify whether a member is a class member or an instance member.

**b) final :** It is used to restrict the further modification of a class or a method or a field. (for more on final, [click here](https://javaconceptoftheday.com/final-keyword-in-java/)).

**c) abstract :** abstract class or abstract method must be enhanced or modified further. (For more on abstract,  [click here](https://javaconceptoftheday.com/abstraction-in-java/" \o "Abstraction In Java" \t "_blank)).

**d) synchronized :** It is used to achieve thread safeness. Only one thread can execute a method or a block which is declared as synchronized at any given time. (for more on synchronized, [click here](https://javaconceptoftheday.com/synchronization-in-java/).)

**4) Can we use a field or a method declared without access modifiers outside the package.?**

No, we can’t use a field or a method with no-access (default) specifiers outside the package in which their class is defined.

**5) Can a method or a class be final and abstract at the same time.?**

No, it is not possible. A class or a method can not be final and abstract at the same time. final and abstract are totally opposite in nature. final class or final method must not be modified further where as abstract class or abstract method must be modified further.

**6) Can we declare a class as private.?**

We can’t declare an outer class as private. But, we can declare an inner class (class as a member of another class) as private.

**7) Can we declare an abstract method as private also.?**

No, abstract methods can not be private. They must be public or protected or default so that they can be modified further.

**8) Can we declare a class as protected.?**

We can’t declare an outer class as protected. But, we can declare an inner class (class as a member of another class) as protected.

**9) A class can not be declared with synchronized keyword. Then, why we call classes like Vector, StringBuffer are synchronized classes.?**

Any classes which have only synchronized methods and blocks are treated as synchronized classes. Classes like Vector, StringBuffer have only synchronized methods. That’s why they are called as synchronized classes.