How to Describe Interfaces in Interviews?

- An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods.
- An interface cannot be instantiated.
- An interface does not contain any constructors.
- An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
- An interface is not extended by a class; it is implemented by a class.
- An interface can extend multiple interfaces.

Question: What is an Interface in Java?

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.
Along with abstract methods, an interface may also contain constants, default methods, static methods, and nested types. Method bodies exist only for default methods and static methods.
Writing an interface is similar to writing a class. But a class describes the attributes and behaviors of an object. And an interface contains behaviors that a class implements.
Unless the class that implements the interface is abstract, all the methods of the interface need to be defined in the class.

Question: What will happen if we define a concrete method in an interface in Java?

By default, interface methods are abstract.
if we declare any concrete method in an interface compile time error will come.
Error: Abstract methods do not specify a body.

Question: Can we create non static variables in an interface?

No. We cannot create non static variables in an interface. If we try to create non static variables compile time error comes.

By default, members will be treated as public static final variables, so it expects some value to be initialized.

```
package codespaghetti.com;
interface JavaInterface{

int x, y; // compile time error
}
```

Question: What will happen if we do not initialize variables in Java interface.

Compile time error will come because by default members will be treated as public static final variables, so it expects some value to be initialized.

```
package codespaghetti.com;
interface JavaInterface{

int x, y; // compile time error: The blank final field y may not have been initialized
}
```

Question: Can we declare interface members as private or protected?

No.

```
package codespaghetti.com;
interface JavaInterface{

private int x; // compile time error: Illegal modifier for the interface field Sample.x; only
public, static & final are permitted
protected int a; // compile time error: Illegal modifier for the interface field Sample.a; only
public, static & final are permitted
}
```

Question: When we need to use extends and implements?

A class will implements an interface. A class will extends another class. An interface extends another interface.

Question: Can we create object for an interface in Java?

NO. We can not create object for interface. We can create a variable for an interface

```
package codespaghetti.com;
interface JavaInterface{

void show();

}

package com.instanceofjava;
interface A implements JavaInterface {

void show(){
// code
}
public static void main(String args[]){

 JavaInterface obj= new JavaInterface(); // Error: Cannot instantiate the type JavaInterface

}
}
```

Question: Can we declare interface as final?

No. Compile time error will come. Error: Illegal modifier for the interface Sample; only public & abstract are permitted

we can't declare the interface as final because the implementation of the interface is provided by another class. If we make the interface as final, it will throw a compile-time error.

Question: Can we declare constructor inside an interface?

No. Interfaces does not allow constructors. The variables inside interfaces are static final variables means constants and we cannot create object for interface.
So, there is no need of constructor in interface that is the reason interface doesn't allow us to create constructor.

Question: Question: What will happen if we are not implementing all the methods of an interface in class which implements an interface?

A class which implements an interface should implement all the methods (abstract) otherwise compiler will throw an error. The type Example must implement the inherited abstract method JavaInterface.show() If we declare class as abstract no need to implement methods. No need of overriding default and static methods.

```
package codespaghetti.com;
interface JavaInterface{

void show();
}

package com.instanceofjava;
interface A implements JavaInterface { // The type Example must implement the inherited
abstract method JavaInterface.show()

public static void main(String args[]){

}
}
```

Question: How can we access same variables defined in two interfaces implemented by a class?

Yes, By Using corresponding interface.variable_name we can access variables of corresponding interfaces.

Question: If Same method is defined in two interfaces can we override this method in class implementing these interfaces.

Yes, implementing the method once is enough in class. A class cannot implement two interfaces that have methods with same name but different return type.

Question: Can we re-assign a value to a field of interfaces?

No. The fields of interfaces are static and final by default. They are just like constants. You can't change their value once they got.

Question: Can we declare an Interface with "abstract" keyword?

Yes, we can declare an interface with "abstract" keyword. But there is no need to write like that. All interfaces in java are abstract by default.

Question: For every Interface in java, .class file will be generated after compilation. True or false?

True, .class file will be generated for every interface after compilation.

Question: Can we override an interface method with visibility other than public?

No. While overriding any interface methods, we should use public only. Because, all interface methods are public by default and you should not reduce the visibility while overriding them.

Question: Can interfaces become local members of the methods?

No. You can't define interfaces as local members of methods like local inner classes. They can be a part of top-level class or interface.

Question: Can an interface extend a class?

No, A class can implement an interface, but interface cannot extend a class.

No, a class cannot become super interface to any interface. Super interface must be an interface. That means, interfaces don't extend classes but can extend other interfaces.

Question: Like classes, does interfaces also extend Object class by default?

No. Interfaces don't extend Object class. But all the classes extend Object class by default.

Question: Can interfaces have static methods?

No. Interfaces can't have static methods. Interfaces can have static methods since Java 1.8.

all the methods are by default public and abstract in the interface

Question: Advantage and disadvantages of Interfaces

Advantages

- Interfaces are mainly used to provide polymorphic behavior.
- Interfaces function to break up the complex designs and clear the dependencies between objects.

Disadvantages

- Java interfaces are slower and more limited than other ones.
- Interface should be used multiple number of times else there is hardly any use of having them.

What is default keyword in an interface?

By the help of default keyword, we can keep non-abstract method in java interface i.e with method body {}. This is the new feature of JAVA 8.

Syntax of default keyword:

```
interface Test

{

default void show ()

{

-----

-----

}

}
```

What is marker or tagged interface?

Marker interface is an interface that has no data member and method like Serializable, Cloneable, etc.

After compilation of interface program, .class file will be generated for every interface in java... true or false.?

This is true.

Difference between abstract class and interface?

There are many differences between abstract class and interface.

abstract class

- Through abstract class, you cannot achieve multiple inheritance.
- You can keep non-abstract method(with method body) in the abstract class.
- In the abstract class, fields are not public, static, final and methods are not public abstract by default.

interface

- Through the interface, you can achieve multiple inheritance.
- In an interface, you can't keep non-abstract method but since java 8 it is possible.
- In an interface, fields are public, static, and final and methods are public and abstract by default.