Interface solution

1. In Java, an interface is a reference type similar to a class that can contain only constants, method signatures, default methods, static methods, and nested types. It provides a way to achieve abstraction and multiple inheritance in Java.

2. In an interface, the following modifiers are allowed for methods:

- public: Methods in an interface are implicitly public and abstract if not specified otherwise.

- abstract: This modifier is redundant as interface methods are implicitly abstract.

- default: Starting from Java 8, interfaces can have default method implementations.

- static: Starting from Java 8, interfaces can have static methods.

Example:

java

interface Animal {

void makeSound(); // public and abstract by default

default void eat() {

System.out.println("Animal is eating");

}

static void sleep() {

System.out.println("Animal is sleeping");

}

}

3. Interfaces in Java are used to define a contract for classes that implement them. They help in achieving abstraction, multiple inheritance, and provide a way to establish a communication between unrelated classes through a common interface. They also facilitate code reusability and maintainability by allowing polymorphic behavior.

4. Differences between abstract class and interface in Java:

- An abstract class can have both abstract and concrete methods, while an interface can only have method signatures (abstract methods) and constants.

- A class can extend only one abstract class, but it can implement multiple interfaces.

- Abstract classes can have constructors, instance variables, and non-static blocks, whereas interfaces cannot.

- Abstract classes can have access modifiers for their constructors and methods, while interface methods are implicitly public and abstract.

- Abstract classes are used to provide a default implementation and common functionality to related classes, while interfaces are used to define a contract for unrelated classes.