OOPs - 4

Access Control

→ Who can access which variables, methods or classes in our program

Public

Accessible from anywhere —

```
public int num;
```

private

Accessible only inside the same class.

```
private String password;
```

>> we use getter - setter to access & modify these values.

protected

Accessible in same class & same package & sublasses even in diff packages.

```
protected int marks;
```

Default

then it's package-private, means only can access in same package.

Ex

```
int roll_no;
```

	Class	Package	Sub class (same pkg)	subclass (diff pkg)	Everywhere
public	✓	✓	✓	✓	✓
protected	✓	✓	✓	✓	✗
private	✓	✗	✗	✗	✗
Default	✓	✓	✓		

diff pakage — com.ujjawal.access —

A.java

```
public class A {
    public int num;
    private ~~String~~ pass;
    protected int age;
    A(int num, ~~String~~ int age){
        this.num = num
        this.age = age
    }
}
```

& diff package — com.ujjawal.test

subclass.java

```
import com.ujjawal.access.A;
public class subclass extends A {
    public subclass (int num, int age){
        super (num, age);
    }
    public static void main (){
        subclass obj = new subclass (45, 18);
        int n = obj.num;
```

✓ work as it's subclass

but if try to run

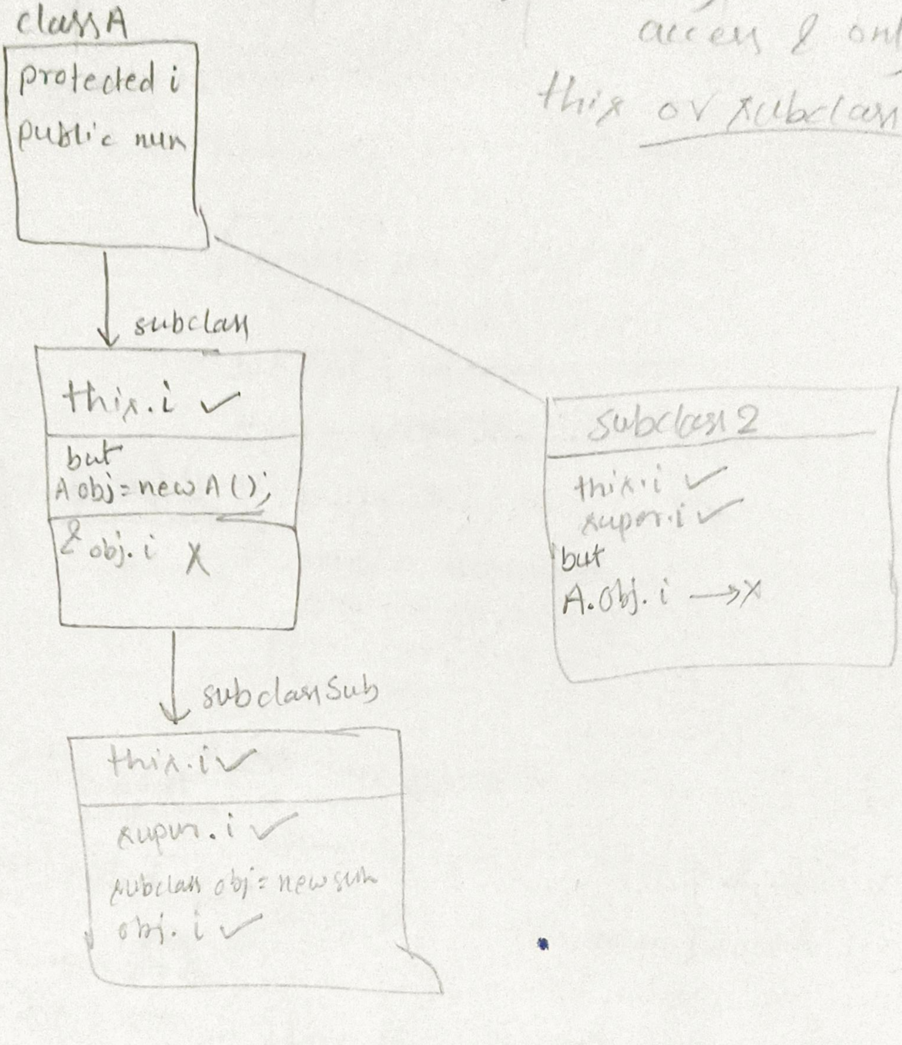
A obj = new A (10, 22);

int n = obj.age // error

why → only subclass knows what A has even in diff package but A itself cant use it's object to access protected

Thumb Rule for protected →

inside same pkg → act as default
outside same pkg → only subclass can access & only through this or subclass ref.



Packages
- → user-defined → ✓ done last video
- → In-built
  - lang — Imported Automatically
  - io — input/output
  - utils → collection
  - applet
  - awt & swings
  - net

Object class ? → root(super) class of all classes.

it's in java.lang & got extended by each class indirectly

how can each class extends Object class when multiple inheritence is not possible in java?

→ Here is how →

$$\begin{tikzcd} A \arrow[r] & \text{object class} \end{tikzcd}$$

$$\begin{tikzcd} B \arrow[r] & A \arrow[r] & \text{object class} \end{tikzcd}$$

$$\begin{tikzcd} C \arrow[r] & B \arrow[r] & A \arrow[r] & \text{Object class} \end{tikzcd}$$

So basically, we use single inheritence & uses multilevel for that.

hashcode method ?

public int hashCode() {. . . }

just some random stuffs to identify two same valued object differently ——

Ex

```
ObjectDemo obj1 = new ObjectDemo(12, 13)
ObjectDemo obj2 = new ObjectDemo(12, 13)
ObjectDemo obj3 = obj1;
```

Now

```
SOUT (obj1.hashCode());
SOUT (obj2.hashCode());
```

Diff hashcodes for each

we can override it —— - - - - see code - - - -

what is .equals() & "==" ?

.equals() → compare obj reference

"==" → for premitive compare values & for Objects compare ref

like $\boxed{\text{obj1.equals(obj2)}}$ → false

But $\boxed{\text{obj1.equals(obj3)}}$ → true

same as == But if override

~~@override~~
~~boolean equals(Object obj)~~
~~{ Objects [illegible]~~

Next page

```
@overide
public boolean equals (Object obj){

    ObjectClass other = (ObjectClass) obj ;
    return this.num == other.num;

}
```

Now if we do —

SOUT (obj1.equals (obj2)) → true
because values of num is same

SOUT (obj1.equals (obj3))
↳ true as before

But

SOUT (obj1 == obj2) → False

VVI.com

By default .equals() is same as "=="
unless you override it by your needs.

There are more —
- ↳ instance of
- ↳ get Class

etc