Mahindra
École Centrale
COLLEGE OF ENGINEERING

Object Oriented Programming JAVA

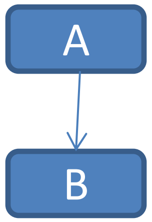
Dr. Prafulla Kalapatapu
Computer Science Engineering
Mahindra Ecole Centrale
prafulla.kalapatapu@mechyd.ac.in

Mahindra École Centrale
COLLEGE OF ENGINEERING

Inheritance

Inheritance

- Define an Inheritance
  Deriving a new class from an existing class.
  new class is child/sub class/derived class
  existing class is parent/super class/base class



- What is the advantage of an inheritance.
  Reusability.

- How can we implement inheritance in java
  Using extends keyword

- Syntax :
  class Base {-----}
  class Derive extends Base{-----}

Program on Inheritance

Mahindra École Centrale
COLLEGE OF ENGINEERING

```
class Bank{
    int bid;
    String bankname;
    String banklocation;
    void pfLoan()  {
            System.out.println("pf loan at 12%"); } }
class AndhraBank extends Bank{
    int brid;
    String brmanager;
    void vLoan() {
            System.out.println("vehicle loan at 15%"); }
    public static void main(String...   a) {
            AndhraBank ab=new AndhraBank();
            ab.vLoan();
            ab.pfLoan();
    } }
```

When you create a subclass object, what will happen in the background

Case (i) : No constructor for super class and no constructor for subclass.

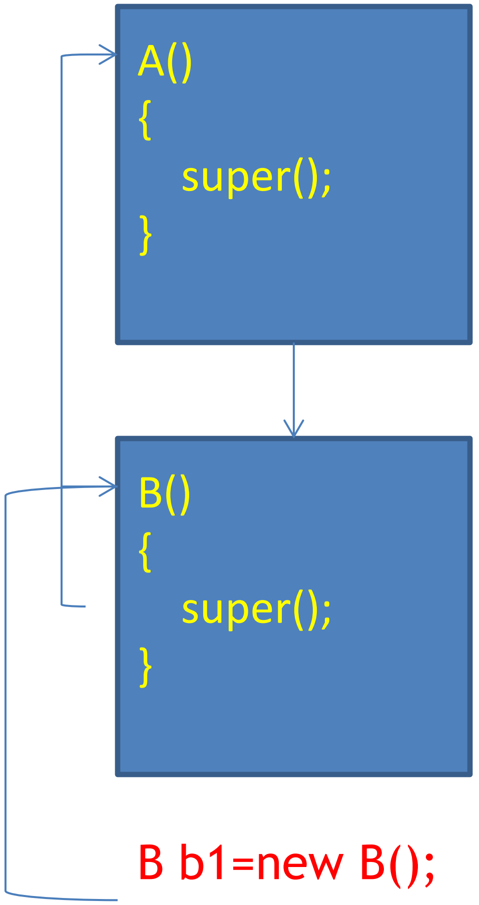
- At the compilation time, it will write default constructor in sub, super class.
- Which constructor will call first

  Sub class constructor.
- Which constructor executes first

  Super class constructor.
- What is super()

  It calls super class constructor
- In A class constructor, purpose of super()

  If my class is not extended from another class explicitly, by default your class is inherited from "Object" class.

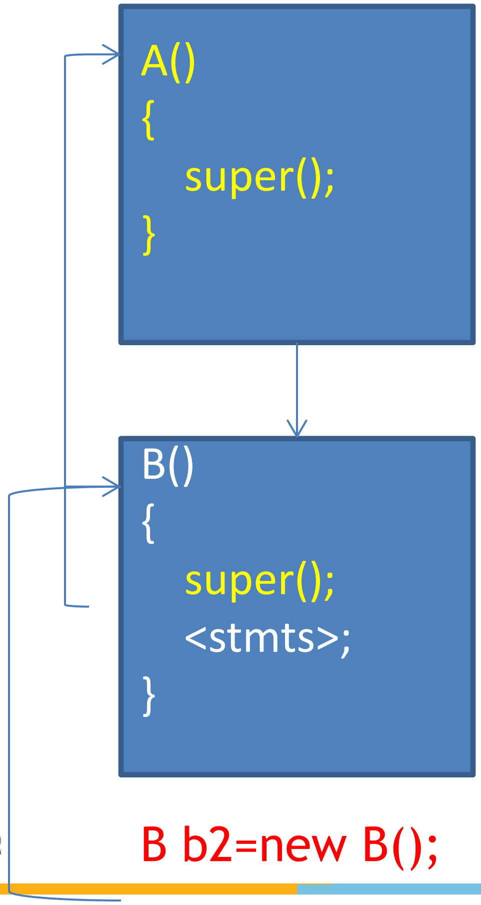


```
A()
{
   super();
}

B()
{
   super();
}

B b1=new B();
```

LEADER ■ ENTREPRENEUR ■ INNOVATOR

Case (ii) : No constructor for super class and constructor for sub class

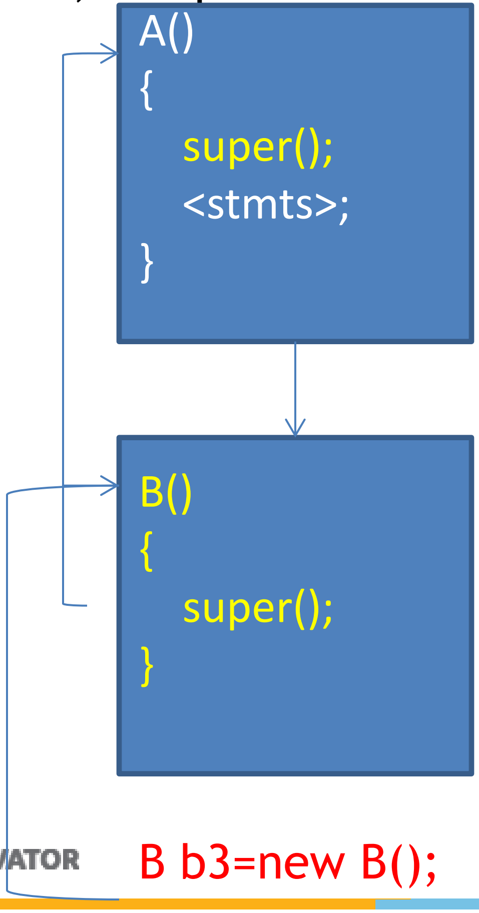
- At the compilation time, it will write default constructor for super class. In subclass constructor, compiler will add super() as a first statement.



B b2=new B();

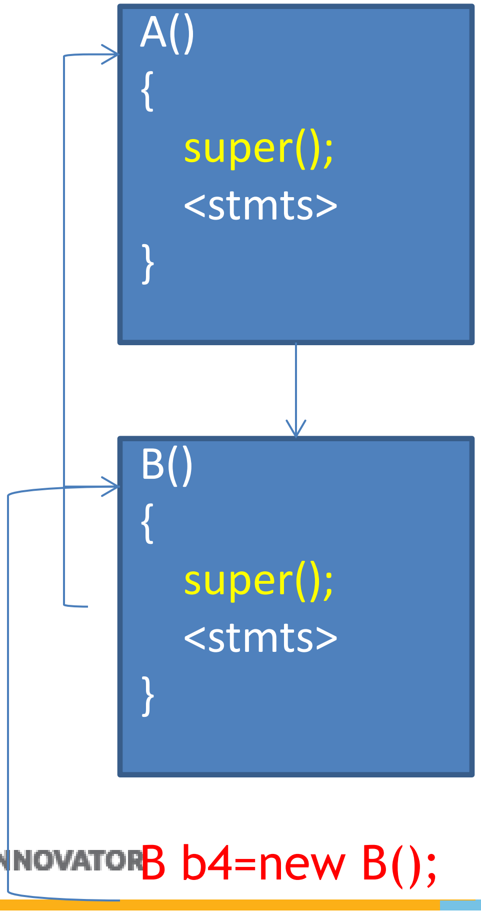
Case (iii) : constructor for super class and no constructor in sub class.

- At the compilation time, compiler will write default constructor in subclass. In super class constructor, compiler will add super() as a first statement



Case (iv) : constructor in super and sub class.

- At the compilation time, compiler will add super() as first statement in super class and as well as in sub class constructor.



```
A()
{
    super();
    <stmts>
}

B()
{
    super();
    <stmts>
}
```

B b4=new B();

Constructor overloading

Defining more than one constructor in the same class with following 3 rules:

Either of the rules has to follow

1. No of arguments must be different
2. Type of arguments must be different
3. Order of arguments must be different

a.

```
class A
{
A() {....}
A (int a) { .... }
}
```

Rule 1
Constructor overloading

b.

```
class A
{
A(int a) {....}
A(float b) {....}
}
```

Rule 2
Constructor overloading

c.

```
class A
{
A(int a, float b) {....}
A(float x, int y) {....}
}
```

Rule 3
Constructor overloading

Example Program

```
Class Sample
{
Sample(int a) {
    this(10,20);
    System.out.println("arg1");}
Sample(int a, int b){
    this(10,20,30);
    System.out.println("arg2");}
Sample(int a,int b,int c){
    System.out.println("arg3");}
public static void main(String []args){
    Sample s1=new Sample(10);
}
}
```

this() is a method, it will call same/local class constructor with provided arguments

As a 3rd step in object creation process, One argument constructor will call.

```
O/P: arg3
     arg2
     arg1
```

Single line definition for super() and this() :

- super() : It calls super class constructor.
- this() : It calls same class constructor.

Note : both should be as a first statement.

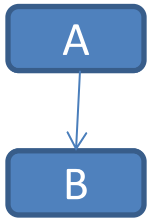
Types of inheritance

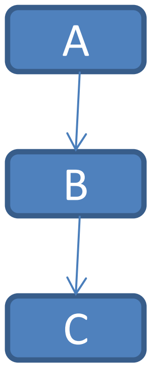
There are two types of inheritance

1. Single Inheritance
2. Multiple Inheritance

1. Single Inheritance :

if any class is derived from only one class(directly), then it is single inheritance.

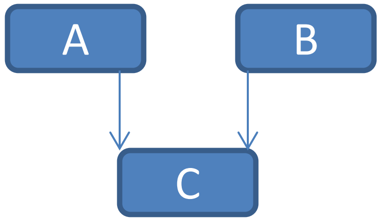




You can call this as multilevel also

2. Multiple Inheritance :

if any class is derived from more than one class (directly), then it is multiple inheritance.



$$\begin{tikzcd} A \arrow[dr] & & B \arrow[dl] \\ & C & \end{tikzcd}$$

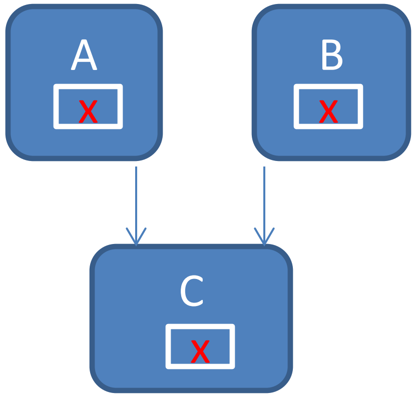
- Which inheritance is possible in Java ?

  only single Inheritance.

- Multiple inheritance is not available in java

- Why multiple inheritance not available in java.

  Multiple inheritance leads to confusion for the programmer.
- For example, class A has got member x, class B has got member x, when an other class C extends both the classes then there is a confusion regarding which copy of x is available in c.



class C extends A, B

- How can we achieve multiple inheritance in java

  using interfaces

Mahindra École Centrale
COLLEGE OF ENGINEERING

Polymorphism

polymorphism

- Define polymorphism

  One name acts as multiple forms based on user interaction.

Polymorphism

Compile time polymorphism
Ex: Method overloading
Method hiding

Runtime polymorphism
Ex: overriding

Method overloading

Mahindra École Centrale
COLLEGE OF ENGINEERING

- What is an overloading

  Defining more than one method with the same name in the same class with the following condition : either of the condition has to satisfy.

  1. No of arguments must be different
  2. type of arguments must be different
  3. order of arguments must be different

- Can we overload a function in C ?

  No

- What is the advantage of polymorphism

  flexibility

Program on compile time polymorphism or overloading

```
class Test {
void m1(int i) {
    System.out.println(“int arg”); }
void m1(long l) {
    System.out.println(“long arg”); }
void m1(int k, char c) {
    System.out.println(“int,char arg”); }
public static void main(String...   a)
{
    Test t=new Test();
    t.m1(10);
    t.m1(10l);
    t.m1(10,’c’);
}
}
```

- In overloading, method resolution is always taken care by compiler based on reference type and arguments.

- Method resolution : Association/binding/linking between method call and method definition.

- Overloading is also considered as compile time polymorphism, static polymorphism, early binding.

Method overriding

- What is a method overriding

  Redefining super class method in the sub class with the following rules :
  All rules has to satisfy.

  1. Return type should be same
  2. Method signature should be same
  3. subclass method should not have weaker privilege than super class method (greater or equal is ok)

Program on method overriding

```
class A {
void m1() {
        System.out.println(“super m1()”); }
}
class  B extends A {
public void m1() {
        System.out.println(“sub m1()”); }
}
class Sample {
public static void main(String …a) {
        B b1=new B();
        b1.m1();
}
}
```

Overriding method

Overridden method

O/P : sub m1()

- Method resolution always taken care by JVM based on runtime object.
- Overriding is also considered as runtime polymorphism, dynamic polymorphism, late binding.
- What is the pre-requisite for method overriding.

  inheritance