Java Interview Question and Answers

➢ Topic – Java Interface:

➢ **Java Interface Interview Questions and Programming with Answers**

➢ **1. What is an interface in Java?**

➢ Ans: An interface in Java is a mechanism that is used to achieve complete abstraction. It is basically a kind of class that contains only constants and abstract methods.

➢ **2. Can we define private and protected modifiers for data members (fields) in interfaces?**

➢ Ans: No, we cannot define private and protected modifiers for variables in interface because the fields (data members) declared in an interface are by default public, static, and final.

➢ **3. Which modifiers are allowed for methods in an Interface?**

➢ Ans: Only abstract and public modifiers are allowed for methods in interfaces.

➢ **4. Suppose A is an interface. Can we create an object using new A()?**

➢ Ans: No, we cannot create an object of interface using new operator. But we can create a reference of interface type and interface reference refers to objects of its implementation classes.

➢ **5. Can we define an interface with a static modifier?**

➢ Ans: Yes, from Java 8 onwards, we can define static and default methods in an interface. Prior to Java 8, it was not allowed.

➢ **6. Suppose A is an interface. Can we declare a reference variable a with type A like this: A a;**

➢ Ans: Yes.

➢ **7. Can an interface extends another interface in Java?**

➢ Ans: Yes, an interface can extend another interface.

➢ **8. Can an interface implement another interface?**

➢ Ans: No, an interface cannot implement another interface.

➢ **9. Is it possible to define a class inside an interface?**

➢ Ans: Yes, we can define a class inside an interface.

➢

10. Which of the following is a correct representation of interface?

```
➢ a) public abstract interface A {
➢    abstract void m1() {};
➢   }
➢ b) public abstract interface A {
➢    void m1();
➢   }
➢ c) abstract interface A extends B, C {
➢     void m1();
➢   }
➢ d) abstract interface A extends B, C {
➢    void m1(){};
➢   }
➢ e) abstract interface A {
➢    m1();
➢   }
➢ f) interface A {
➢    void m1();
➢   }
➢ g) interface A {
➢    int m1();
➢   }
➢ h) public interface A {
➢    void m1();
➢   }
➢   public class B implements A {
➢   }
➢ i) public interface A {
➢     void m1();
➢   }
➢   Public interface B {
➢     void m1();
➢   }
➢   public interface C extends A, B {
➢    void m1();
➢   }
```

➢ Ans: b, c, f, g, i.

➢ **11. Identify the error in the following code.**

➢ a)

```
➢ interface A {
➢  void m1();
➢ }
```

```
public class B implements A {
void m1(){
 System.out.println("One");
 }
}
```

- Ans: We cannot reduce the visibility of inherited method from interface A.
- b)

```
interface A {
 A() { }
 void m1();
}
public abstract class B implements A {
public void m1(){
 System.out.println("One");
 }
}
```

- Ans: An interface cannot have a constructor.

c)

```
interface A {
 int x;
 void m1();
}
public class B implements A {
 int x = 20;
public void m1(){
 System.out.println("One");
 }
}
```

- Ans: A variable in an interface must be initialized at the time of declaration.
- **12. What will be the output of the following program?**

```
interface A {
 int x = 10;
 void m1();
}
public class B implements A {
int x = 20;
public void m1(){
 System.out.println("One");
 }
}
public class Test {
```

```
public static void main(String[] args){
 A a = new B();
  a.m1();
  System.out.println(a.x);
  }
}
```

- Ans: Output: One, 10.
- **13. Can an interface extend multiple interfaces?**
- Ans: Yes, an interface can extend multiple interfaces.
- **14. Can an interface has instance and static blocks?**
- Ans: No.
- **15. What will be the output of the following program?**

```
interface A {
 int x = 10;
}
interface B {
 int x = 20;
}
interface C extends A, B{
 int x = 30;
public static void main(String[] args){
 int a = A.x;
 int b = B.x;
 int c = C.x;

 System.out.println(a);
 System.out.println(b);
 System.out.println(c);
 }
}
```

- Ans: Output: 10, 20, 30
- **16. What happens if a class has implemented an interface but has not provided implementation for that method defined in Interface?**
- Ans: The class has to be declared with an abstract modifier. This will be enforced by the Java compiler.
- **17. Why an Interface method cannot be declared as final in Java?** Or, Can a method within an interface be marked as final?
- Ans: Not possible. Doing so will result the compilation error problem. This is because a final method cannot be overridden in java. But an interface method should be implemented by another class.

- So, the interface method cannot be declared as final. The modifiers such as public and abstract are only applicable for method declaration in an interface.
- **18. Can an interface be final?**
- Ans: No. Doing so will result compilation error problem.
- **19. Why an interface cannot have a constructor?**
- Ans: Inside an interface, a constructor cannot be called using super keyword with hierarchy.
- **20. Why an Interface can extend more than one Interface but a Class can't extend more than one Class?**
- Ans: We know that Java doesn't allow multiple inheritance because a class extends only one class. But an Interface is a pure abstraction model. It does not have inheritance hierarchy like classes.
- Therefore, an interface allows to extend more than one Interface.
- **21. What is the use of interface in Java?**
  Or, why do we use an interface in Java?
- Ans: There are many reasons to use interface in java. They are as follows:
- a. An interface is used to achieve fully abstraction.
  b. Using interfaces is the best way to expose our project's API to some other project.
  c. Programmers use interface to customize features of software differently for different objects.
  d. By using interface, we can achieve the functionality of multiple inheritance.
- **22. Is it necessary to implement all abstract methods of an interface?**
- Ans: Yes, all the abstract methods defined in interface must be implemented.
- **23. Can we define a variable in an interface? What type it should be?**
- Ans: Yes, we can define variable in an interface that must be implicitly static and final.
- **24. Can we re-assign a value to a variable of interface?**
- Ans: No, variables defined inside the interface are static and final by default. They are just like constants. We can't change their value once they got.

- **25. What is the difference between abstract class and interface in Java?**
- Ans: Refer to this tutorial: 12 Difference between Abstract class and Interface in Java
- **26. What is the difference between class and interface in Java?**
- Ans: Refer to this tutorial: Difference between Class and Interface in Java
- **27. What is a Marker Interface in Java?**
- Ans: An Interface that doesn't have any data members or methods is called marker interface in java. For example, Serializable, Cloneable, Remote, etc.
- **28. What is a Nested interface?**
- Ans: An interface declared inside another interface is called nested interface. By default, it is static in nature. It is also known as static interface.
- **29. Can we reduce the visibility of interface method while overriding?**
- Ans: No, while overriding any interface methods, we must use public only. This is because all interface methods are public by default. We cannot reduce the visibility while overriding them.
- **30. Can we define an interface inside a method as local member?**
- Ans: No, we can't define an interface as local member of a method like local inner class.
- **31. Will the code compiled successfully? If yes, what will be the output?**

```
interface A {
 void m1(int x, double y);
 void m2(String z);
}
class B implements A {
public void m1(int x, double y) {
  System.out.println("Hello");
 }
 public void m2(String z){
  System.out.println("Java");
 }
 void m3() {
  System.out.println("Hello Java!");
 }
}
public class Test {
```

```
public static void main(String[] args){
 B b = new B();
 b.m1(20, 10.0);
 b.m2(null);
 b.m3();
 }
}
```

- Ans: Yes, the code will be compiled successfully. The output is Hello, Java, Hello Java!.

- **32. What will be the output of the following programs?**
- a)

```
interface A {

 void m1(int x, double y);

}

abstract class B implements A {

public void m1(double x, int y){

 System.out.println("One");

 }

}

public class Test extends B {

public void m1(double x, int y){

 System.out.println("Two");

 super.m1(20.5, 20);

}

public void m1(int x, double y){

  System.out.println("Three");
```

```
➢ }
➢ public static void main(String[] args){
➢  Test t = new Test();
➢  t.m1(20.5, 10);
➢  t.m1(10, 5.5);
➢  }
➢ }
```

➢ Ans: Output: Two, One, Three

➢ b)

```
➢ interface A {
➢  void m1(A a);
➢ }
➢ abstract class B implements A {
➢  void m1(B b){
➢  System.out.println("One");
➢  }
➢ }
➢ public class Test extends B {
➢ public void m1(A a){
➢  System.out.println("Two");
➢ }
➢ public void m1(B b){
```

```
    System.out.println("Three");
}
public static void main(String[] args){
 A a = new Test();
 a.m1(new Test());
 }
}
```

- Ans: Output: Two
- c)

```
interface A {
 void m1(A a);
}
abstract class B implements A {
 void m1(B b){
  System.out.println("One");
 }
}
public class Test extends B {
 void m1(A a){
 System.out.println("Two");
 }
 void m1(B b){
```

```
    System.out.println("Three");
}
public static void main(String[] args){
 A a = new Test();
 a.m1(null);
 }
}
```

- Ans: No, *IllegalAcceessError* exception because we cannot reduce the visibility of inherited method from interface A.
- d)

```
interface A {
 void m1();
}
abstract class B implements A {
public void m1(){
  System.out.println("One");
 }
}
public class Test extends B {
public void m1(){
 System.out.println("Two");
 super.m1();
```

```
}
public static void main(String[] args){
 A a = new Test();
 a.m1();
 B b = new Test();
  b.m1();
 }
}
```

- Ans: Output: Two, One, Two, One
- e)

```
interface A {
 int m1();
}
class B implements A {
public int m1(){
  return 20;
 }
}
public class C implements A {
 public int m1(){
  return 30;
 }
```

```
}
public class Test {
public static void main(String[] args){
 A a = new B();
int aa =  a.m1();
System.out.println(aa);

C c = new C();
int cc = c.m1();
System.out.println(cc);
 }
}
```

- Ans: Output: 20, 30.
- f)

```
interface A {
 A m1();
}
class B implements A {
public A m1(){
 System.out.println("Red");
 return null;
 }
```

```
}
public class C implements A {
 public A m1(){
  System.out.println("Orange");
  return null;
  }
}
public class Test {
public static void main(String[] args){
 A a = new B();
  a.m1();
 C c = new C();
  c.m1();
 }
}
```

- Ans: Output: Red, Orange.
- **33. Will the below code compile successfully? If yes, what will be the output of the following program?**

```
interface A {
 A m1();
}
class B implements A {
```

```
public B m1(){
 System.out.println("Red");
 return new B();
 }
}
public class C implements A {
 public C m1(){
 System.out.println("Orange");
 return new C();
 }
}
public class Test {
public static void main(String[] args){
 A a;
 a = new B();
 a.m1();
 a = new C();
 a.m1();
 }
}
```

- Ans: Yes, the code will be successfully compiled. The result is Red, Orange.
- **34. What will be the output of the following program?**

```
interface A {
 int x = 20;
 interface B {
  int x = 30;
 }
}
class C implements A {
 int x = 40;
}
public class D implements A.B {
 int x = 50;
}
public class Test {
public static void main(String[] args){
 System.out.println(A.x);
 System.out.println(A.B.x);

 C c = new C();
 System.out.println(c.x);
```

```
➢
➢  D d = new D();
➢  System.out.println(d.x);
➢  }
➢ }
```

➢ Ans: Output: 20, 30, 40, 50.

➢ **35. Identify the error inside the following code and correct it. What will be the output after the correction of error?**

```
➢ interface A {
➢  void m1();
➢  interface B {
➢    void m1();
➢  }
➢ }
➢ class C implements A.B {
➢  public void m1(){
➢  System.out.println("Green");
➢  }
➢ }
➢ public class Test {
➢ public static void main(String[] args){
➢  A a = new C();
```

```
a.m1();
}
}
```

- Ans: This code will generate compilation error because class C is implementing inner interface, not outer interface. But the reference of a is a type of outer interface. The correct code is as follows:
- A.B ab = new C();
- After correction, the output is Green.
- **36. Is there any error in the following code? If yes, correct it.**

```
interface A {
 void m1();
 interface B {
  void m2();
 }
}
class C implements A, A.B {
 public void m1(){
 System.out.println("Green");
 }
}
```

- Ans: Yes, there is an error in the above code because class C must implement inherited abstract method from inner interface A.B.
- **37. Will the code compile successfully? If yes, what will be the output?**

```
interface A {
```

```
    void m1();
    interface B {
      void m2();
    }
  }
  class C implements A, A.B {
   public void m1(){
    System.out.println("Green");
    }
   public void m2(){
    System.out.println("Red");
    }
   }
   public class Test {
   public static void main(String[] args){
    C c = new C();
    c.m1();
    c.m2();
    }
  }
```

- Ans: Yes, the above code will be compiled successfully. The output is Green, Red.

38. Identify the error in the below code. If no error, what will be the output of the program?

```
interface A {
 void m1();
 interface B {
  void m2();
 }
}
interface C extends A, A.B {
 strictfp default void m1(){
 System.out.println("Green");
 }
 strictfp default void m2(){
 System.out.println("Red");
 }
}
public class Test implements C{
public static void main(String[] args){
 Test t = new Test();
 t.m1();
 t.m2();
```

```
}

}
```

Ans: No error. The result is Green, Red.

39. What will be the output of following program if no error?

```
public class A {

void m1(){

 System.out.println("One");

}

 interface B {

   void m2();

  }

}

class C extends A {

 void m1(){

  System.out.println("Two");

  super.m1();

  }

 }

public class Test implements A.B{

public void m2(){

 System.out.println("Three");

}
```

```
public static void main(String[] args){

 Test t = new Test();

  t.m2();

 C c = new C();

  c.m1();

 }

}
```

Ans: No error. The output is Three, Two, One.

40. What is a Functional interface in Java 8?

Ans: An interface that has exactly one abstract method is called functional interface in java. It is also known as single abstract method interface.

Functional interface can have three kinds of methods: abstract, default, and static methods.

Functional interface can have default methods with implementation. A default method cannot be abstract. For example, java.lang.Runnable and java.util.concurrent.Callable are two very popular Functional interfaces in java 8.

41. How to define a Functional interface in Java 8?

Ans: To define a Functional interface in Java 8, we can create an interface with exactly one abstract method like this:

```
interface A {

  void m1();

}
```

Another way is to define an Interface with annotation @FunctionalInterface. This annotation gives an instruction to the java compiler to verify that the interface has only one abstract method.

42. Is it mandatory to use @FunctionalInterface annotation with Functional interface in Java 8?

Ans: No, it is not mandatory to define a Functional interface with @FunctionalInterface annotation. Java does not impose this rule.

But, if we define an interface with @FunctionalInterface annotation, Java Compiler will give an error in case we define more than one abstract method within that interface.

43. What is default method in Java 8?

Ans: A method that is defined inside an interface with the default keyword and concrete method body is called default method. It provides a default implementation for classes that implements an interface.

This method can be overridden by a class that implements interface. A default method is public by definition and cannot be declared with private, protected, static, final, or abstract modifiers.

44. Which of the following are valid or non-valid default methods defined in an interface?

```
a) interface A {

   abstract void m1();

   default int m2() { return 10; }

}

b) interface A {

    abstract void m1();

    public default void m2(){ System.out.println("Hello Java");}

  }

c) interface A {

    static default m1() { }

   }

d) interface A {
```

```
    abstract void m1();

    static void m2() { System.out.println("Hello Java");  }

    public default void m3();

   }
```

Ans: a and b are valid codes. c and d are non-valid codes.

45. What is the output of the following program?

```
interface A {

 abstract void m1();

 static void m2(){

  System.out.println("Static method");

 }

 default void m3(){

  System.out.println("Default method");

 }

}

class B implements A {

 public void m1(){

  System.out.println("Overridden m1 method");

 }

}

public class Test{

public static void main(String[] args){
```

```
A a = new B();

a.m1();

A.m2();

a.m3();

 }

}
```

Ans: The result is Overridden m1 method, Static method, Default method.

46. Will the code successfully compile? If yes, what will be the output of the program?

```
interface A {

 abstract void m1();

 public static void m2(){

  System.out.println("One");

 }

 default void m3(){

  System.out.println("Two");

 }

}

class B implements A {

 public void m1(){

  System.out.println("Three");

 }
```

```
void m3(){
 System.out.println("Four");
 }
}
public class Test{
public static void main(String[] args){
 A a = new B();
 a.m1();
 A.m2();
 a.m3();
  }
}
```

Ans: No, the code will not be compiled successfully because we cannot reduce the visibility of inherited m3() method from interface A.

47. Identify the error in the following code and correct it.

```
interface A {
 abstract void m1();
 abstract default void m2(){
  System.out.println("Two");
 }
}
class B implements A {
```

```
public void m1(){
 System.out.println("Three");
}
}
public class Test{
public static void main(String[] args){
 A a = new B();
 a.m1();
 a.m2();
 }
}
```

Ans: Default method cannot be abstract. Remove it and then compile.

48. Why do we need Functional interface in Java 8?
Or, what is the use of Functional interface in Java 8?
Ans: A Functional interface is mainly used in Lambda expressions, method reference, and constructor references.

49. Can we declare private method inside an interface in JDK 9?
Ans: Before JDK 9, all methods defined in an interface were implicitly public. But JDK 9 allow to declare private method inside an interface.

The private method defined in an interface cannot be called or overridden by implementing classes.

50. Can we declare a method in an interface with combinations of modifiers like abstract, public, static, and private in JDK 9?
Ans: No, because they do not make sense.

Table: Supported Modifiers in Method Declaration in Interface

Modifiers	Supported?	Description

public static	Yes	Supported since JDK 8.
public abstract	Yes	Supported since JDK 1.
public default	Yes	Supported since JDK 8.
private static	Yes	Supported since JDK 9.
private	Yes	Supported since JDK 9. It is a non-abstract instance method.
private abstract	No	This combination does not make sense because a private method is not inherited. So, it cannot be overridden.
private default	No	It is also not possible.

Recommended Interview Questions

- Top 85++ OOPs Interview Questions with Answers
- Top 32 Interview Questions on Polymorphism
- Top 65 Interview Questions on Exception Handling
- Exception Handling Programming Questions in Java for best Practice

---

Hope that these 50 Java interface interview questions and answers will help you to understand the level of questions asked by interviewers in different companies. I hope that you will have practiced interface interview programming questions and enjoyed them.

Please, share it on social networking sites for your friends.
All the best!!!
