Class 2.9 - OOPS - 2

Saturday, June 21, 2025 10:40 AM

class Person {
String name;
int age;
static String company = "Intel";
}

Inheritance

→ Reuse the code

class Animal {
int legs;
int eyes;
void walk();
}

class Camel {
int legs;
int eyes;
int humps;
void walk();
}

child → class Camel extends Animal ← parent {
int humps;
}

Child extends Parent → (10)

(A)

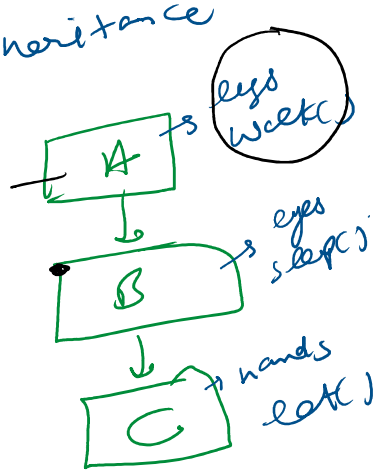
(A) Package : collection of similar types of classes.

Types of Inheritance

Class B extend A

Types of Inheritance

1) Multi level inheritance



class B extend A

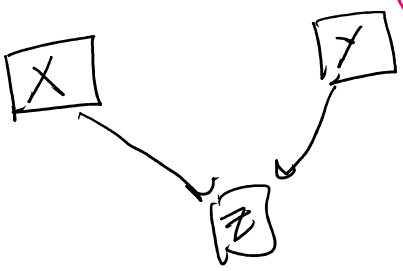
class C extend B

A - legs, walk()

B - eyes, legs, walk(), sleep();

C - all the methods and variables

⇒ Multiple Inheritance ⇒ not allowed in Java



[Z extends X and Y]

```
class X {
  walk() {
    sout("X can walk");
  }
}
```

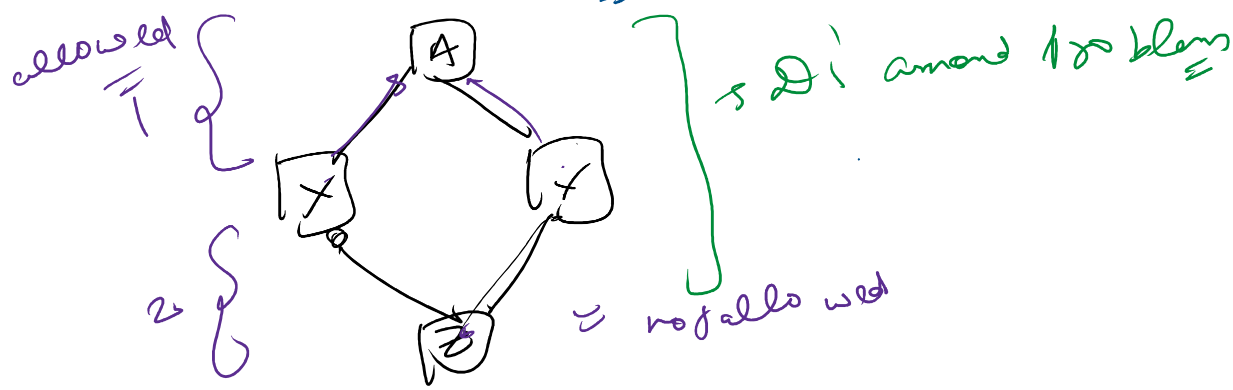
```
class Y {
  walk() {
    sout("Y can walk");
  }
}
```

class Z extend X and Y {

3

2. z = new Z();

Z z = new Z();
z.walk();



Parent p = new Child(); Allowed
Child c = new Parent(); Not allowed

Ref   Obj

How constructor works in Inheritance?

→ when we make an object of a child class, first parent constructor is called, then the child constructor is called.

Super keyword

used to refer to the parent object.

→ 1. call the constructor of parent.

para

super() → Call the constructor of parent

super → member associates and variables

Encapsulation

Binding of the variables and methods operations the variables in a single class

to achieve the desired level of access.

⇒ Access Modifiers

(keywords used to control the access visibility of member variables/methods.

- private
- default
- protected
- public

→ private is accessible only in the same class.

→ ... more

→ getters and setters give you more granular level of access on the variables.

→ default : same package

→ protected : same package + subclass / child class in the diff package.

→ public : anywhere

Polymorphism → applicable on methods

many forms

→ method overloading / compile time poly.

→ method overriding / run time poly.

⇒ Method overloading

↳ same func name but diff signature.

not allowed { double fun(int a, int b)
int fun(int a, int b)

not allowed

int fun(int a, ...

=) Method overriding

→ Runtime

→ It's the object that decides which method has to be called

→ reference decides the variables.

Abstraction

↳ Hiding the implementation

only showing the necessary details.

→ ways to achieve abstraction

1) abstract class → partial abstraction

2) Interface → complete

Interface

↳ only contain method signatures

but doesn't have impl.

→ child class gives the impl.

→ object / instance

(P) Interface and abstract classes cannot be instantiated.

(B) class extends class

class implements interface

interface extends interface

(I) Interfaces allow multiple inheritance