OOPs classes

① Pillars of OOPs
objects & classes
access modifiers
constructors

} today's Agenda

② constructors
deep copy vs shallow copy
static keyword.

③ Inheritance
polymorphism

④ abstract classes
interfaces
final keyword.

Example → Real life
Definition
additional information.

Intro to OOPs

OOPs → Object Oriented Programming .

A paradigm of programming
↳ fundamental Style .

procedural programming
↳ procedures → a set of instructions
↓
functions

$f^1() \rightarrow f^2() \rightarrow f^3()$ $\boxed{C}$ language

printDetails(name, roll no, age, batch)
↓
Struct student {
name
roll no
age
batch.
}
→ property

↙

printDetails (Student s)
verb subject.

Akash is teaching
Students are learning
Some student is sleeping
I will go to work.

subject verb.

Someone is doing something.

Student s;
s.printDetails( ). → behaviour

class (entity)

properties behaviour → Control.

Readable
understandable
extensible
maintainable.

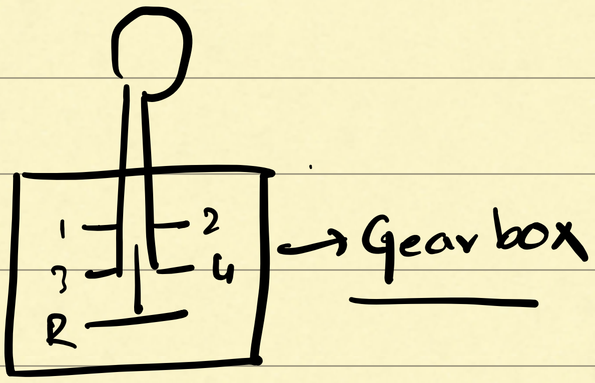
} final targets

Pillars of OOPs

Abstraction → Principle

$$\left.\begin{array}{l}\text{Inheritence}\\ \text{Polymorphism}\\ \text{Encapsulation}\end{array}\right\}$$ Pillars ↑ supports .

Abstraction



. sort ( )
↓
comparator.

- Good abstraction

- representing complex systems in terms of ideas.
  ↳ class /entity

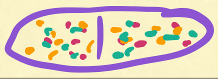
- Not showing unnecessary details to others

```
class Bird {      ← example
    #wings
    colour
    weight
    typeOfBeak
    fly()
    eat()
    makeNest()
}
```

Example of polymorphis inheritance

Encapsulation

↳ capsule →



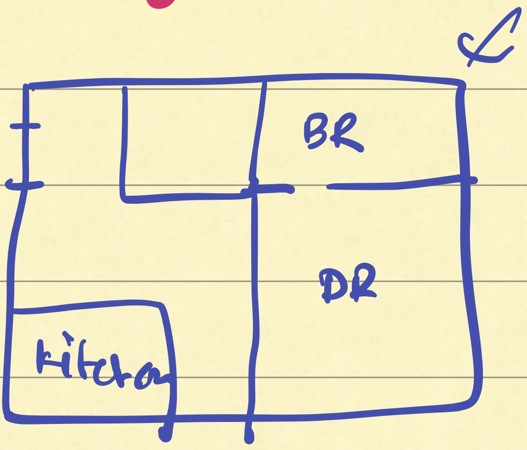
- To hold all the things inside. (Class)
- Protect it from outsid environment.
  (Access modifiers)

- Class encapsulates the properties & behaviours together

- Accen modifier protect the properties & behaviours from illegal accen.

Class → A blueprint of your entity.
→ a custom datatype.

```
class Student {
    name;
    rollno;
    batch;
    psp;
    attendClass();
    solveAssignment();
    pauseBatch()
}
```



Class → entity
defined

Object → class
existing (memory)

int $\boxed{x}$ = 5

Refernce data type

Student s = new Student();

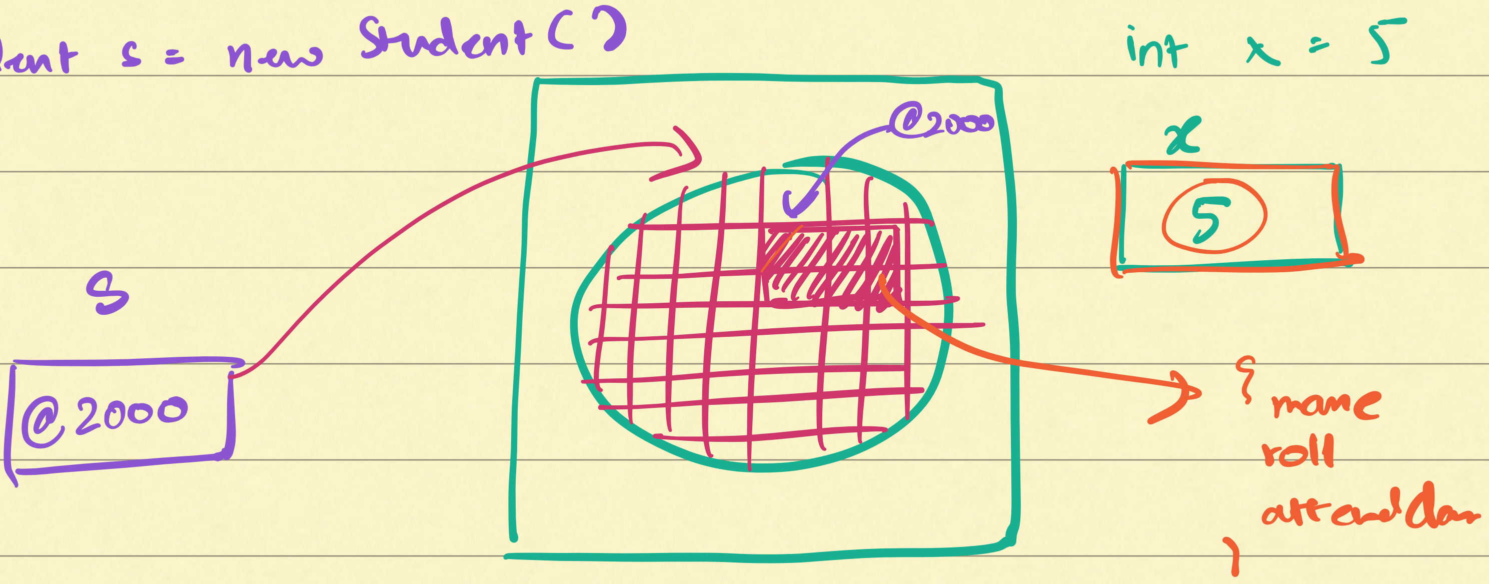
s → variable name.

new → new operator (allocating memory)

Student() → className

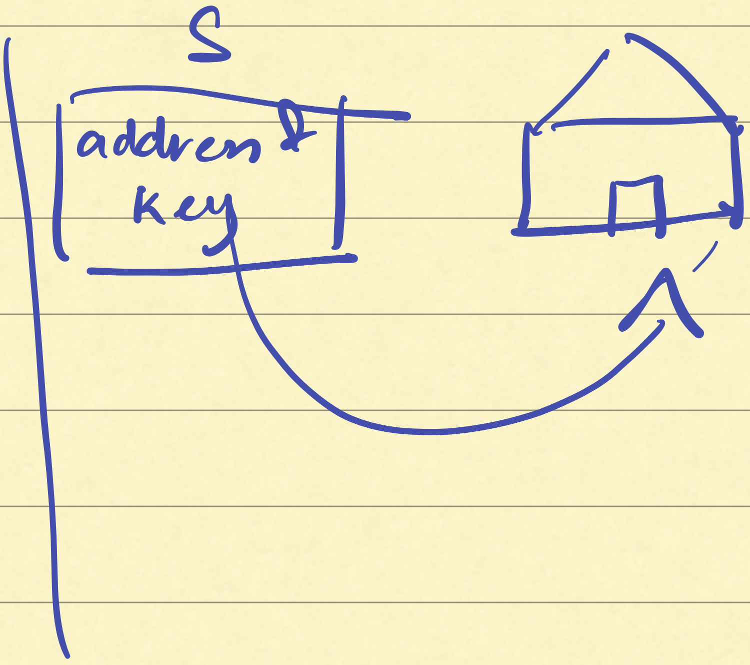
student s = new Student()

int x = 5



s.name

s. attendClass



POTUS

access to nukes

nuclear bombs

nuclear launch codes

Break till 8:27

```
int x = 10;
int y = x;
y = y+10;
print (x);
```

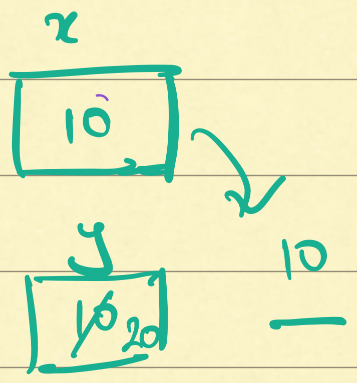
↙

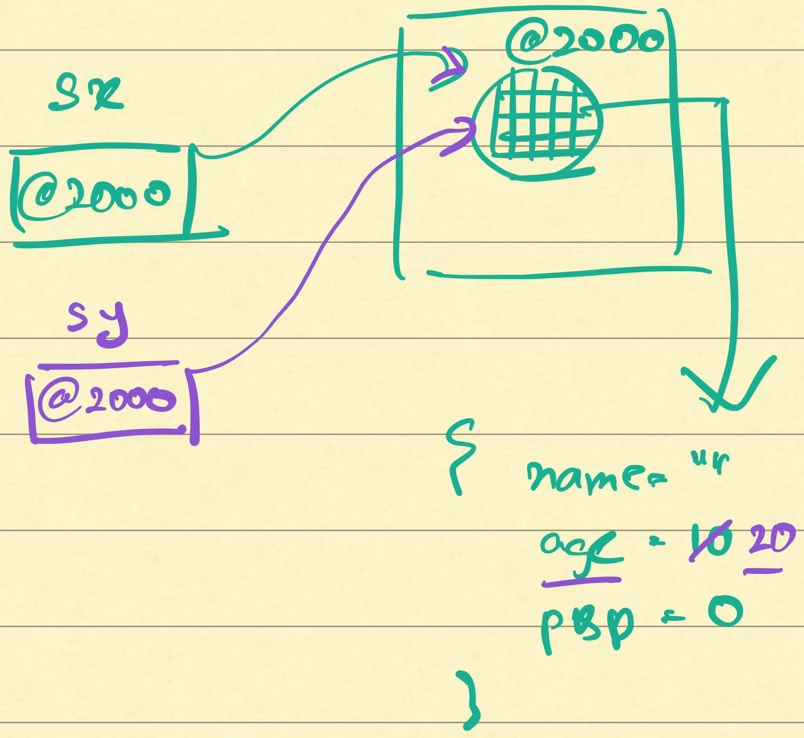
10

```
Student sx = new Student
sx.age = 10;
Student sy = sx;
sy.age = sy.age +10;
print (sx.age)
```

↙

$\boxed{20}$





Access Modifiers

public	public int x = 10;	accessed from anywhere
private	private float y = 0.0;	accessed only in it's own class
protected	protected char z = 'x';	accessed only with in th package
default	int z = 10;	accessed only with in the package

**

Inheritance

can be accessed outsid
package with in a
child class.