AGENDA

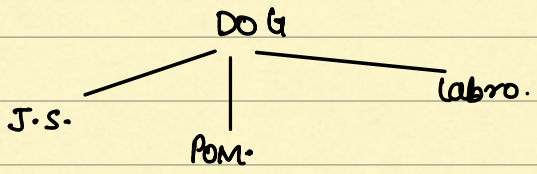
✓1.) Inheritance

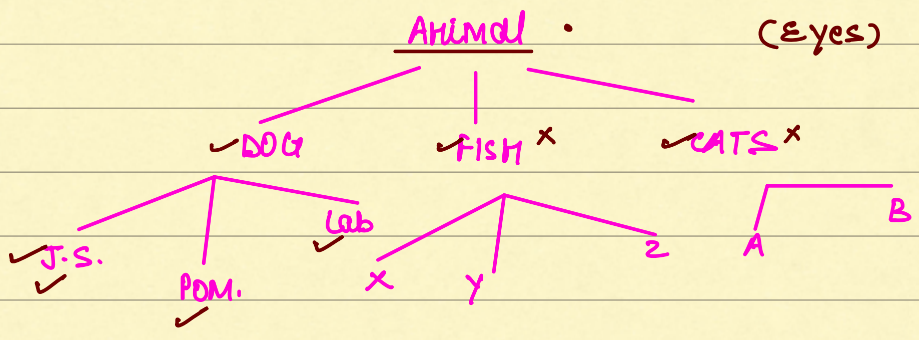
✓2.) Polymorphism

start by 9:05 PM

*) INHERITANCE:

"Representation of hierarchy in classes"

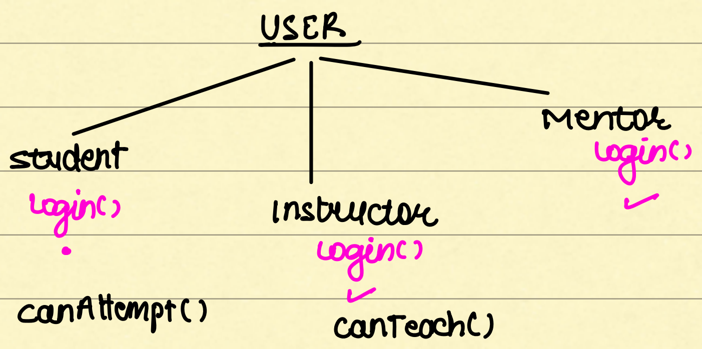




Inheritance.

eg: scaler

user → canlogin()



Parent class ⟶ superclass
child class ⟶ subclass

How do implement Inheritance:

```
class User {
        • email
        • Password

}

class student extends User {
        • psp



}
```

$$\begin{tikzcd} \text{USER} \arrow[d] \\ \text{student} \end{tikzcd}$$

```
student s = new student()
        s. emailId      ✓
        s. pwd          ✓
```

$$\begin{tikzcd}
\text{User} \arrow[d] \\
\text{student}
\end{tikzcd}$$

student s= new student();

*) HOW CONSTRUCTORS WORK IN INHERITANCE:

(superMost)

$$\begin{tikzcd}
\boxed{A} \arrow[d] \\
\boxed{B} \arrow[d] \arrow[u, bend left, dotted] \\
\boxed{C} \arrow[d] \arrow[u, bend left, dotted] \\
\boxed{D} \arrow[u, bend left, dotted]
\end{tikzcd}$$

```
A() {
}

B() {
}

    C() {
    }
```

D d= new D();

A > B > C > D

```
class  B {

        B() {          ✓

        }
        B(string s) {

        }
    }
```

Above concept is called: constructor chaining.

9:58 PM IST

★) POLYMORPHISM:

(best usecase of Inh.)

POLY - Many } - Many forms
MORPHS - forms } of same thing.

```
AuthenticateUser(User u) {
    .....
    ....
}
```

↓

```
AuthUser(List<User> u) {
    .......
    ....
}
```

Users → student/Mentor/Instructor

```
List<User> u = new ArrayList();
u.add(new Student());
```

u.add(new mentor());

✓ This is Polymorphism.

#
(**)

$$\begin{tikzcd} \text{Animal} \arrow[d] \\ \text{DOG} \end{tikzcd}$$

# ① Animal a = new Dog(); ✓

② Dog d = new Animal () ×

Note: Allowed to put child object in Parent datatype.

Animal→ a. height ×

a. name ✓

a. bark(); ✓ // DOG

a. walk(); × // Animal

*) TYPES OF POLYMORPHISM:

1.) compile

2.) Runtime

COMPILE TIME: / Method overloading

•) same method names for different purpose -

compiletime Polymor Eg:

#1.

```
A {
   greet(){
      "hi"
   }
}
```

```
B             {
      greet(){
         "hello"
      }
}
```

```
C Extends A {
   greet(){
      "bye"
   }
}
```

$$\text{list<A>} = [\overset{✓}{A}, \overset{✓}{B}, \dot{C}, \dot{A}]$$

.greet()

hi ✓

hello ✓

bye

hi ✓

#2.)

$$\begin{tikzcd}
\boxed{A} \arrow[d] & \text{greet()} \arrow[r] & \text{// hello} \\
\boxed{B} \arrow[d] & \text{......} & \\
\boxed{C} \arrow[d] & \text{greet()} \arrow[r] & \text{// hi} \\
\boxed{D} & \text{....} &
\end{tikzcd}$$

```
D  d= new D();
   hi
A  a = new D();

C  c = new D();
```