Java Interface

1. **What is an Interface in Java?**
   Ans: An interface in Java is a mechanism that is used to achieved complete abstraction. It is basically a kind of class that contains only constants and abstract methods.

2. **Which modifiers are allowed for method in an Interface? Explain with an example.**
   Ans: Only abstract and public modifiers are allowed for methods in interfaces.

   Example:
   ```
   interface MyInterface{
      public abstract void display();
      public abstract void setName(String name);
      public abstract void setAge(int age);
   }
   ```

From Java8 onwards interfaces allow default methods and static methods.

- **Static methods** – A static method is declared using the static keyword and it will be loaded into the memory along with the class. You can access static methods using class name without instantiation.
- You need to call static method of an interface using the name of the interface.
-
```
package Interface_Ex;

public interface InterfaceEx1 {
   public void demo();
   public static void display() {
      System.out.println("This is a static method");
   }
}
class InterfaceExample{
   public void demo() {
      System.out.println("This is the implementation of the demo method");
   }
   public static void main(String args[]) {
      InterfaceExample obj = new InterfaceExample();
```

```
            obj.demo();
            InterfaceEx1.display();
        }
    }
```

- **Default methods –** A default method is a default implementation of a method of an interface, if you have default method in an interface, there is no need to implement it in the classes that already implement this interface.
- A default method is also known as defender method or virtual extension method. You can define a default method using the *default keyword*

```
•   package Interface_Ex;

    interface sampleInterface{
        public void demo();
        default void display() {
            System.out.println("This is a default method");
        }
    }
    public class InterfaceEx3 implements sampleInterface{
        public void demo() {
            System.out.println("This is the implementation of the demo method");
        }
        public static void main(String args[]) {
            InterfaceEx3 obj = new InterfaceEx3();
            obj.demo();
            obj.display();
        }
    }
```

From Java9 onwards interfaces allow private and private static methods.

```
package Interface_Ex;

interface MyInterface {
    public abstract void demo();
    public default void defaultMethod() {
        privateMethod();
        staticPrivateMethod();
        System.out.println("This is a default method of the interface");
    }

    public static void staticMethod() {
        staticPrivateMethod();
```

```java
        System.out.println("This is a static method of the interface");
    }

    private void privateMethod(){
        System.out.println("This is a private method of the interface");
    }

    private static void staticPrivateMethod(){
        System.out.println("This is a static private method of the interface");
    }
}

public class InterfaceEx2 implements MyInterface {
    public void demo() {
        System.out.println("Implementation of the demo method");
    }

    public static void main(String[] args){
        InterfaceEx2 obj = new InterfaceEx2();
        obj.defaultMethod();
        obj.demo();
        MyInterface.staticMethod();
//        obj.privateMethod();
    }
}
```

3. **What is the use of interface in Java? Or, why do we use an interface in Java.**

   Ans: There are many reasons to use interfaces in java. They are as follows:
   - An interface is used to achieve full abstraction.
   - Using interfaces is the best way to expose our project's API to some other project.
   - Programmers use interfaces to customise features of software differently for different objects.
   - By using interface, we can achieve the functionality of multiple inheritance.

4. **What is the difference between abstract class and interface in Java?**

   Ans:

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare 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 keyword "extends".	An **interface** can be implemented using 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();
	}