Interface Assignment Questions

Assignment Questions:

1. What is an interface in Java?

Ans → In Java, an interface is a collection of abstract methods (methods without a body) and constant values that define a contract for a class to implement. It provides a way to achieve abstraction, as it defines a set of methods that a class must implement without specifying how they should be implemented.

1. Which modifiers are allowed for methods in an Interface? Explain with an example.

Ans → In an interface in Java, the only modifiers that are allowed for methods are public and abstract. These modifiers are automatically assumed for methods declared in an interface if they are not explicitly stated.

The public modifier makes the method accessible from any other class in the program, while the abstract modifier indicates that the method does not have an implementation in the interface and must be implemented by any class that implements the interface.

Here is an example interface in Java with a method that uses both public and abstract modifiers:

public interface MyInterface {

public abstract void doSomething();

}

In this example, the method doSomething() is declared with both public and abstract modifiers. The public modifier makes the method accessible to any class that implements this interface, while the abstract modifier indicates that this method does not have an implementation in the interface and must be implemented by any class that implements it.

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

Ans → In Java, interfaces are used for achieving abstraction and creating a contract between different classes. The main reasons for using interfaces in Java are:

1. Abstraction: An interface provides a way to achieve abstraction in Java by defining a set of methods that a class must implement without specifying how they should be implemented. This allows the programmer to focus on what the class should do, rather than how it should do it.
2. Multiple Inheritance: Java does not allow multiple inheritance of classes, but a class can implement multiple interfaces. This allows a class to inherit the behavior of multiple interfaces and provides more flexibility in designing object-oriented programs.
3. Polymorphism: Interfaces allow for polymorphism in Java, which means that an object of a class that implements an interface can be treated as an instance of that interface. This allows for more flexibility in designing code that can work with multiple types of objects.
4. Loose Coupling: Interfaces help to achieve loose coupling between classes, which means that the classes are not dependent on each other's implementation details. This makes the code more modular and easier to maintain.
5. Contract: An interface provides a contract between the class that implements it and the code that uses it. This contract defines the behavior of the class and ensures that any code using the class can rely on that behavior.
6. What is the difference between abstract class and interface in Java?

Ans →

1. Implementation: An abstract class can have both abstract and non-abstract methods, while an interface can only have abstract methods. In an abstract class, the abstract methods must be implemented by the subclass, but the non-abstract methods can be inherited without implementation. In contrast, all methods in an interface must be implemented by the class that implements the interface.
2. Multiple Inheritance: A class can only extend one abstract class, but it can implement multiple interfaces. This allows for more flexibility in designing object-oriented programs.
3. Accessibility: An abstract class can have private, protected, and public members, while all members of an interface are implicitly public and cannot have private or protected access.
4. Constructor: An abstract class can have a constructor, while an interface cannot have a constructor.
5. Variables: An abstract class can have instance variables that are inherited by its subclasses, while an interface cannot have instance variables. An interface can have constant variables, which are implicitly static and final.
6. Purpose: Abstract classes are designed to be extended by subclasses that provide a specific implementation, while interfaces are designed to be implemented by classes that provide their own implementation.