(3) Encapsulation

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

→ Wrapping everything inside class.

(4) Abstraction

↳ hiding all the unnecessary 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 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

→ Can create reference variable

static abstract methods ✗

static methods / variables ✓

normal methods ✓

constructors ✓ (super())

abstract Constructor ✗

final abstract class ✗

multiple inheritance ✗

Interface

↳ multiple inheritance

↳ achiving abstraction

Syntax

Class → extends

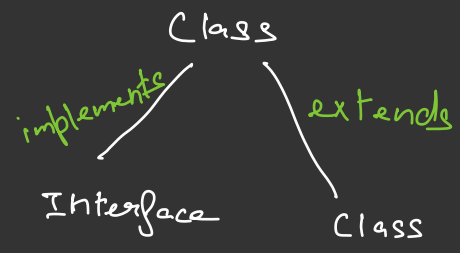
```
interface Vehicle {        → implements
    int getSpeed();
}
```

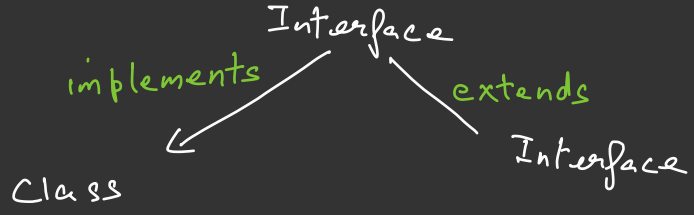
→ Every method is abstract and public

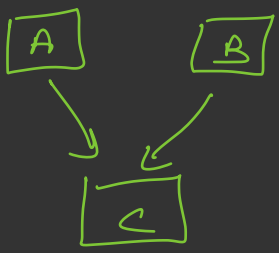
→ Can't create object. Only reference can be created

→ data members / variables will be → static final public

→ multiple inheritance








```
A {
  int speed();
}
```

```
B {
  int speed
}
```

```
C c = new C();
c.speed();
```

```
class C {
  int speed() {
    return 20;
  }
}
```