Agenda

OOPs

① Principle → Abstraction

③ Pillars → Encapsulation
→ Inheritance
→ Polymorphism

Java → pass by value ✓ or pass by ref ✗

Q1

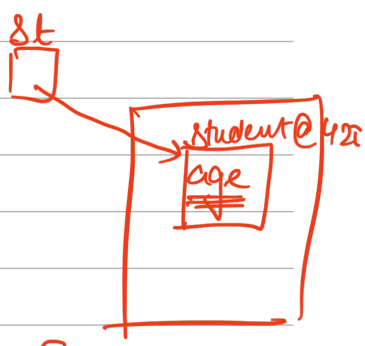
```
Main () {
    Student st = new Student();
    doSomething (st);
    Sout (st.age) : //40
}

doSomething (Student funcSt)
{
    funcSt.age = 40;
}
```

st → Student@420

[Student@420 funcSt = st = Student@420]

[Student@420.age = 40]



eg2 Main() {

✓ ① Student st1 = new Student();

② st1.age = 20; // Student@1.age=20

③ doSomething(st1); → Student@1

④ sout(st1.age); // 20

}

→ Student@1

Void doSomething(Student st2)

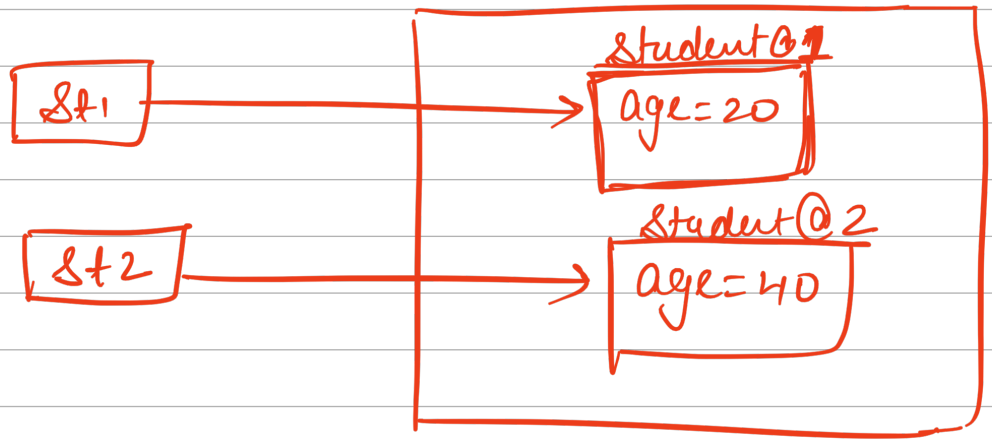
{ → Student@2

3.1 st2 = new Student();

3.2 st2.age = 40; // Student@2.age=40

}

Ref & Address



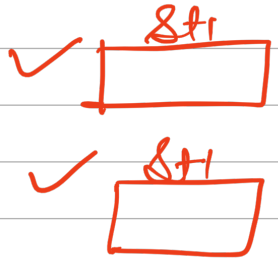
Void doSomething(Student st1)

{

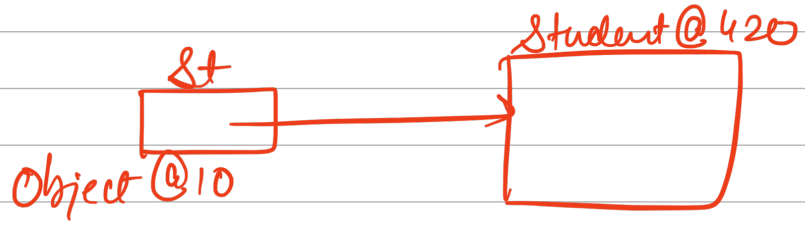
st1 = new Student();

st1.age = 40;

}




```
Student St1 = new Student
doSomething (st1)
SOut (St1.age); // 0
```




```
Student st = new (→
doSomethig (st);
```

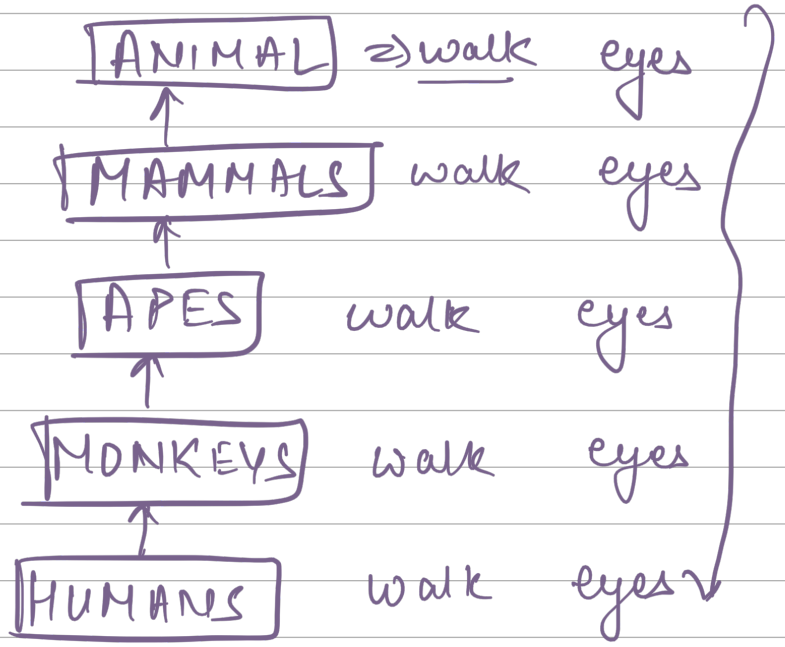
doSomething (Student st)

key = value

St = Student@420

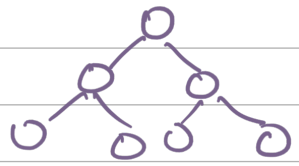
address object@10

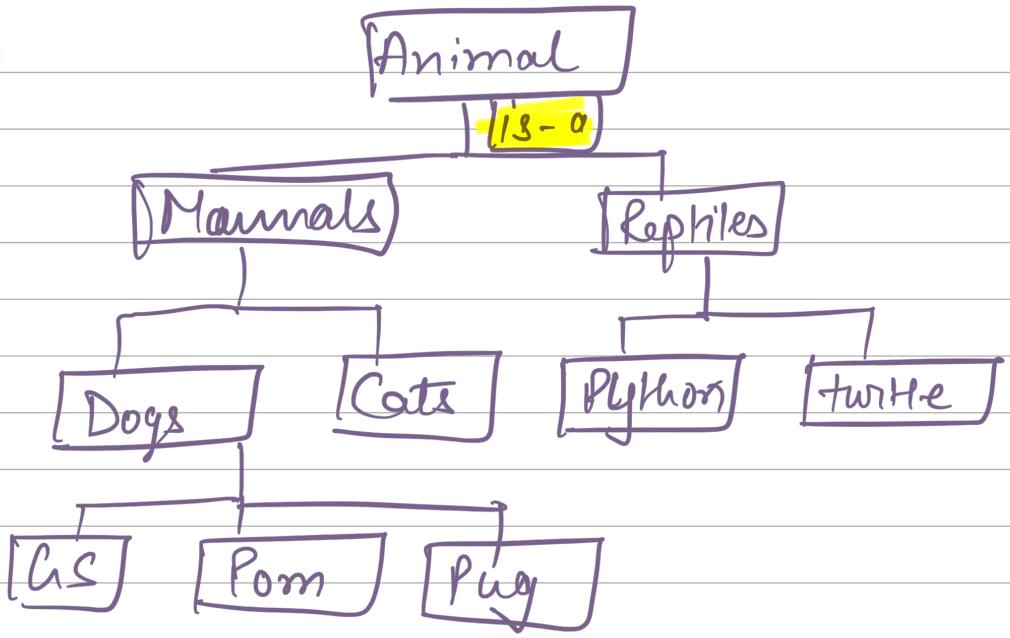
Inheritance



We form a hierarchy / relations b/w entities

OOPs also provides a way to form hierarchy ⇒ INHERITANCE





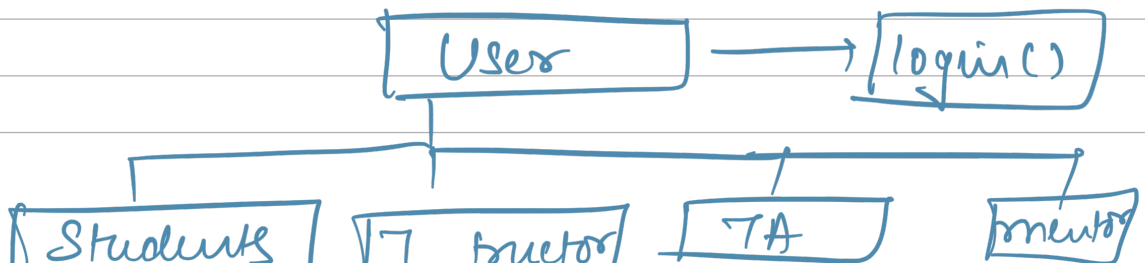
Pug is a dog, a dog is a Mammal,
a Mammal is a Animal

Inheritance ≡ IS-A Relationship

⇒ Inheritance allows us to share attrs and behaviors (Methods) with childrens.

Inheritance ≡ IS-A Relationship
≡ Parent - Child
↓ Superclass ↓ Subclass

Q4



↳ solveProblem() ↳ scheduleClass()

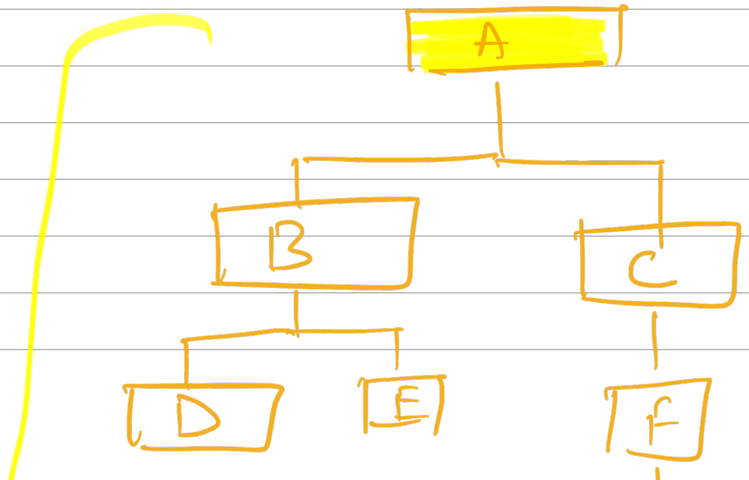
2) Child entity will inherit all attrs and behaviors of parent but it can additional attrs and behaviors

Eg

```
User {
    String username;
    String password;
}

Instructor extends User {        // inheritance
    double avgRating;
}

Instructor i = new Instructor();
{ i.username = "xyz";
  i.password = "abc";
```

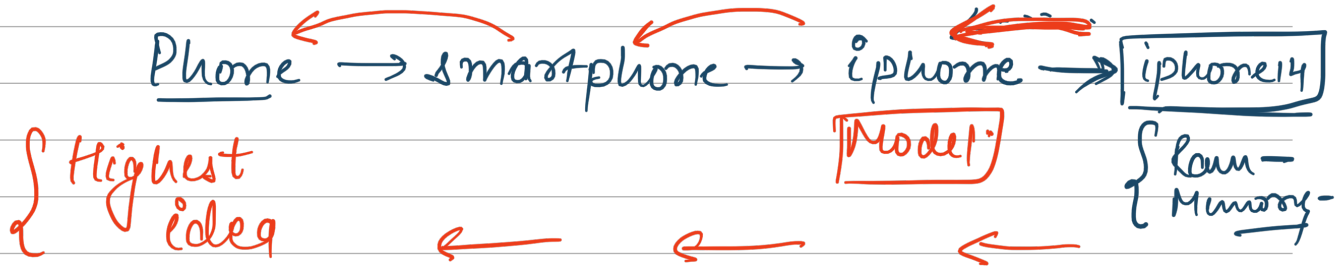


Q Which of the class has highest level of abstraction

↓

Ideas

highest level of Idea

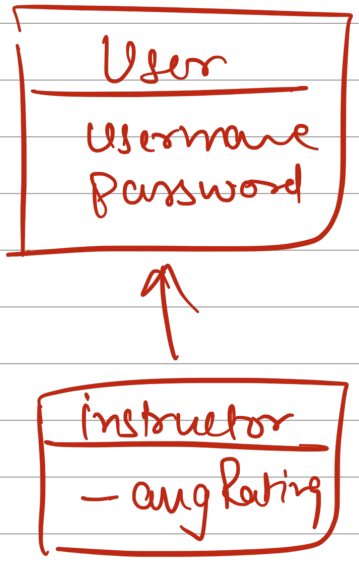


Problem :→ Who has born first
Chicken or egg

Atom →
Super class of any sub class

How the object of child gets created

Can child be created before parents
↳ NO



Instructor i = new Instructor();
i. username = "abc";
i. password = "xyz";

When we create a child object by default the default Constructor call parent default constructor.

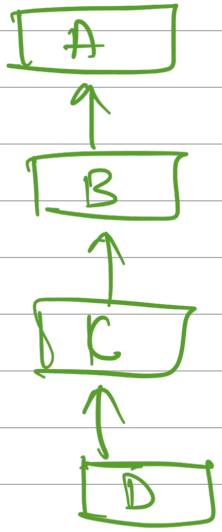
⇒ Parent object will be created before child.

⇒ Custom constructor you need to call

super();

→ is special ref to the constructor of immediate parent

Eg



B extends A

C extends B

D extends C

```
D d = new D();
```

Q1 Sequence of const called

D() → C() → B() → A()

Q2 sequence of object creation.

A → B → C → D

Chaining of Constructors

```
D(){
    super()  →  C(){
                    super()  →  B(){
                                    super()  →  A(){
                                    }
                                }
                }
    }
```

Class A {

A(int i) {

}

}

Class B extends A

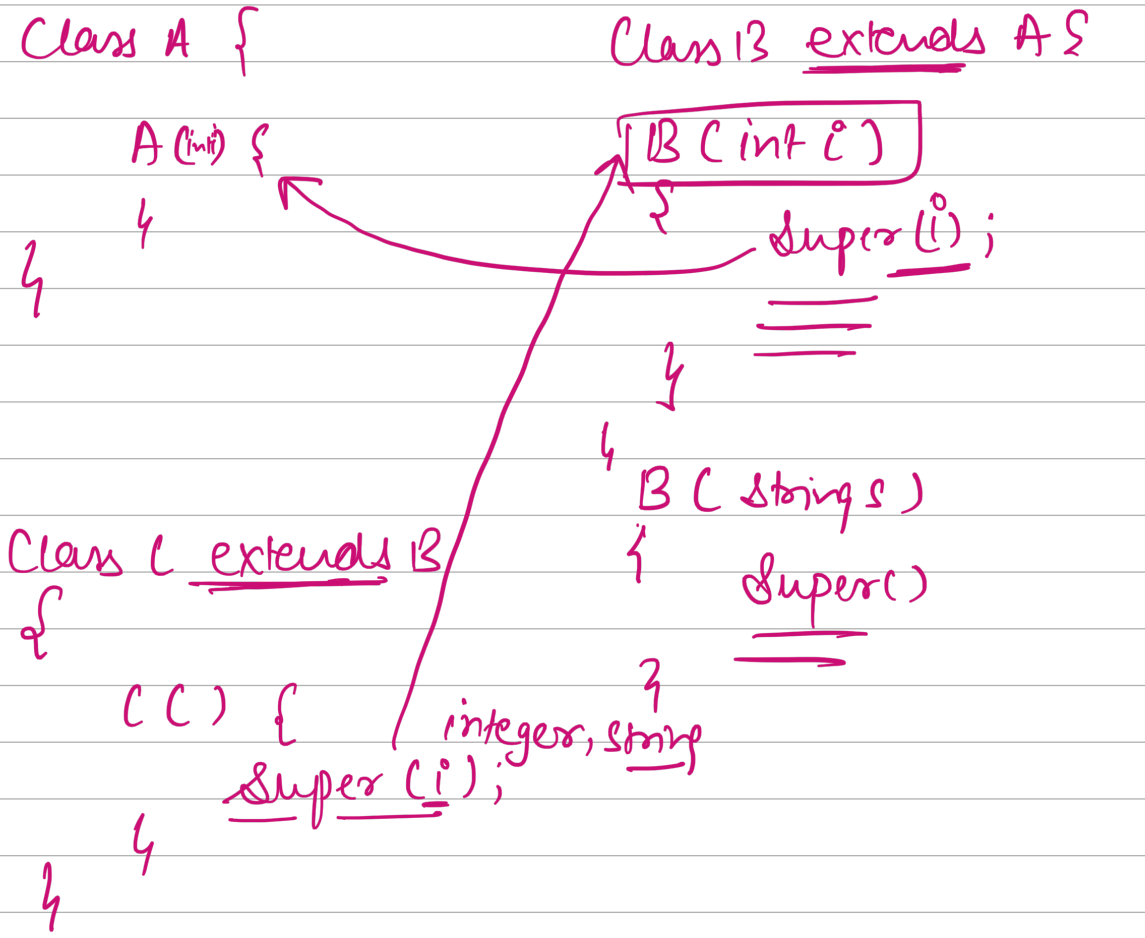
{

B() {

A a = new A(i); X

super(i);

}

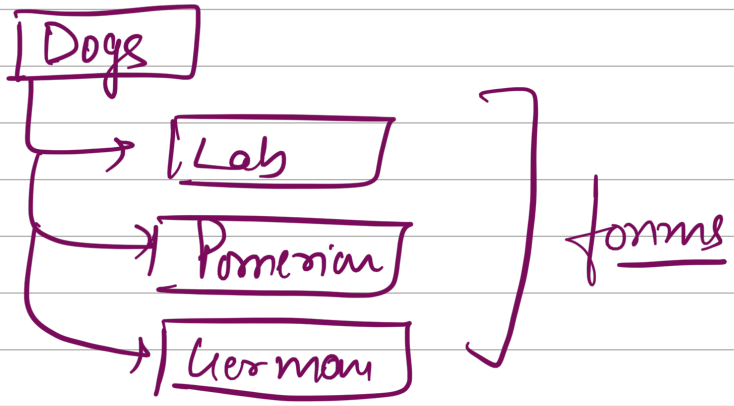


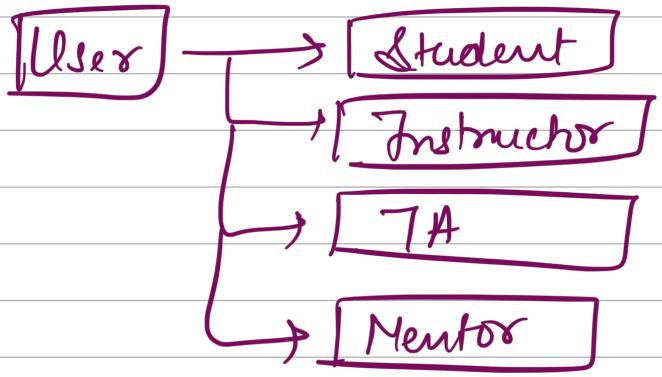
NOTE: Super() will always be first statement of Const

Poly Morphism

Many forms

Something which has multiple forms






```
printUsers ( List<X> users)
{

}
```

X → User → Student, TA, Mentor, Instructor

```
User u = new Student();   ✓
         new Mentor();    ✓
```

A parent variable can refer to Child.

[U]. solveproblem ✗ This is not allowed

```
List<Users> users
for( User u: users)        → Instructor / Students
{
    System.out.println(u.avgrating);
}
```

Assumption: All users are instructor

All instructors are users
but all users are not Instructors

NOTE: We can store an object of child class in the parent ref but vice versa is not possible

```
User u = new student()
but
Student s = new User() ✗
```

Q

```
Class A {
    int age;
    String name;
}

Class B extends A {
    String company;
}
```

A a = new B();

Q Is below statement correct

```
a.company = "Google"
sout(a.company)
```

Compile error

Eg 2

List → ArrayList
List → LinkedList

① ArrayList a = new ArrayList();

→ LinkedList

ArrayList a = a

② List a = new ArrayList();

⇒ More generic your code is, the better the readability and reusable

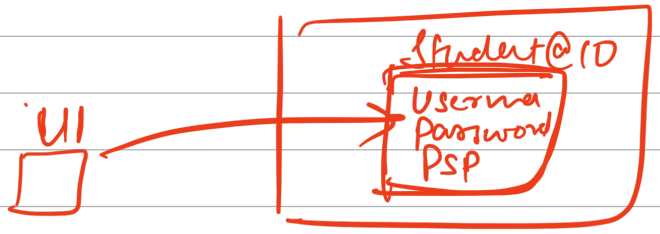
PolyMorphism

- Runtime
- Compile

UserName = null
Password = null

PSP = 0.0

```
User UI = new Student();
```



A

↳ A(int i)

B extends A

```
Bb = new B();
```

```
B() {
    super(i);
}
```

A()

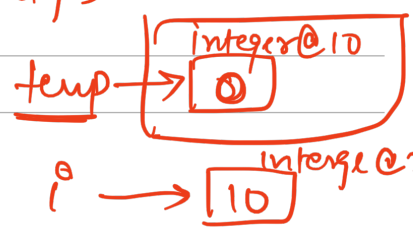
```
B(int i) {
    super();
}
```

```
doSomething(int i, int j)
{
    int temp = 0;
    temp = i;
}
```

i = 10

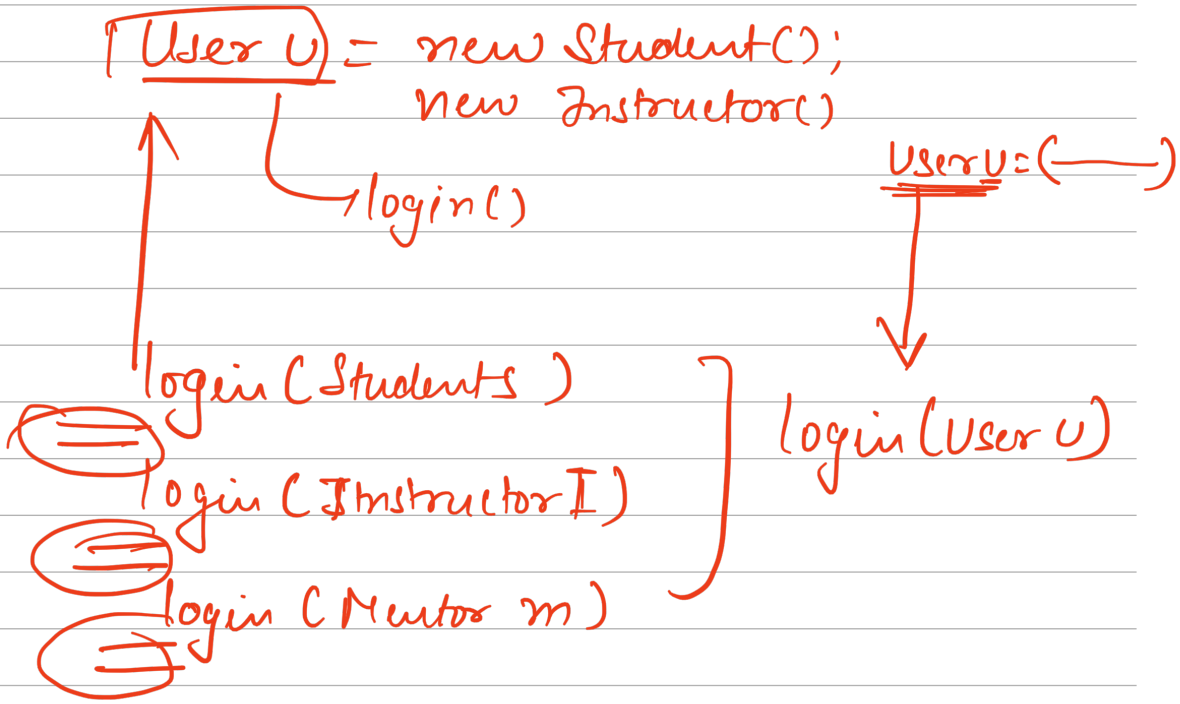
temp = int@10

temp = int@20

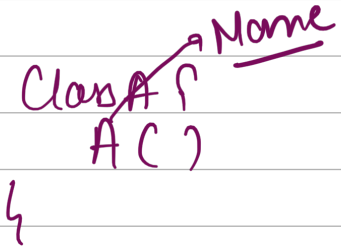


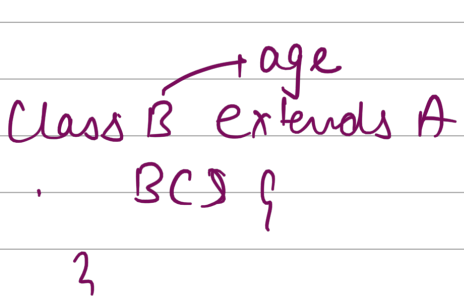
Student st = new student;

Q4



List<parent>k=new ;



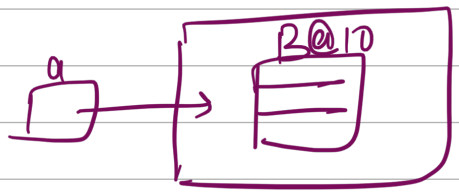


① Variable

A a = new BC);

Constcall BC) → AC)

Object creation A → B



② a.age="20"; X

Bb = new BC);

Aa

Aa = new BC)

login(⟶

B b = new A();

b.age Runtime

A
name

User u = new Student();

login(User u)
{
}

solve problems

List<User>
List<Student>

Objects

A()

B() extends —
super()

B b = new A();

Parent
List<> al = new ArrayList<>

ArrayList<> al = new List<>();

User u = new Student();

runtime
Student

Student s = new User(); ✗

User u = new Student()

Student s = (Student) u; Mentorm
= (mentor) u;

Phonepe

Bank

Bank2
Bank1 → Shutdown

Bank b = new Bank1(); // newBank2();

JAM

OAuth

User

login (

TA
Mentor
Student
Admin
HE
Engg
DE
DS

ArrayList<String> al = new ArrayList<>();

LinkedList

doSorting(List L)

doSortingExtra(List L1)

doSortExtraExtra(List l2)

doSortingExtraExtra2ExtraExtra(ArrayList al)