3. Encapsulation

↳ binding all the data members and functions together in a class.

→ Wrapping everything inside class.

4. Abstraction

↳ hiding all the unneccasary details and showing only the required part

→ Abstract class → Interface

Access Modifiers / Specifiers

↳ Specifies access / permission

1. Public
2. Protected
3. Default
4. Private

	Class	Package	Sub class in same pack	Sub class in diff package	Anywhere
Public	✓	✓	✓	✓	✓
Protected	✓	✓	✓	✓	
Default	✓	✓	✓		
Private	✓				

Abstract classes and interfaces

↳

Abstract method

↳ don't have body / definition

Syntax

```
abstract void sum(int a, int b);
```

Any class having 1 or more abstract method need to be made abstract;

→ Can't create an object

Interface

↳ multiple inheritance

↳ achiving abstraction

Syntax

Class → extends

```
interface Vehicle {
  int getSpeed ();

}
```

→ implements

→ Every method is abstract and public

→ Can't create object. Only reference can be created

→ data members / variables will be → static
final
public

→ multiple inheritance

$$\begin{tikzcd} & \text{Interface} \arrow[dl, "\text{implements}"'] & \\ \text{Class} & & \text{Interface} \arrow[ul, dash, "\text{extends}"'] \end{tikzcd}$$