INTERVIEW QUESTIONS

CSE JAVA – 8

Abstraction

Chapter 1 - Interfaces

1. What is an Interface?

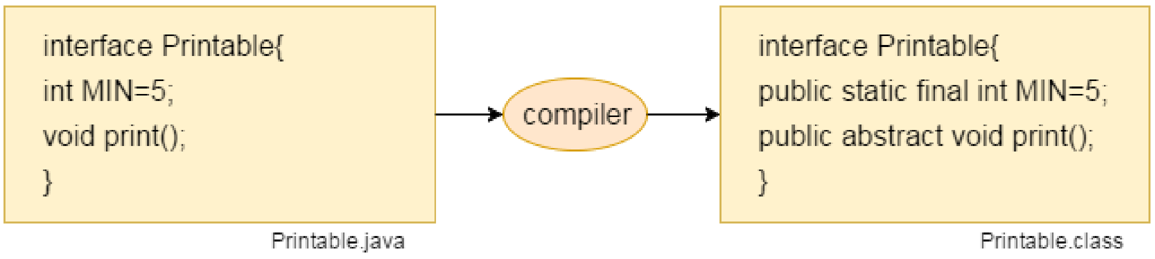
⇨ An interface is a reference type in java or rather we can say it's a collection of different types of methods and which are by default public and abstract which means the interface can only have method declarations but implementations should be done by the implementing class of that interface. A class that implements an interface should override all the abstract methods present in that interface otherwise must be declared as abstract class.

2. What is the key feature of an interface?

⇨ Interfaces can be said as a blueprint of a class because it holds all the method declarations but not the definitions and hence it shows abstraction.

⇨ Key points of interfaces are ➔

- All the methods of an interface are by default **public and abstract**.
- All the variables of an interface are by default **public static and final.**
- From the **JDK version –8.0.0, we can able to add static and default methods in an interface that can include implementations.**
- An interface can contain the main method from JDK—8
- One interface can extend multiple interfaces that are not possible in between classes.
- One class can implement multiple interfaces by a comma separator that's why multiple inheritances are possible in interfaces.



```
interface Printable{
int MIN=5;
void print();
}
```

```
interface Printable{
public static final int MIN=5;
public abstract void print();
}
```

3. Is it necessary to declare a method as public during implementation?

⇨ Yes, it is compulsory to declare the method as public during implementation because of these reasons ➔

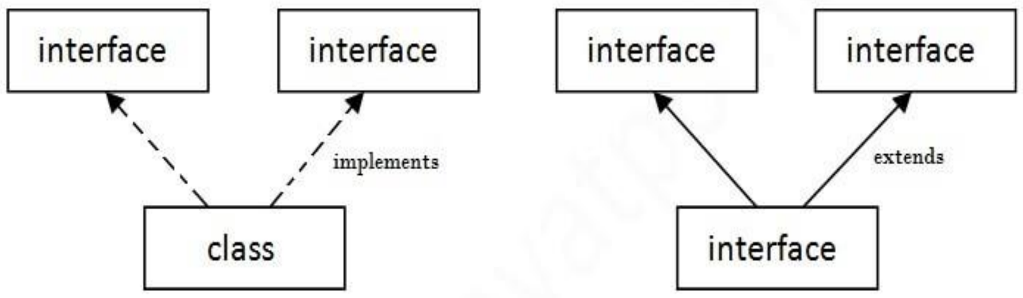
- All the methods present in the interface are public so when we define it inside implementing class then we must declare it as public otherwise privileges will be reduced from public to default This is not possible in java overriding.
- java compiler will not compile the application if implementing class not declare the method of an interface as public.
- In java widening of privileges are possible in overriding but narrowing the privileges are not possible

4. How multiple inheritance is possible via interfaces?

⇨ As we see that a single class can able to implement more that one interface separated by comma separator but a single class can not able to extend more than one class so using the interface we can achieve multiple inheritances in java.

⇨ One more point to be noted that interfaces also one kind of "Is a relationship"

⇨ And one more point one interface can able to extend another interface or interface that means it also supports multiple inheritances in between interfaces.



Multiple Inheritance in Java

5. Points ➔ Interface and abstract class both can not be instantiated but both can have reference variables of the implementing or extending classes.

6. Why java doesn't support multiple inheritances for class and why this scenario is changes for interfaces?

⇨ Java doesn't support multiple inheritances for mainly two reasons ➔

- The first reason I would like to say is ambiguity because if one class extends suppose two another classes and by any chance, those two classes have a method with same name and signature then at runtime JVM will confuse about which one to choose as both of them contain same method **(Death of Diamond -- Problem)** so as this multiple inheritance creates these type of problems so to maintain simplicity java omits these multiple inheritances.
- Another big problem comes in multiple inheritances in **the constructor chaining process** while one class extends two different classes and creates an object of own then at that time all the parent constructors will be called if there is more than one parent then it will create a problem in chaining. So make the whole picture simple java removes multiple inheritances.

7. What is the difference between abstract methods and interfaces?

Abstract class	Interface
1) An abstract class can **have abstract and non-abstract** methods.	An interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritances**.	The interface **supports multiple inheritances**.
3) An abstract class **can have final, non-final, static, and non-static variables**.	The interface has **only static and final variables**.
4) An abstract class **can provide the implementation of the interface**.	Interface **can't provide the implementation of an abstract class**.
5) The **abstract keyword** is used to declare an abstract class.	The **interface keyword** is used to declare an interface.
6) An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.

7) An **abstract class** can be extended using the keyword "extends".	An **interface** can be implemented using the keyword "implements".
8) A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.

9)**Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

8. What are the advantages of using an interface?

⇨ There are many advantages of using interfaces.

- We can achieve multiple inheritances in java that seems to be impossible in java without interface.
- We can achieve an end to end abstraction using interfaces.
- From JDK version 8.0 it is possible to add default and static methods in an interface so it becomes easy to debug and modify any project on the go.
- It can be used to achieve loose coupling.

9. What is a maker or tagged interface && nested Interface?

⇨ Tagged or maker interface are some special type of interface with nobody within it. It has no member within it is known as the tagged interface. Example ➔ some tagged interfaces are cloneable, serializable interface, etc.

⇨ When interfaces have other interfaces within themselves then its known as nested interface.

Chapter 2 – Abstract Classes

1. What is an abstract class in java?

- ⇨ Abstract classes are those classes that are declared with abstract keyword and may contain abstract methods as well as non-abstract methods and these types of classes can not be instantiated and using these abstract classes we can able to achieve abstraction in java.

2. Some key points should be remembered about abstract classes

- ⇨ Abstract classes can contain abstract methods as well as non-abstract methods.
- ⇨ Abstract classes must be inherited by other classes to get achieve abstraction and provide implementation details for the abstract methods.
- ⇨ Abstract classes can not be instantiated.
- ⇨ Abstract classes can have reference variables of the objects of the extending classes.
- ⇨ Abstract classes can have constructors and it is as same as a normal constructor.
- ⇨ Abstract classes may contain static & final methods **(main methods also).**

3. Can we ever able to create an object of an abstract class?

- ⇨ Yes, directly using a new keyword it's not possible as an abstract class may contain abstract methods without implementation so using new directly it will give a compile-time error.
- ⇨ But we can create an object of abstract classes using two ways ➔
  - Using a Concrete subclass we can instantiate abstract classes.
  - By defining all the abstract methods of an abstract class we can create an object of abstract class along with the new keyword.

```
abstract class DemoAbstractClass {

    abstract void display();
}

public class JavaApplication {
    public static void main(String[] args)
    {
        DemoAbstractClass AC = new DemoAbstractClass()
{
            void display()
            {
                System.out.println("Hi.");
            }
        };  /// We are creating instance of an
abstract class by defining all the abstract methods.
        AC.display();
        System.out.println("How are you?");
    }
}
```

4. What is the difference between abstract class and concrete class?

Abstract class	Concrete class
An abstract class is declared using an abstract modifier.	A concrete class is a note declared using an abstract modifier.
An abstract class cannot be directly instantiated using the new keyword.	A concrete class can be directly instantiated using the new keyword.
An abstract class may or may not contain abstract methods.	A concrete class cannot contain an abstract method.
An abstract class cannot be declared as final.	A concrete class can be declared as final.

5. Can abstract methods or classes be declared with the final?

- ⇨ No, it's not possible at all because final keyword disallows the inheritance feature of any class and when declared with method then those methods can not be overridden so if we declare any abstract methods as final so it can not be overridden but abstract methods must be overridden by the extending class. So it is not possible to declare method or class with final and abstract both.

6. Can any class declare as an abstract modifier without any abstract methods?

- ⇨ Yes, it is possible to declare any class with abstract non-access specifier without abstract methods.
- ⇨ As an example we can say some predefined classes of java are like this --> " HttpServlet " class of java declared with abstract keyword but not containing any abstract methods.
- ⇨ Now the main reason for it that this class has not any proper implementation for any methods within it so there is no need to create an object for that class this is the reason for declaring a class as abstract to prevent instantiation.
- ⇨ Now the question arises that if it is so then why those not properly implemented methods are not declared as abstract to be defined properly by its child classes --> If those methods are declared with abstract class then it will create a problem that whenever we need to use HttpServlet class we must extend and override all of the abstract methods while we need only two.

7. Can we have the main method within any abstract class?

- ⇨ Yes, we can have the main method within the abstract class as we can able to declare static final and default methods within abstract classes.
- ⇨ //abstract class with the main method