Abstraction

- Encapsulation
- Inheritance
- Polymorphism

⇒ Encapsulation :-

Encapsulate ⇒ capsulate ⇒ capsule

purpose of the capsule covering :-

i) binds the medicines together

ii) protects the medicine from outside env

iii) consumption is easy

iv) packaging and transportation is easy

Two things

→ attributes — properties

→ methods — behaviours

We need to bind / encapsulate attributes & methods for an entity together :-

⇒ Class

```
class Student {

    String name;
    int marks;
    int rank;
    String SchoolName;
    String phoneNo.;

    public Student() {

    }

    public void goesToSchool() {

    }

    public void attendsLecture() {

    }

    public int getMarks() {

    }

}
```

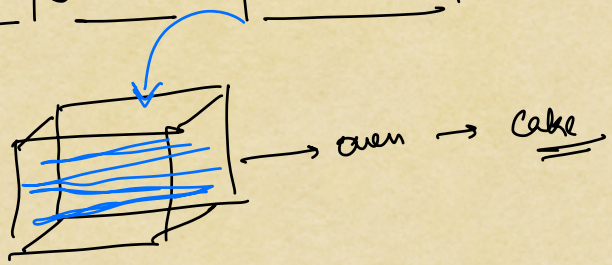
attributes

Constructors

methods

⇒ Class and Object :

Cake tin | Cake mould | box mould



1 cake tin → no existence as an edible item

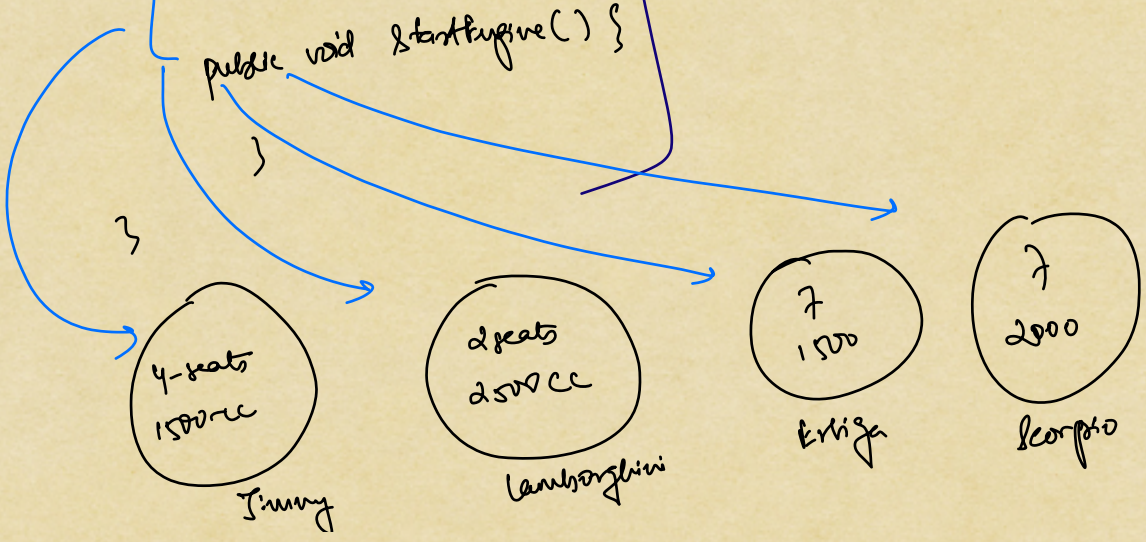
↓

Cake → edible and has existence as edible item

* all cakes from 1 cake tin will be same structurally
* all cakes even though having same structure might different flavours, toppings

```
class Car {
    int seats;
    int engineSize;
    public void startEngine() {
    }
}
```

← cake mould | cake tin
* provides the structure



* All Objects from the same class will have the same structure
* Values of the attributes for each might differ.

Capsule → Object [attribute & values & methods]

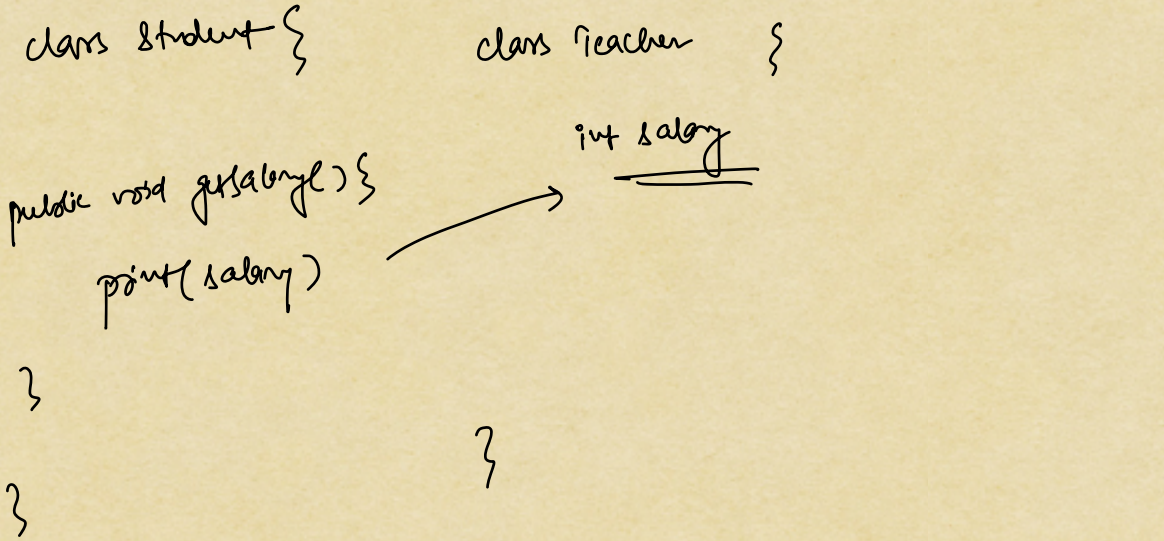
class → mould for capsule

→ How do you protect the data in your Objects :-

* access modifiers
  * public
  * private
  * protected
  * default / package-private

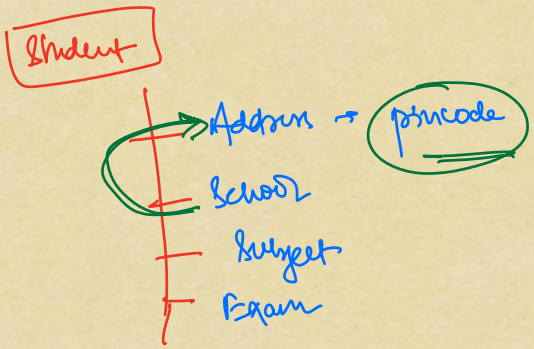
→ access modifiers / access specifiers :-

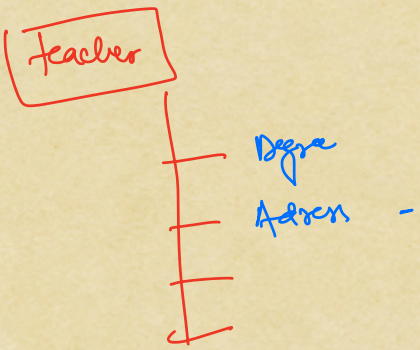
they define the access of members of a class from a location outside of a class.



different package child class

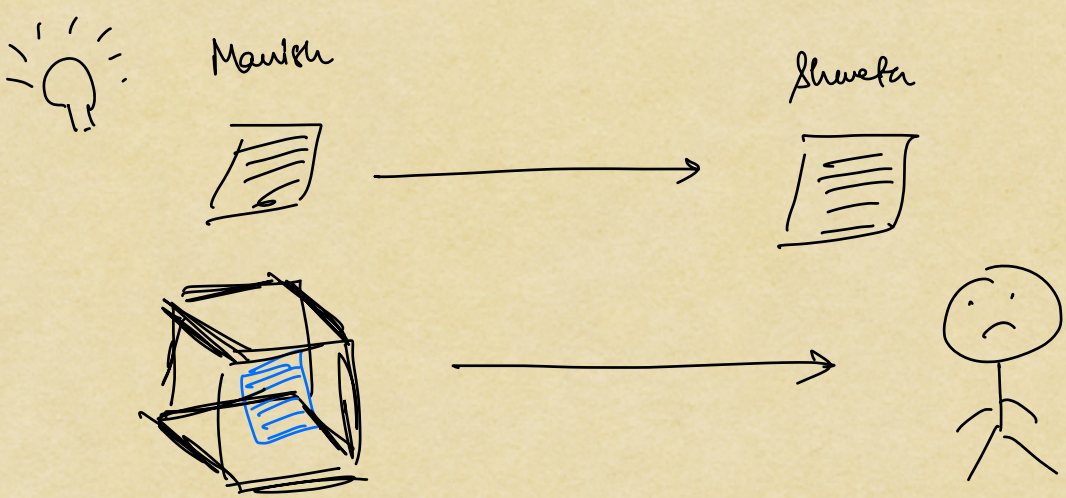
	inside the class	outside the class but same package	outside the same package	outside the same package but child class	global
public	✓	✓	✓	✓	✓
private	✓	✗	✗	✗	✗
protected	✓	✓	✗	✓	✗
default	✓	✓	✗	✗	✗

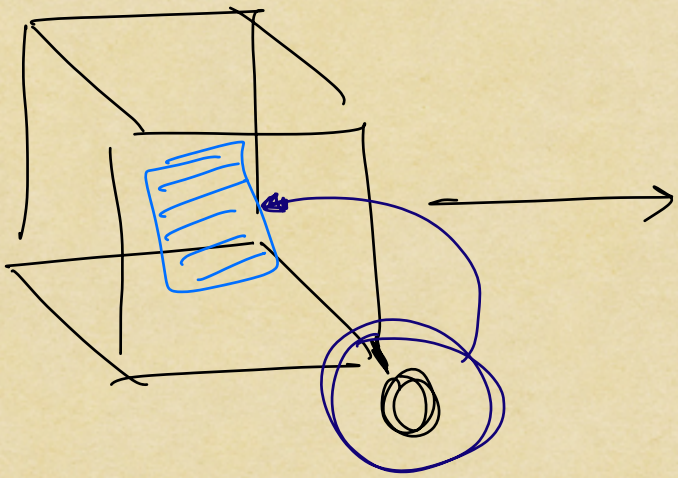




⇒ Getter/Setters

* private for all attributes [majorly]





i) protect the document from others → private

ii) provide some way for specific people to access the document. → getter / setter

getter → read

setter → writing

* Constructor

↓

Encapsulation → abstraction

Java

- primitive → int, boolean, char, long, double
- reference → object
- special → String

```
class Student {
    String name;
    int marks;
    int rank;
    Address address;
}
```

primitives

address → reference / object

```
class Address {
    String floorNo;
    String flatNo;
    String buildingName
    String street;
    String -
```

Saturday 8:30 PM