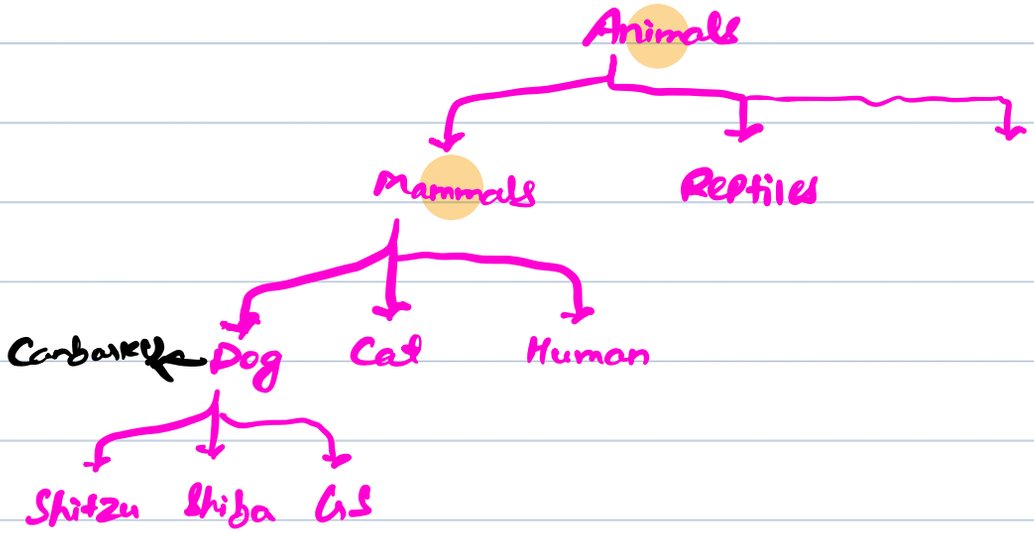
Today's agenda

- Inheritance
- Constructor Chaining
- Polymorphism
  - method overloading

each class → entity

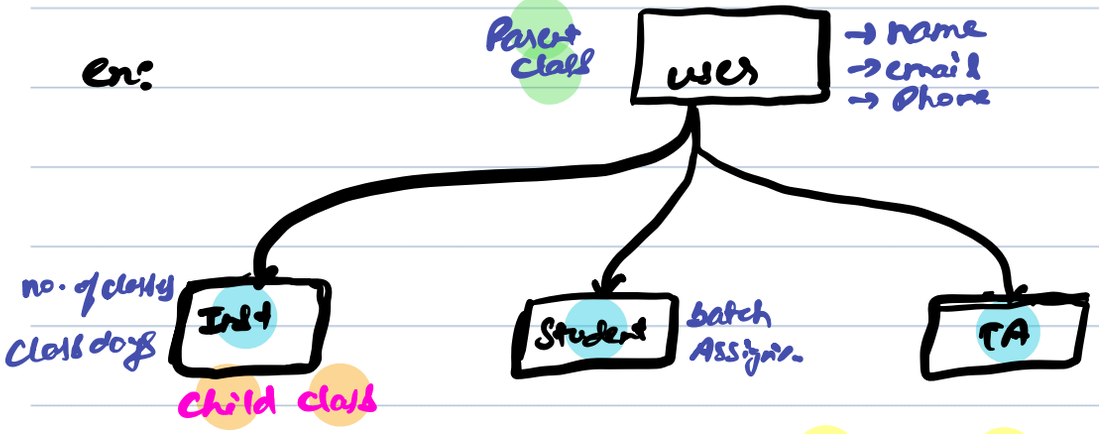
// Inheritance



→ all the child class will have features of their parents but not vice-versa.

↳ Inheritance is like assigning hierarchy to class.

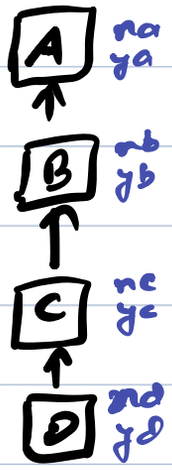
Ex:



↳ Parent - Child relation

↳ "extends" → keyword to inherit Parent class.

Constructor Chaining



D obj = new D();

↓ triggers default constructor

D() → C() → B() → A()

↓ initialize A attr.

↓ initialize B attr.

initialize C attr.

initialize D attr.

A.java *B.java C.java D.java *client.java

```java
package Student;

public class A {
    int xa;
    int ya;

    A(){
        System.out.println("Constructor of A");
    }
}

```

A.java *B.java C.java D.java *client.java

```java
package Student;

public class B extends A {
    int xb;
    int yb;

    B(){
        System.out.println("Constructor of B");
    }
}

```

A.java *B.java C.java D.java *client.java

```java
package Student;

public class C extends B {
    int xc;
    int yc;

    C(int v1){
        System.out.println("Constructor of C");
    }
}
```

A.java *B.java C.java D.java *client.java

```java
package Student;

public class D extends C {
    int cd;
    int yd;

    D(){
        super(10);
        System.out.println("Constructor of D");
    }
}
```

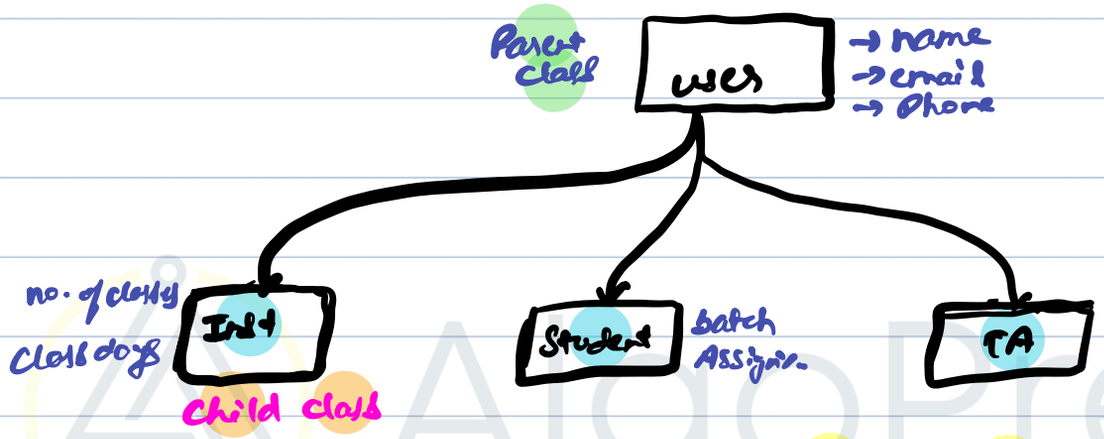
A.java *B.java C.java D.java *client.java

```java
package Student;

public class client {

    public static void main(String[] args) {
        //constructor chaining
        D obj = new D();


    }

}
```

Polymorphism

many form

How ⇒ ① Inheritance
② method overloading
③ method overriding



↳ Parent - Child Relation

int i = 10;

↳ User u = new User();
↳ Student s = new Student();

→ User u = new Student(); → Possible

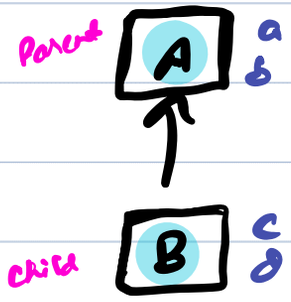
↳ u.email ✓
↳ u.batch?? → Can't be accessed

User u = new User();
new Student();
new Instructor();
new TA();

→ Student s = new user();

↳ Not Possible

Q.



A obj1 = new A(); ✓

obj2.b ✓ A obj2 = new B(); ✓

obj2.a ✗✗ B obj3 = new A(); ✗✗

* Method overloading → compile time Polymorphism.

↳ A class having multiple methods with same name (with diff set of Param).

```
class Student {

    void hello() {
        S.O.p("Hello!");
    }

    void hello(int i) {
        S.O.p("Hello2");
    }

}
```

Q)

```
class Student {

    void hello(int j) {
        S.O.p("Hello!");
    }

    void hello(int i) {
        S.O.p("Hello 2");
    }

}
```

✗✗

↳ Names won't be considered in method overloading

Q)

```
class Student {

    void hello() {
        S.O.p("Hello!");
    }

    int hello(int i) {
        S.O.p("Hello 2");
        return 0;
    }

}
```

✓

Q)

```
class Student {

    void hello() {
        S.O.P("Hello!");
    }

    int hello() {
        S.O.P("Hello 2");
        return 0;
    }

}
```

XX

↳ return type won't be considered during method overloading.