* Modifiers in Java:

- There are Two Types of modifiers
  1. Access modifiers
  2. Non Access modifiers.

1. Access modifiers:

- The Access modifiers control the visibility and accesability of classes, methods, variables, constructors and other elements within your code.
- There are four types of access modifiers in Java.

* **Public** : The member is accessable from all other classes in packages.

* **Protected** : The member is accessable within the same package and by subclasses (which can be in a different package)

* **default (No modifier)** : if no access modifier is specified, it default to package-private, meaning it is accesable only within it's own package.

* **private** : The member is accessable only within the class it is declared.

2. Non Access Modifiers:

- Non-access modifiers provide additional functionality beyond access control.

- These are some common non-access modifiers.

- **final**: Declares a member (variable, method, or class) to be constant or unchangeable.

  Ex: The Pi (π) has a constant value so Math.PI is constant in Java has a final modifier, ensuring it's value (3.14) remains consistent throughout the program.

- **Static**: Declares a member (variable or method) to be class-level rather than instance level. static members are shared across all instances of the class and can be accessed without creating an object.

  Ex: A school system where a static variable could represent the school's name. It's common accross all instances (students, Teachers).

- **Abstract**: Declares a method or class as incomplete, requiring subclasses to provide their own implementation. This is often used in inheritence to establish a common interface.

Ex: a general concept like a credit card which can't be an actual card until specific details like card Number, Holder Name are defined.

Abstract class & Abstract Methods

* Abstract class:

- in Java, an abstract class is a class that cannot be instantiated on it's own and must be inheriteed by other classes.
- An abstract class is used to set up a "template" for other classes.
- it's a way of enforcing certain methods to be implemented by any subclass that extends the abstract class

charecteristics of Abstract classes:

- Cannot be instantiated: you cannot create an object of an abstract class directly.
- can have Constructors: Abstract classes can have constructors, but these constructors are called only when an instance of a derived class is created.
- Can have methods with or without implementations: Abstract classes can have both regular methods (with implementations) and abstract methods (without implementations)

Abstract Methods :

* An abstract method is a method that is declared without an implementation.
* abstract methods acts as placeholders for methods that are implemented in Subclasses.
* Declaring a method as abstract enforces all subclasses to override and provide a specific implementation for the method.

Charecteristics of Abstract Methods :

- **No body** : Abstract methods do not have a body: they only have method signature.
- **Must be overridden** : Any non-abstract subclass must override the abstract methods of it's parent class.
- **Enforces a Contract** : By using an abstract method, you ensure that all subclasses adhere to a certain behaviour (they must implement the method)

Purpose of Abstract classes And Methods :

- The Primary purpose of using abstract classes and methods is to define a standard form for a set of subclasses.
- This approach follows the software engineering principle of abstraction, allowing you to work with more general types in your code and manage more specific types dynamically at runtime.