1. Inheritance
2. Constructor Chaining
3. Polymorphism
   - → Compile
   - → Runtime

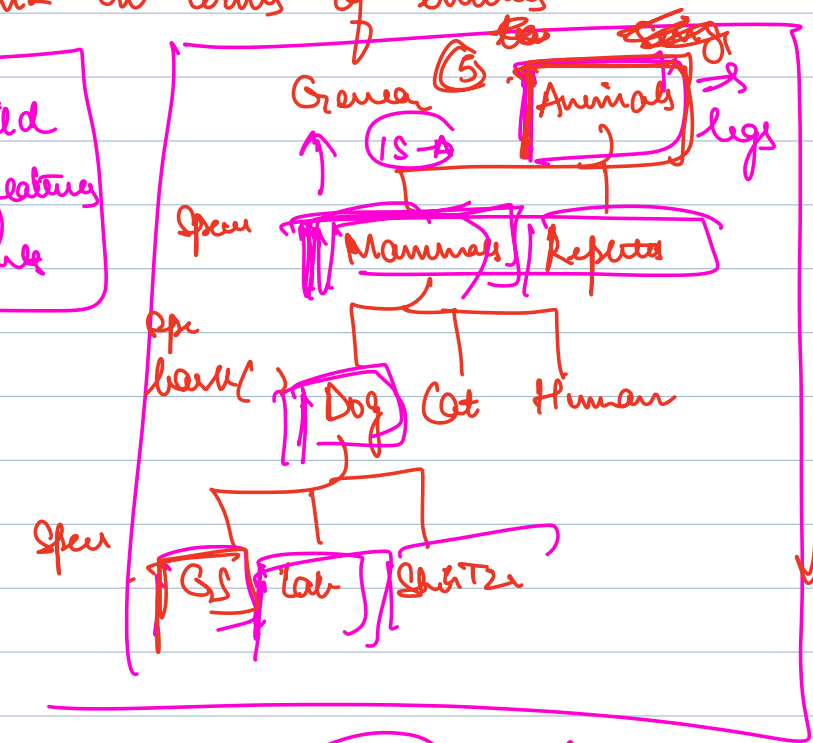
→ Head First Java : OOP
→ Head First Design Patterns
→ Spring Boot

IT
→ Gaurav Sen
→ Sudo Code

Inheritance

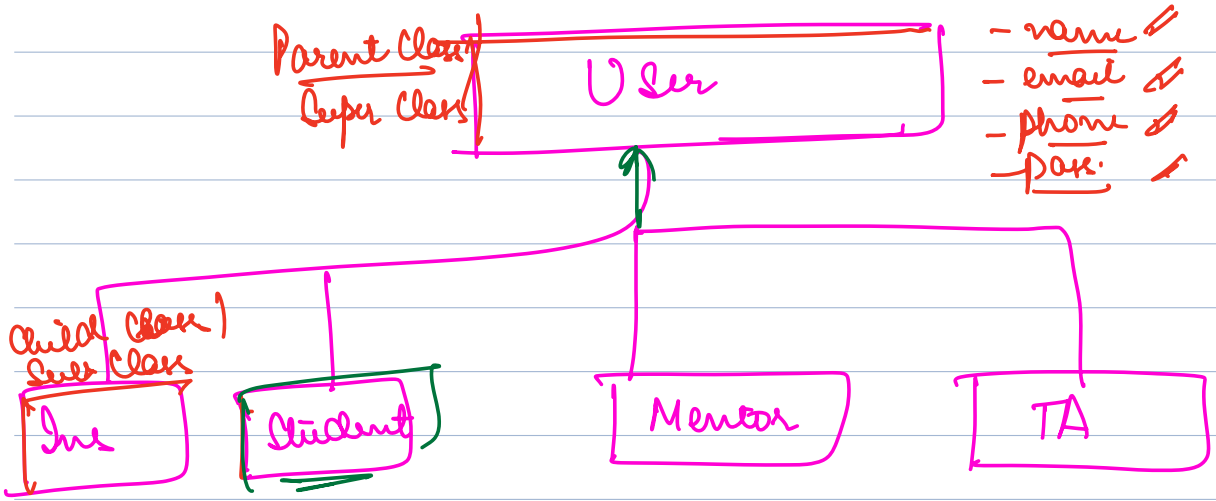
IRL think in terms of entities.

→ all derived/child entities have features of their parents



Dog → Mammal → Animal

exact kind of hierarchy can also be represented in OOP.



Inheritance

→ Rep of hierarchy b/w different entities.

→ Parent – Child Relⁿ

→ Super – Sub Relⁿ

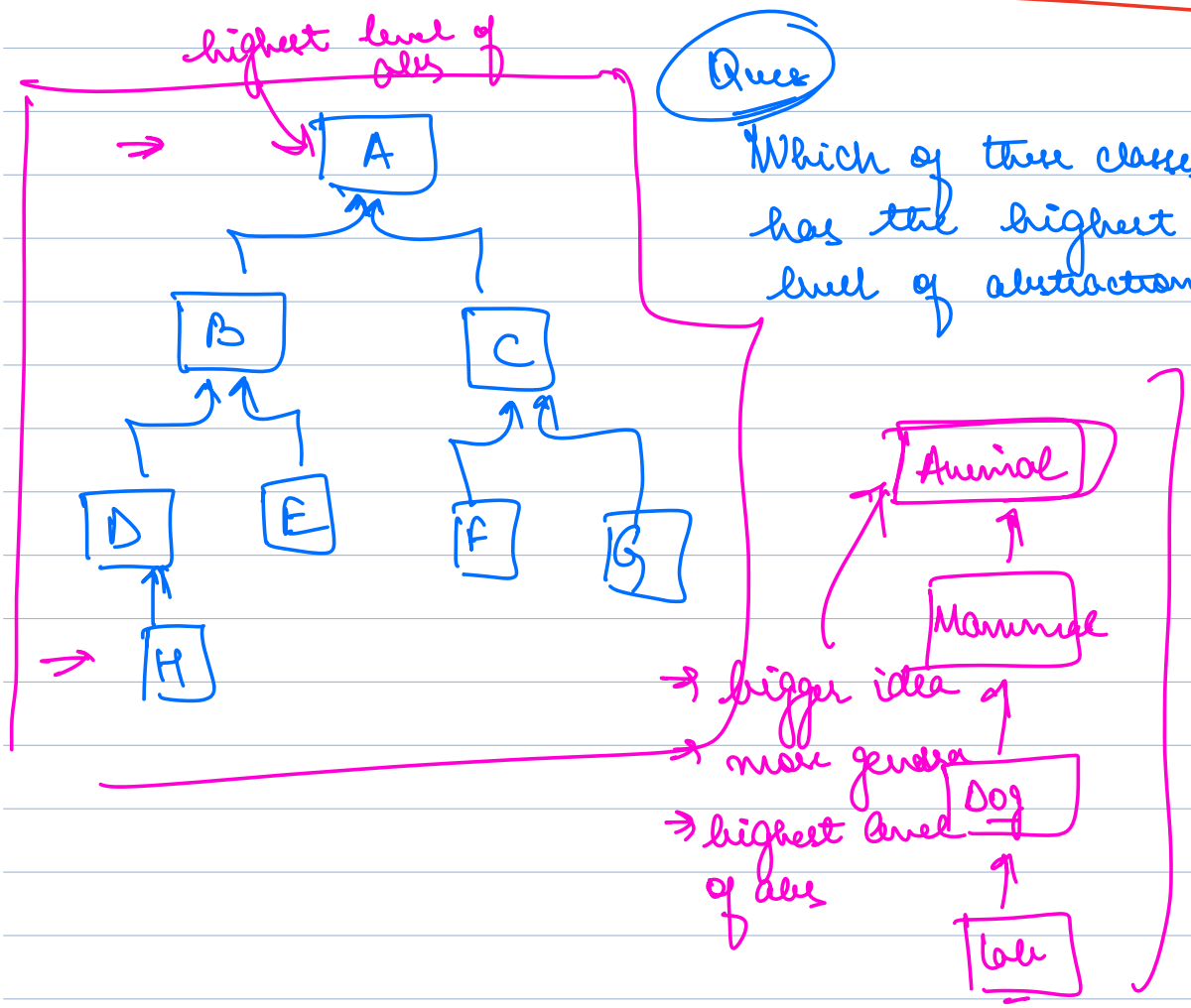
General – Specific Relⁿ

"extends"

```
class User {
    String name;
    String email;
    String password;
    void login();
    void logout()
    void joinMeeting();
}
```

```
class Student extends
        User {
    String batch;
    double psp;
    Mentor mentor;
    seeOpenJobs()
    solveAssgn()
}
```

⇒ all of the attrs and methods of parent class (even private) are also present in the child class. Child class may or may not add any new behaviours.



Ques

Which of these classes has the highest level of abstraction ?

→ bigger idea of more general

→ highest level of abs

```
class User {
    String name;
    String email;
    String password;
    void login();
    void logout()
    void joinMeeting();
}
```

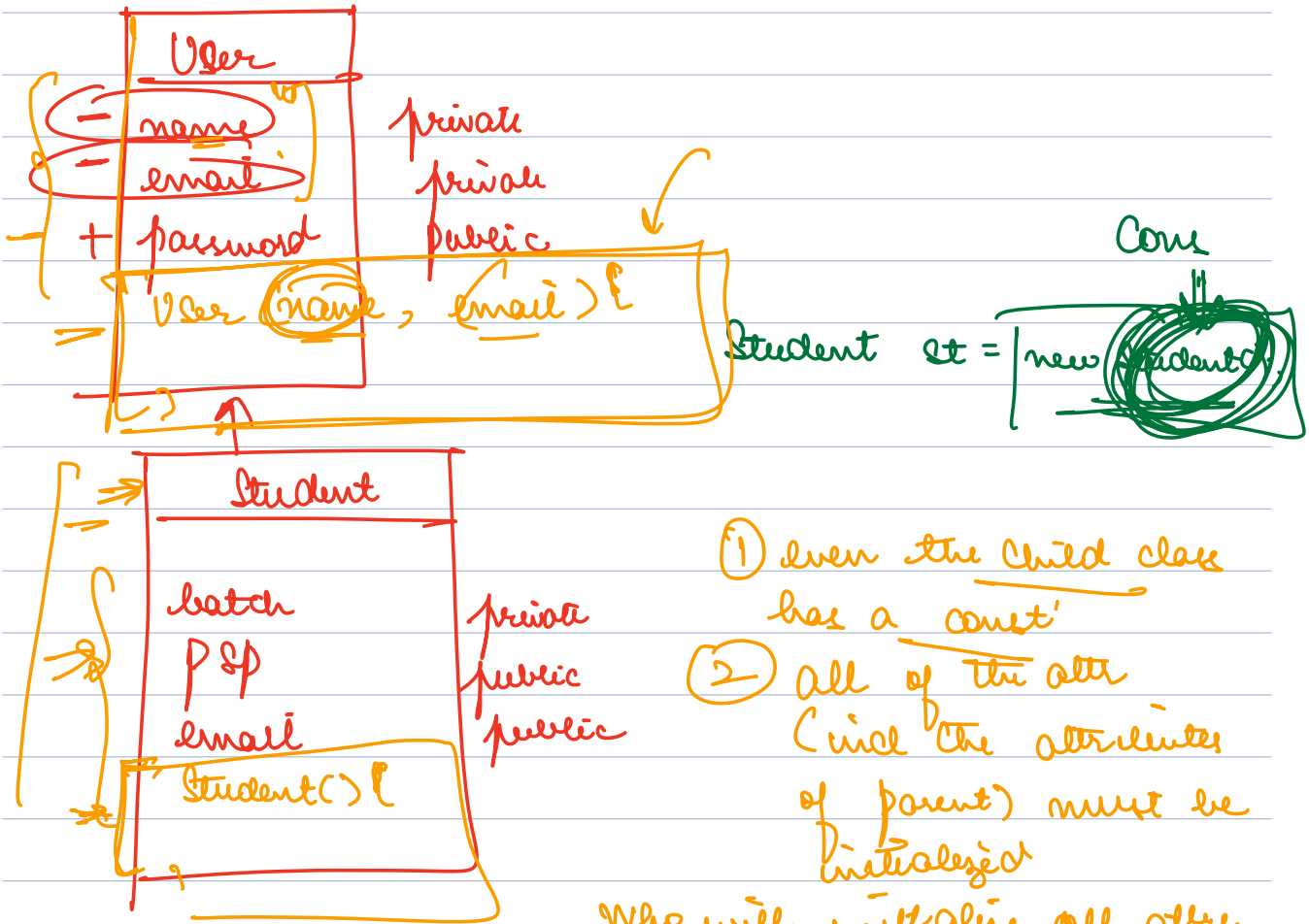
```
class Student extends
        User {
    String batch;
    double psp;
    Mentor mentor
    seeOpenJobs() =>
    SolveAssgn()
```

```
Student st = new Student()
st.login()
st.name = "Naman"
```

HOW ARE OBJECTS OF CHILD CLASSES CREATED

⇒ Are the private attrs of parent class present in the child class.

YES



(1) even the child class has a const'

(2) all of the attr (incl the attributes of parent) must be initialized

Who will initialize all attrs.

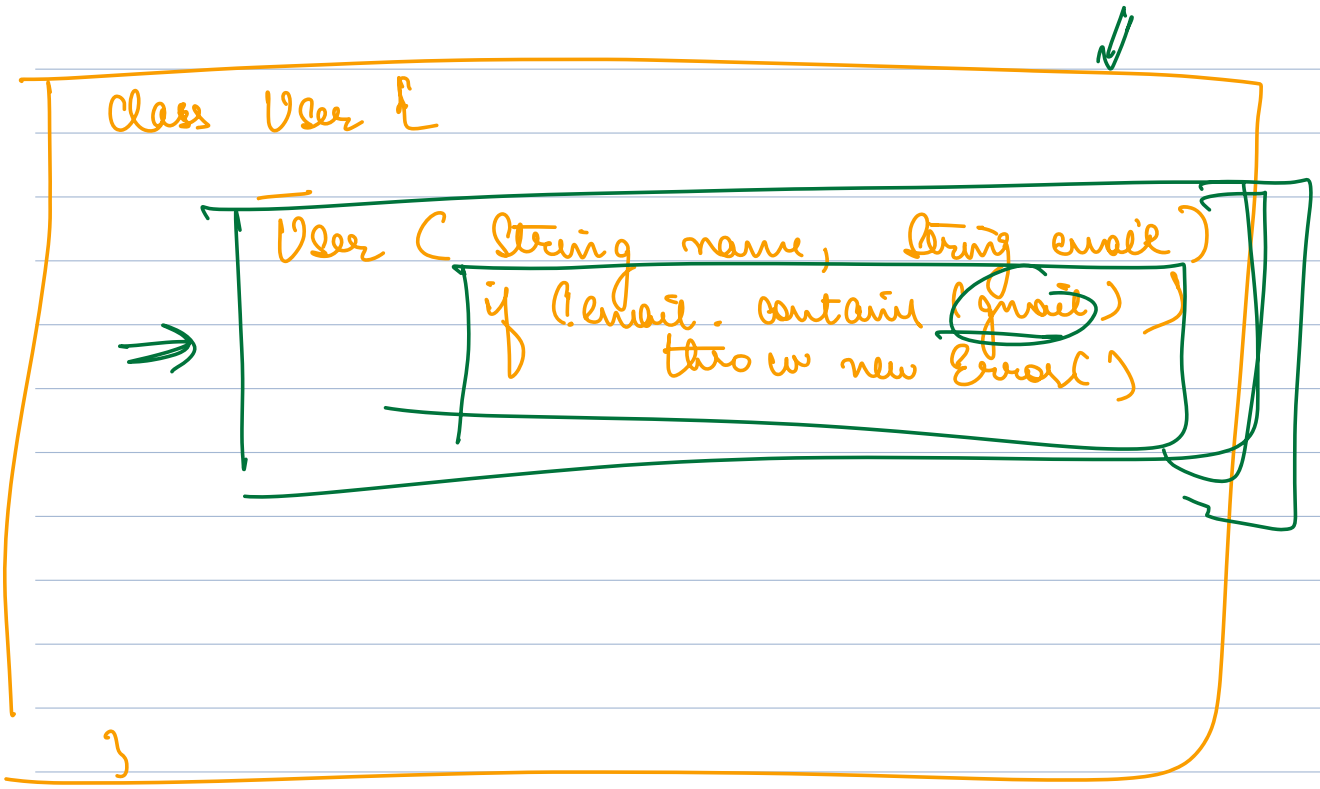
a.) Can the cons of child initialize all attrs of parent → NO ∵ it may not be able to access its private attrs

If all are public → Yes, cons of child can do that

But is that good? NO

Who should initialize attrs of parent?

cons of parent



Class Student {

Student (String name, String email,
String batch) {

this.email = email

}

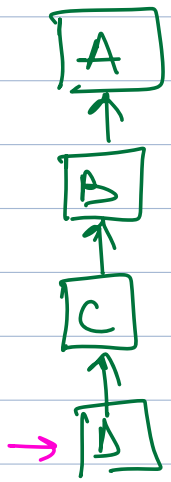
}

Student st = new Student( )

→ cons of child should ideally call cons of parent

CONSTRUCTOR CHAINING

→ How objects of a child class are created



D d = new DC

1. Client calls the cons of D
2. cons of D, even before starting to execute itself, call appropriate cons of its parent [new CC)]
3. C → new BC)
4. B → new AC)

[initialize attrs of A]

DC) → CC) → BC) → AC)

initialize D attrs ← initialize C attrs ← initialize B attrs ← initialized A attrs

DC) → CC) → BC) → AC)

DC) ← CC) ← BC)

[cons chaining]

class C() {

C(String a, String b){

}

}

class D extends C {

D() {

super("XYZ", "DBP");

}

}

must be the first line

You will not be able to extend the class UNLESS

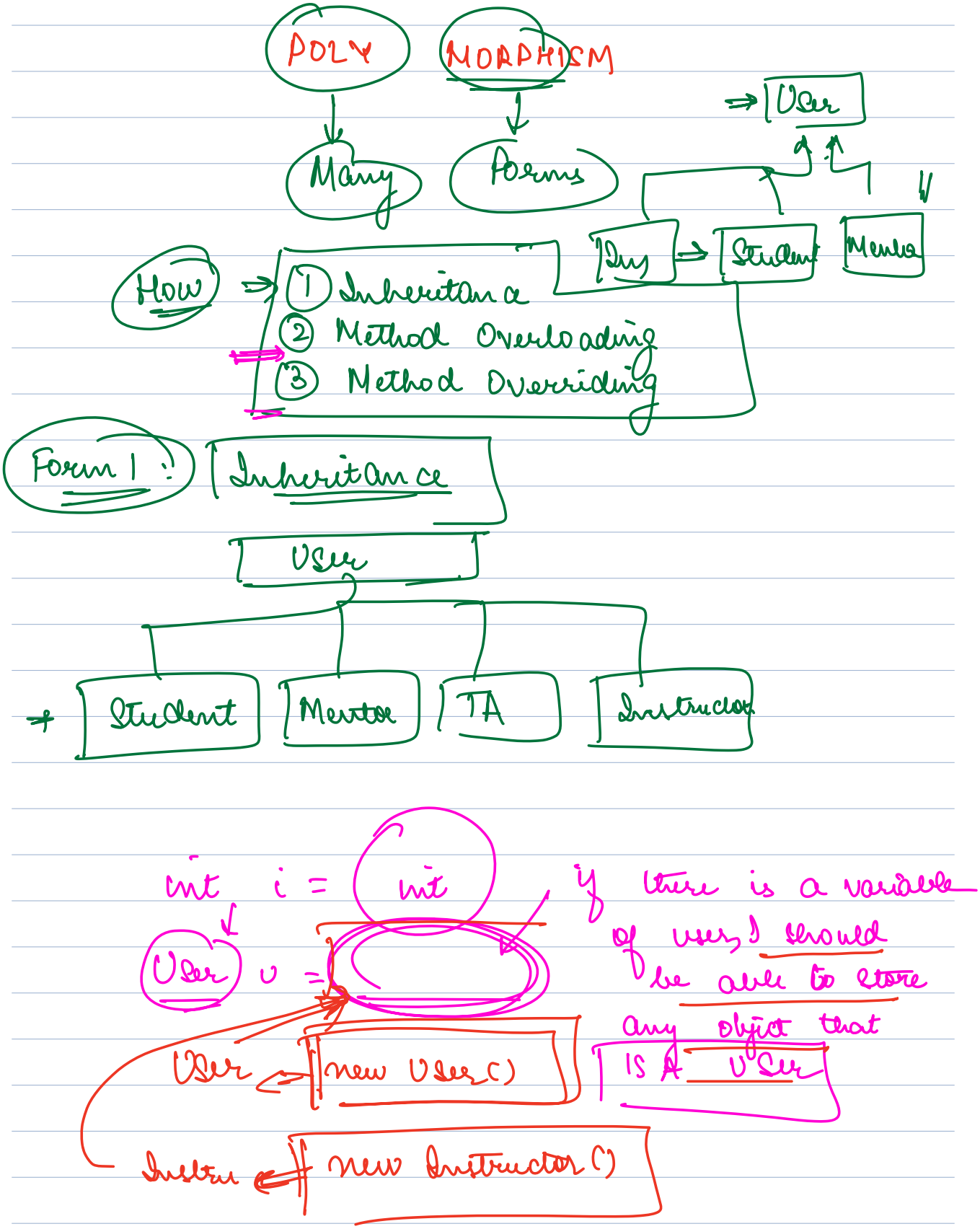
D() {

super("ABC", "DEF");

}

Break till 10:30PM

POLYMORPHISM



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

all of these objects are a user

List<User> = { new TA(), new Mentor(), new User() }

All of them are a User

⇒ A variable of parent class can store objects of child classes as well. because all properties of parent are going to be present in the child.

⇒ But you will be allowed to call only those methods that are declared in the data type (not in child)

psvm () {

User u = new User() / new Student() / new Mentor() ;

u. name =
u. email =
u. login ()

u. solve Assignment () ✗ compile time error

}

class XYZ {
int a
}

class User extends XYZ {
String name;
String email;
String password;
void login ();
void logout()
void joinMeeting();
}

class Student extends User {
String batch ;
double psp ;
Mentor mentor ;
seeOpenJobs()
solveAssignment()
}

User u = new Student()

u.eat();

u.seeOpenTabs();

→ only allowed to access attr and methods of data type

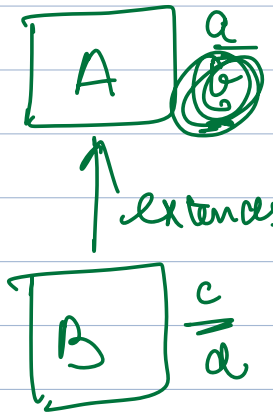
User u = new Student()

Student s = new User()

s.psp = 41;

Indian i = new Naman()

Naman n = new Indian()



A a = new A()

A a = new B()

B b = new A()

A a = new A()

a.a = xyz

a.c = Sbf

a = new ABC()

a.a = XYZ ✓

a.b = DEF ✓

a.c = XYZ ✗ compile error

→ Compiler will check the data type of variable

B b = new BC()

b.c = — ✓

b.d = — ✓

void doSomething(Obj o) {

o.pqr = 41; ✗ ✓

}

main() {

doSomething(new Student());

doSomething(new Mentor());

}

METHOD OVERLOADING

2 type of polymorphism:
1. compile time poly
2. runtime poly

→ compile time polymorphism

→ A class having multiple methods with the same name.
(with diff set of params)

```
class Hello {

    void hello() {
        sout ("Hello World");
    }

    void hello (String name) {
        sout ("Hello " + name);
    }
}
```

Method "hello" has multiple forms.
- one form that doesn't take any param
- one form that takes a param

```
Client {
    psvm() {
        Hello h = new Hello()
        h.hello(Naman);
    }
}
```

Which method to call is known by the compiler. ⇒ an example of compile time polymorphism

```
class Hello {

    void hello() {
        sout("Hello World");
    }

    void hello(String name) {
        sout("Hello " + name);
    }

    void hello(int i) {
        for(j=0; j<i; ++j)
            sout(Hello)
    }
}
```

hello (3) → hello
hello
hello.

Client {

psvm () {

Hello h = new Hello()

h. hello (3);

}

}

Method Overloading : More than one method with same name but diff signature.

Method Signature

void hello (int i) { }

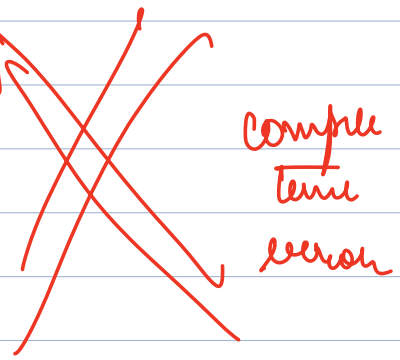
Signature: hello (int)

① Return type is not a part of method signature

② Param. name is not a part

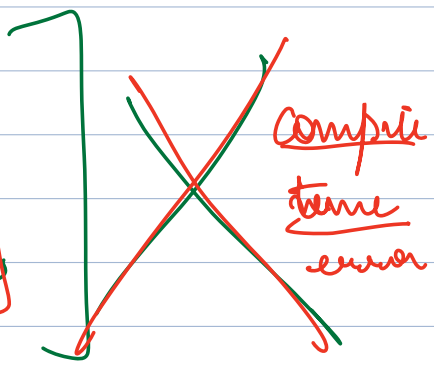
of signature

```
Hello {
    void hello(int i) { }
    void hello(int j) { }
}
```

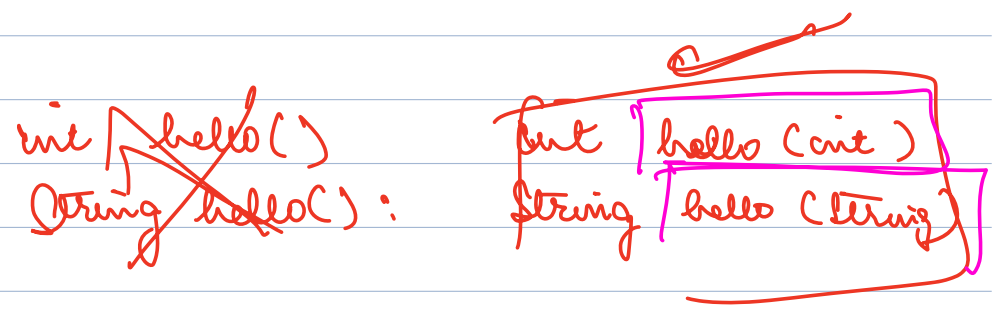


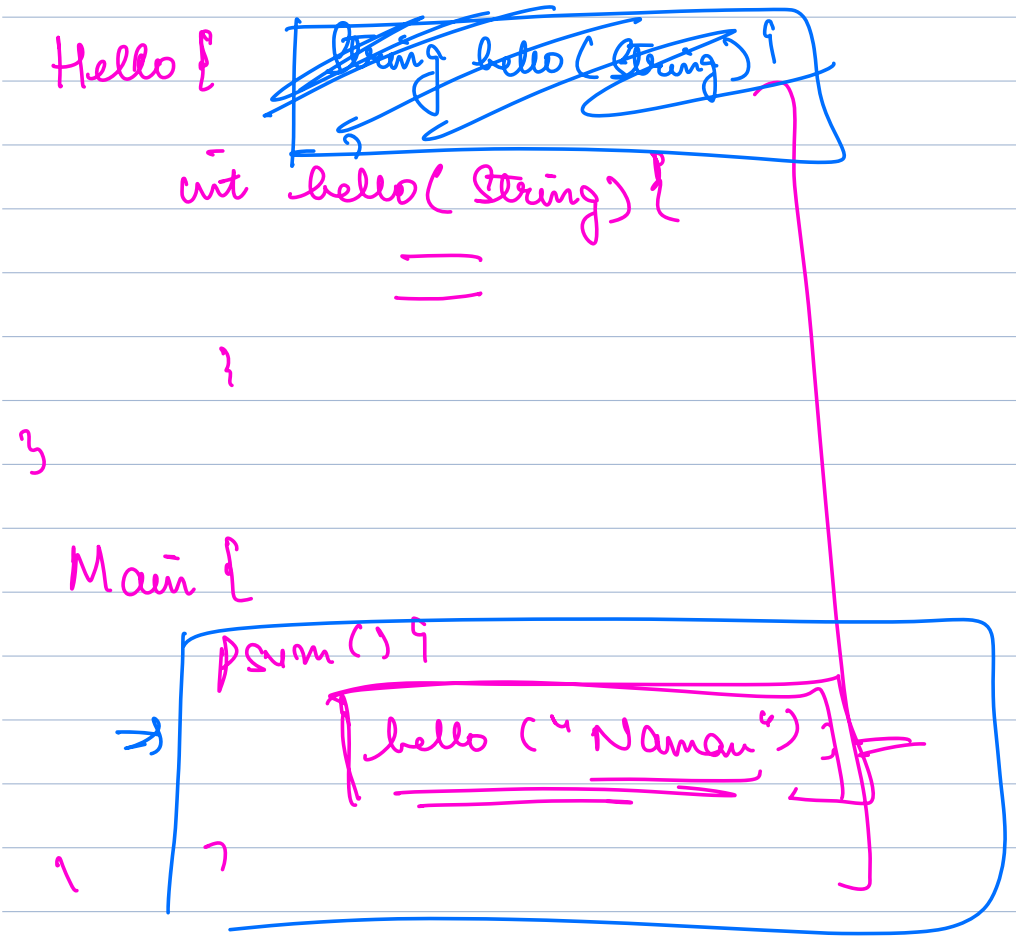
⇒ A class can't have 2 methods with same signature.

```
Hello {
    void hello(int i);
    Object hello(int i);
}
```



⇒ Return type doesn't contribute to method Overloading





Is this Method Overloading

void hello(int i)
void hello(int j)
} X

void hello(int i)
void hello(String i)
} ✓ long

void hello()
String hello()
} X CTE

void hello()
String hello(int j)

void hello(int a, String b)
void hello(String b, int a)

hello(int, String)
hello(String, int)

hello(int) ≠ hello(long)

hello(1) ⇒ int
hello(1L) ⇒ long

getA(Object)
getA(int)