Agenda

Next 4 classes → OOPS

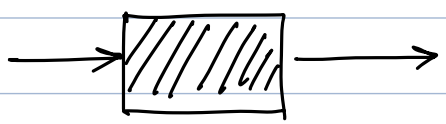
1) classes & objects
2) references
3) access modifiers
4) constructors.

deep / shallow
static
Inheritance
polymorphism
Abstract, Interfaces
final
[Inner classes + Enums.

C → procedural

procedure
↳ a set of instructions
f^n / method.

f^n 1 → f^n 2 → f^n 3



Mohit is Teaching
subject + verb.

printStudentDetails (name , batch , psp ...)

{

⋮

}

↓

struct

printing is happening on Student

mohit.print ( ); → Mohit is printing the data

Subject / entity has the control.

{ different paradigms

OOP

Abstraction]

Encapsulation

Inheritance

Polymorphism.

Abstraction

just knowing necessary details is enough to work.

comparator ✓

sort ([ ] ,)

Encapsulation

↓

Capsule

① holds (classes) the medicine together

② protects the medicine

↓

access modifiers

↓ classes

Student

name
age
psp
gradYear
⋮

- attendClass()
- rateClass()
- solve()
- attempt()

Bird

- type
- #wings
- color
- country

fields ↑ state

methods ↑ behaviour.

- eat()
- fly()
- makeSound()
- dance()
- makeNest()

class → blueprint of entity / custom datatypes

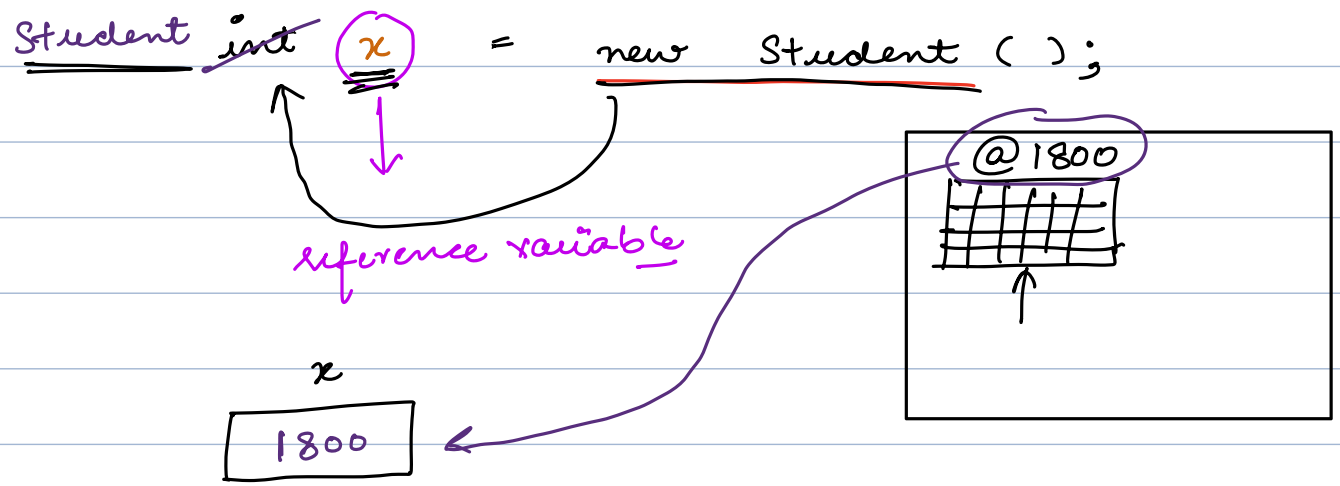
```
class Student {
    int id;
    String name;
    int gradYear;

    void attendClass() {
        ==
    }
    :
}
```

class → existence → object

int x = 10;

Student int x = new Student();



x has the object

✓ x knows about object

x . ____ access the fields/methods

```
x.name = "Mohit";
x.age = 26;
```

nuclear bombs → code → President of America.

Person2

paint
carpenting

↳ create the copy of house

Copy of key

Day 1	Day 2
painter	carpenter
blue → yellow	yellow

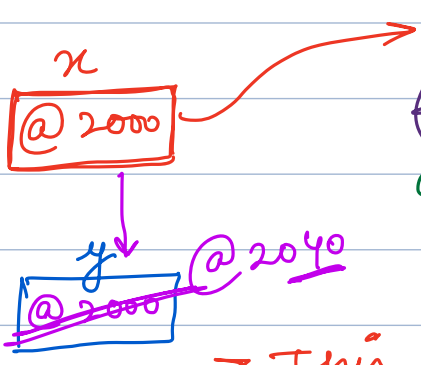
```
Student x = new Student();
x.age = 26;
```

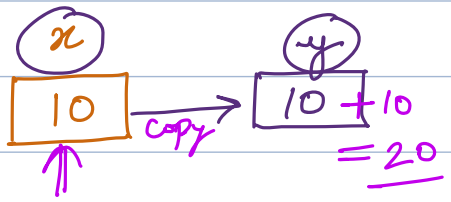
```
int x = 10;
int y = x;
```

```
Student y = x;
y.age += 10;
print(x.age);
```

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

ans = 10

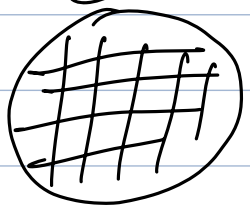




This doesn't create copy of obj, instead copy of ref.

y = new Student();

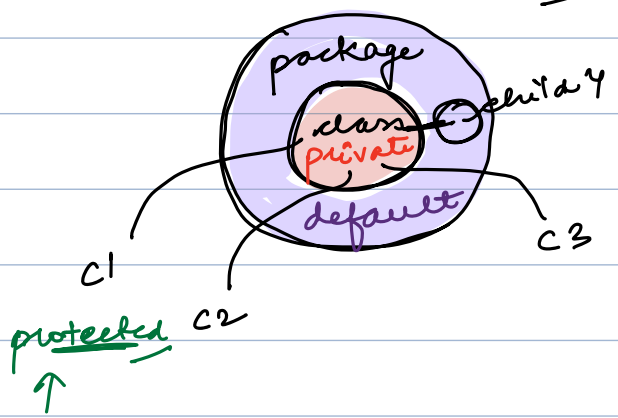
@2040



Break: 10:12 pm

public

world



	class	package	outside child	world
Public	✓	✓	✓	✓
protected	✓	✓	✓	✗
default	✓	✓	✗	✗
private	✓	✗	✗	✗

default + child less
private + package

```
Student s = new Student();
```

call method / fn

constructor : default values to your attributes

If & only if you don't write your own constructor, java will provide you one.