Interface and Abstract Class

27 September 2024 18:51

INTERFACE

- *An interface in Java looks similar to a class but both the interface and class are two different concepts. An interface can have methods and variables just like the class but the variables declared are static and final by default and methods declared in interface are by default abstract until java 8 . In Java 8, interfaces were enhanced to support default and static methods. you can even add private methods on an interface from Java 9 onwards*
- *If we wants to defines a contract for multiple classes then we go for interface*
- *We can't create object for interface*
- *We can achieve pure abstraction using interface until java 8. If we want to achieve pure abstraction after java 8, you can simply avoid using default methods and rely on abstract methods.*
- *Multiple inheritance in Java can also be achieve using Interface*
- ***Eg**: WebDriver in selenium is a good example for Interface.*

When to Use an Abstract Class vs. Interface:

- *Use **Abstract Class** when you have a base class with some common functionality that should be shared among related classes but also requires the subclasses to implement specific methods.*
- *Use **Interface** when you want to enforce a contract or capability that multiple unrelated classes can implement.*

Abstract Class vs Interface:

Feature	Abstract Class	Interface
Methods	Can have both abstract and concrete methods.	Can have abstract, default, and static methods (since Java 8).
Fields	Can have instance variables (fields).	Can only have static and final fields (constants).
Constructors	Can have constructors.	Cannot have constructors.
Multiple Inheritance	A class can extend only one abstract class.	A class can implement multiple interfaces.
Use Case	Use when you want to provide partial implementation or shared code.	Use when you want to define a contract for other classes to follow, without any implementation.
Access Modifiers	Can have different access levels for methods (public, protected, etc.).	Methods in an interface are public by default (abstract methods).

Abstract Keyword:

- *It is a keyword which is used to achieve abstraction and it can be used only with methods and classes*
- *If a method is declared as abstract we can't provide implementation for that method*
- *If a class is declared as abstract we can’t create object for that class*
- *Abstract keyword can't be used with private, static and final since method overriding is not possible for these keywords*

ABSTRACT CLASS

- *An abstract class in Java is a class that is declared with the abstract keyword. It cannot be instantiated directly and is meant to be extended by other classes. The main purpose of an abstract class is to provide a base or blueprint for other classes while allowing subclasses to provide the implementation of certain methods based on the sub class specifications. Used in a situation where some logic is shared across different subclasses, while specific details are left to be implemented by the subclasses.*
- *It contains both **abstract (method without implementation)** and **concrete methods (method with implementation)***
- *Since it is a class it can have constructor*
- *Multiple inheritance is not possible since it is class*

Can Abstract Class Have Constructors?

- *Yes, an abstract class can have constructors. While you cannot instantiate an abstract class directly, its constructor can be called when an object of a subclass is created, and it helps in initializing fields defined in the abstract class.*

Why can't we instantiate an abstract class?

- *An abstract class can contain abstract methods (without a body), which means it does not have a full implementation. Creating an object of such a class would make no sense since some methods would be undefined.*
- *The purpose of abstract classes is to be extended by concrete subclasses that provide the full implementation of all abstract methods.*