Day 23 Interface Assignment

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

Ans. An interface is a reference type that defines a contract for the behaviour of a class. It specifies a set of methods that a class implementing the interface must implement. An interface serves as a blueprint for classes, outlining what methods they should have without providing the implementation details.

2. **Which modifiers are allowed for method in an interface? Explain with an example**

Ans. In an interface, the only allowed modifiers for methods are **public** and **abstract**. However, since all methods in an interface are implicitly **public** and **abstract**, you don't need to explicitly specify these modifiers.

```
public interface MyInterface {
   void method1(); // implicitly public and abstract

   public abstract void method2(); // can also be explicitly specified
}
```

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

Ans. Interfaces in Java serve several purposes and provide various benefits. Here are some of the main reasons why we use interfaces in Java:

1. **Abstraction and Polymorphism**: Interfaces allow for abstraction by separating the definition of behavior from its implementation. They provide a way to program at a higher level, focusing on what needs to be done rather than how it's done. Interfaces also facilitate polymorphism, allowing objects of different classes to be treated interchangeably if they implement the same interface. This enhances flexibility and code reusability.
2. **Multiple Inheritance of Type**: Java classes can only inherit from a single class, but they can implement multiple interfaces. This enables a class to inherit the behavior specified by multiple interfaces, allowing for a form of multiple inheritance of type. This is useful when a class needs to exhibit different behaviors from different sources.
3. **Loose Coupling**: Interfaces promote loose coupling between classes. By programming to interfaces rather than concrete implementations, classes can interact with each other through interfaces, reducing dependency on specific implementations. This enhances flexibility, maintainability, and modularity in the codebase.
4. **API Design and Extension**: Interfaces are commonly used for designing APIs (Application Programming Interfaces) in Java. By defining interfaces, you can create a clear and consistent contract for the external users of your code.

Interfaces also provide a way to extend existing functionality by introducing new interfaces that extend or refine the behavior of existing interfaces.

5. **Testing and Mocking**: Interfaces make it easier to test and mock code. By coding to interfaces, you can create mock implementations of interfaces during testing, allowing for isolated testing of individual components. This promotes effective unit testing and improves the overall testability of the code.

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

Ans.

Aspect	**Abstract Class**	**Interface**
Implementation	Can have both implemented and abstract methods	Can only have method declarations (abstract methods) and constants
Inheritance	Supports single inheritance (extends one class)	Supports multiple inheritances of type (implements multiple interfaces)
Accessibility	Can have different access modifiers for members	All members are implicitly public
Constructors	Can have constructors	Cannot have constructors
Usage	Used when there is shared functionality and "is-a" relationship between classes	Used to define contracts, achieve multiple inheritances of type, and promote loose coupling
Instantiation	Cannot be directly instantiated	Cannot be directly instantiated