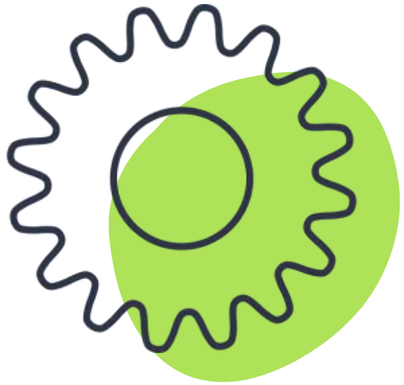
ABSTRACT CLASS & INTERFACES

ABSTRACT CLASS

1. An Abstract class is like a normal class , can contain properties / constructor / concrete method but at least one method that is incomplete (Abstract method)
2. When a method is abstract, the class also becomes abstract (class should be modified with keyword abstract).
3. An object cannot be created for an abstract class,



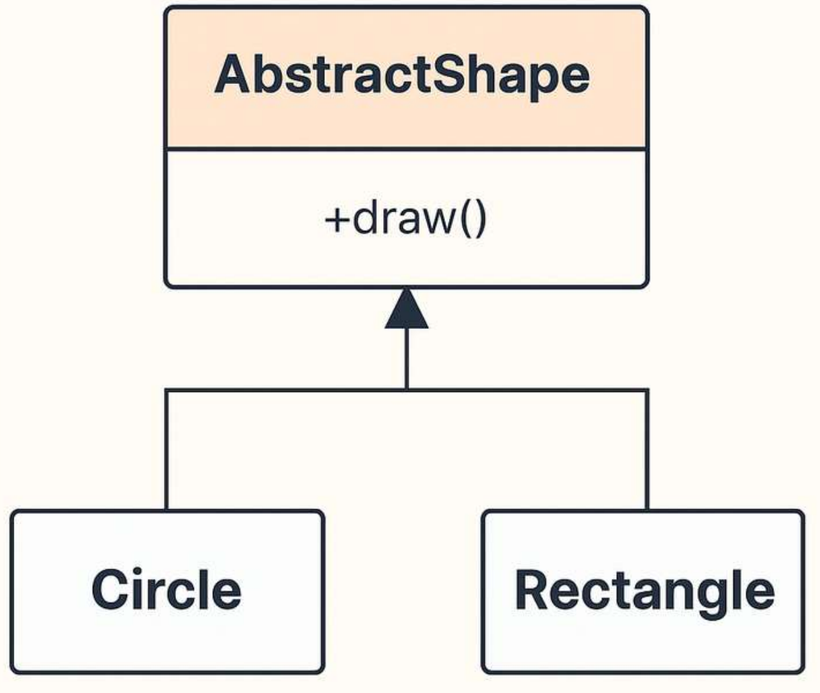
NOTES :

1. An object cannot be created for an abstract class, only a reference is possible.
2. If the child class does not override the abstract method of the parent abstract class, then even the child class also become abstract.

DRAWBACKS OF ABSTRACT CLASS :-

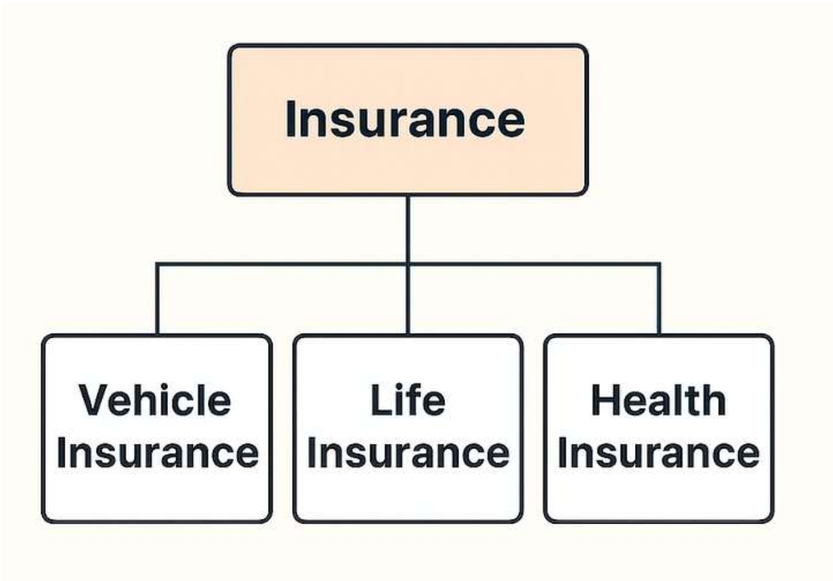
- No multiple inheritance — can't extend more than one class
- Tight coupling — changes in abstract class affect all subclasses
- Less flexible than interfaces — limited adaptability
- Can't instantiate — must use subclass
- Encourages inheritance over composition — less modular
- Risk of incomplete implementation — subclasses must implement all abstract methods

SIMPLE DIAGRAM :



Here’s a simple diagram showing how an abstract class works in object-oriented programming. It illustrates an abstract class Shape with an abstract method draw(), and two concrete subclasses Circle and Rectangle.

REALTIME USECASE :

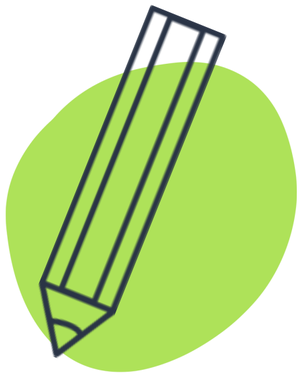


INTERFACES :

- It allows loosely coupled applications.
- It is contract to be signed mandatorily by the implementing class.
- It leads to polymorphism.
- An Interface can contain the following :

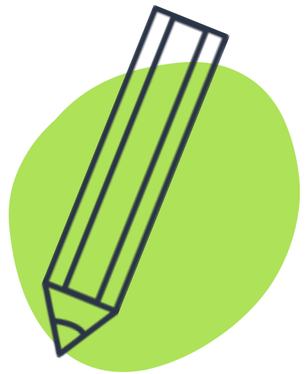
  static & final keyword

  (they are non-access modifiers)



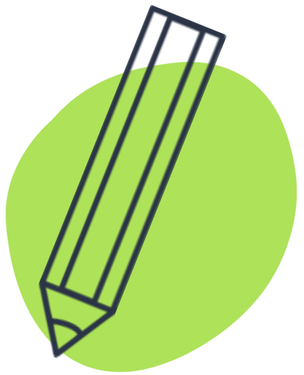
NOTES :

- Object is not possible for an interface, only a reference is possible with the help of Dynamic Method Dispatch (DMD), we can invoke the appropriate method.
- A class can implement “N” no of interfaces.



INTERFACE ADVANTAGES OVER ABSTRACT CLASS :

Abstract Class Drawback	How Interface Solves It
❌ No multiple inheritance	✅ Interfaces support multiple implementation
🔒 Tight coupling	✅ Interfaces promote loose coupling
🧱 Less flexible for contracts	✅ Interfaces define clean, adaptable contracts
🚫 Can't instantiate	✅ Interfaces are meant for implementation only
📦 Inheritance over composition	✅ Interfaces encourage composition and delegation
⚠️ Risk of incomplete methods	✅ Interfaces enforce method implementation



Thank you!