

# **FAST**

# National University of Computer and Emerging Sciences Peshawar

Lecture # 12

## Software Construction and Development (Java Programming)

**Instructor:** Muhammad Abdullah Orakzai

DEPARTMENT OF COMPUTER SCIENCE



الذى علم بالقلم. علم الانسان ما لم يعلم.



# Inheritance in Java

# Contents

- 1) Inheritance in java
- 2) Types of inheritance
- 3) Method Overriding

# Inheritance in java

- ❖ The process by which one class acquires the properties(data members) and behaviours/functionalities(methods) of another class is called **inheritance**.
- ❖ Including features of one class into another class is called inheritance.
- ❖ The idea behind inheritance in java is that you can create new classes that are built upon existing classes, you can reuse method and fields of parent class, and you can add new methods and fields also.

# Inheritance in java...

- ❖ The parent class is unchanged by this process.
- ❖ The aim of inheritance is to provide the reusability of code so that a class has to write only the unique features and rest of the common properties and functionalities can be extended from the another class.

# Child Class

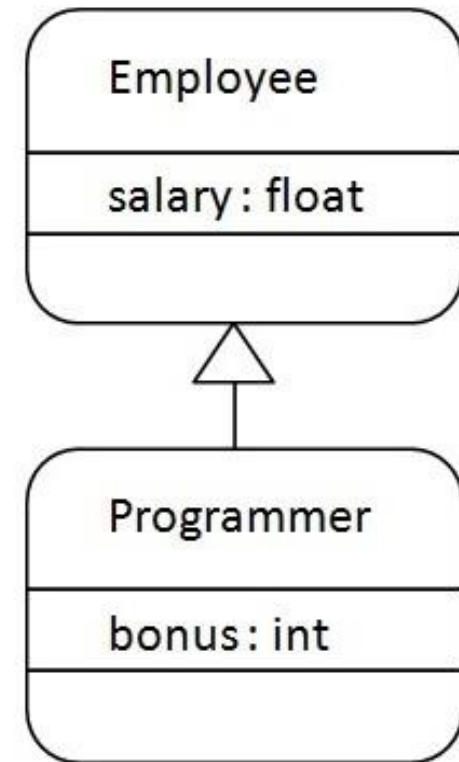
- ❖ The class that extends the features of another class is known as child class, sub class or derived class.
- ❖ The class that does inheritance is called child class.

# Parent Class

- ❖ The class whose properties and functionalities are used (inherited) by another class is known as parent class, super class or Base class.
- ❖ The class that is inherited is called parent class.

# Inheritance in java...

- ❖ In java terminology inherited class is called super class and new class is called sub class.
- ❖ “extends” keyword is used to inherit a class in java.
- ❖ Inheritance represents the “Is a Relationship”
- ❖ Sub class is more powerful than super class because it has its own features as well as of the super class.





# Uses Inheritance in java

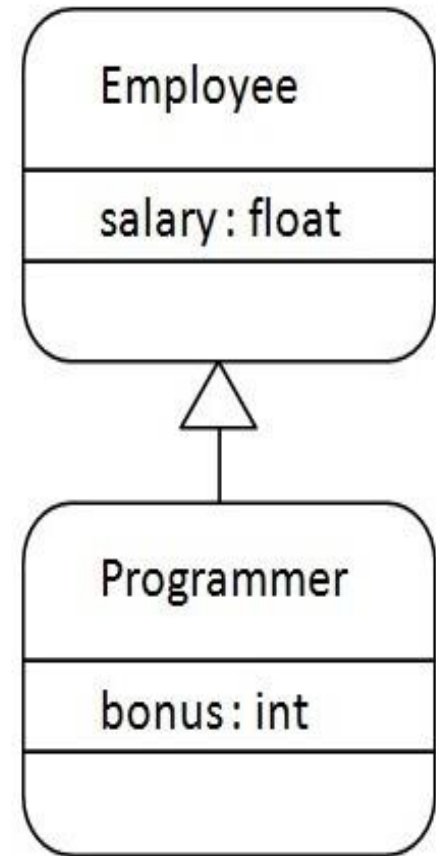
- ❖ For method overriding (runtime polymorphism can be achieved).
- ❖ For code reusability.

# Inheritance Example

```
// This program explains single level inheritance
package inheritance;

class Employee
{
    float salary=40000;
}

class Programmer extends Employee
{
    int bonus=10000;
}
```



# Inheritance Example...

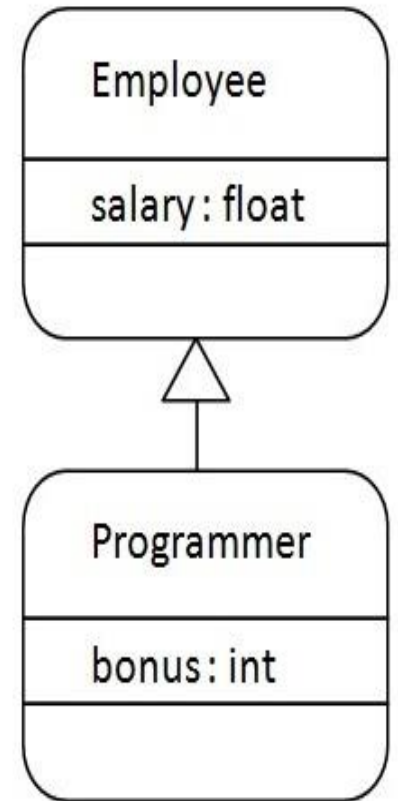
```
public class InheritanceDemo    // main class
{
    public static void main(String args[])
    {
        Programmar p=new Programmar();
        System.out.println("Programmer salary is:"+p.salary);
        System.out.println("Bonus of Programmer is:"+p.bonus);
    }
}
```

## **Output:**

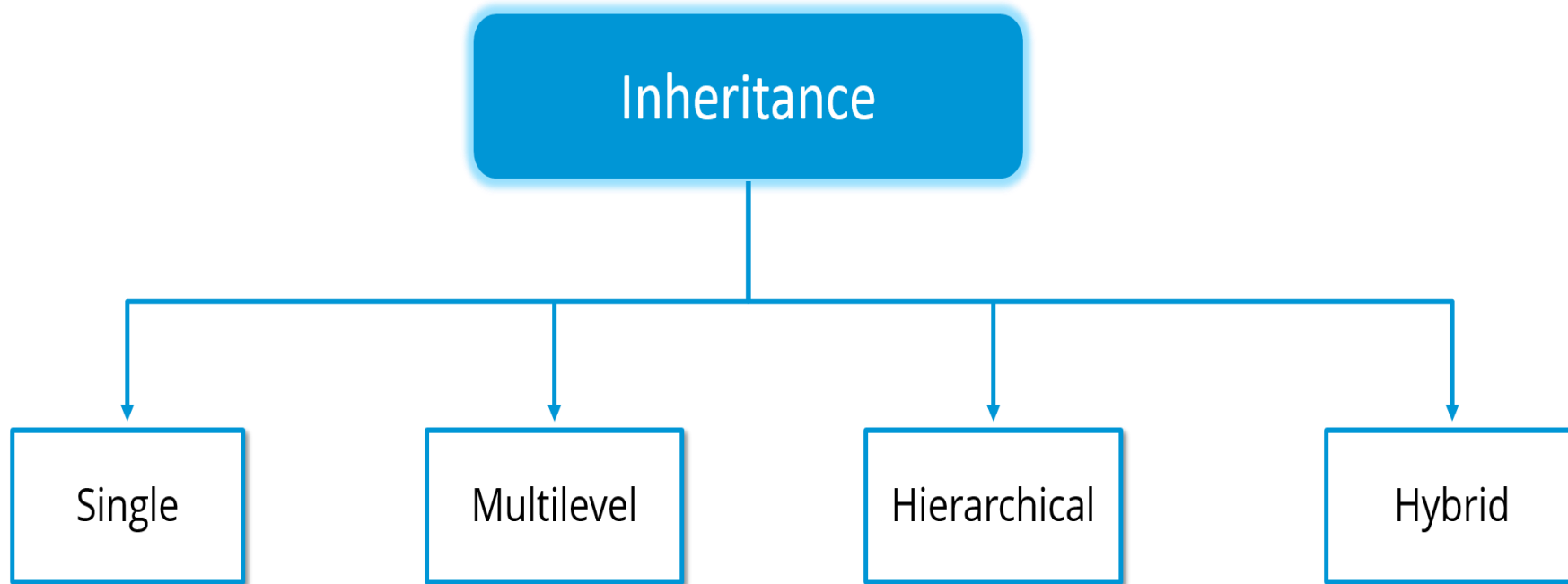
Programmer salary is:40000.0  
Bonus of programmer is:10000

# Inheritance Example...

- ❖ This figure shows that Programmer is the subclass and Employee is the super class.
- ❖ Relationship between two classes is “Programmer IS A Employee.”
- ❖ It means that programmer is a type of Employee.

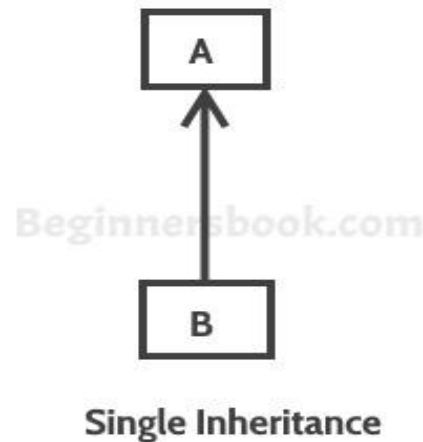


# Types of inheritance in java



# 1) Single Inheritance

- ❖ Refers to a child and parent class relationship where a class extends the another class.
- ❖ In which one new class (child class) is derived from only one super class.



# 1) Single Inheritance Example

```
class C {           // Super class
    int x,y;
    void getData(int a, int b){
        x=a;
        y=b;
    }
    int add() {
        System.out.println("Super Class Method");
        return(x+y);
    }
} // class C end
```

# 1) Single Inheritance Example...

```
class D extends C           // sub Class
{
    int multip()
    {
        System.out.println("Sub Class Method");
        return(x*y);
    }
}
```

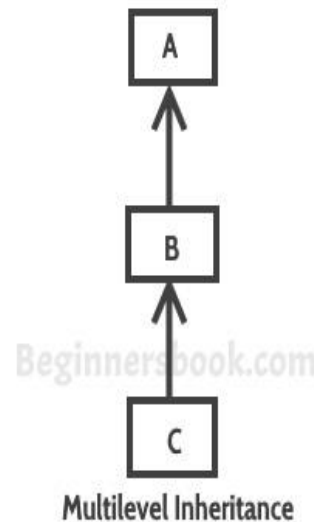


# 1) Single Inheritance Example...

```
public class SingleInheritance    // main class
{
    public static void main(String[] args)
    {
        D obj = new D();
        obj.getData(5,7);
        System.out.println("Addition is= " + obj.add() );
        System.out.println("");
        System.out.println("Multiplication is= " + obj.multip() );
    }
}
```

## 2) Multilevel inheritance

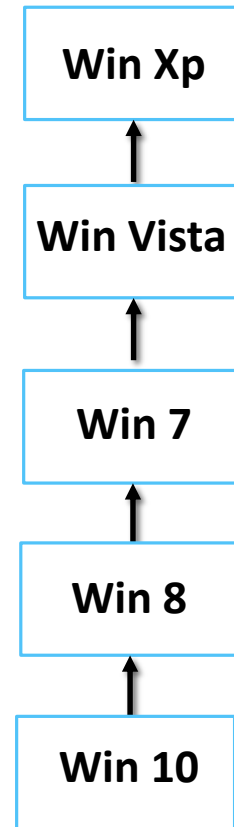
- ❖ Refers to a child and parent class relationship where a class extends the child class.
- ❖ For example class C extends class B and class B extends A.



## 2) Multilevel inheritance Example

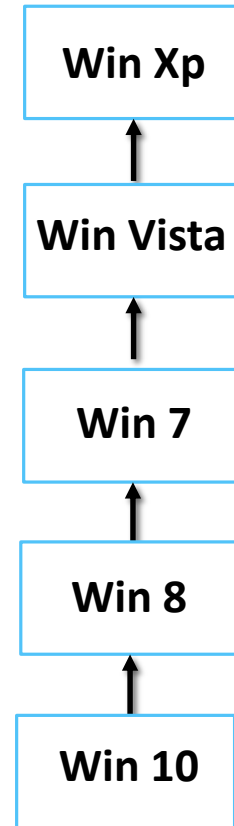
// This program explains Multilevel Inheritance in java  
package inheritance;

```
class WinXp
{
    public void met1()
    {
        System.out.println("Windows Xp");
    }
}
```



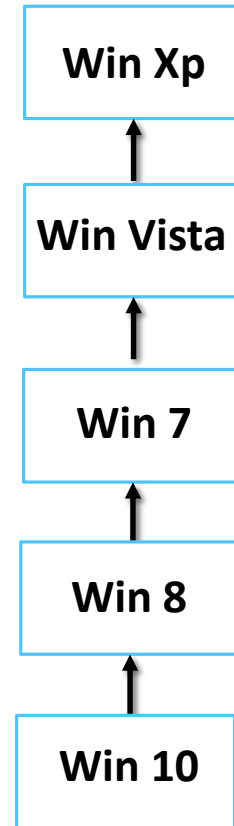
## 2) Multilevel inheritance Example...

```
class WinVista extends WinXp{  
    public void met2()  
    {  
        System.out.println("Windows Vista");  
    }  
}
```



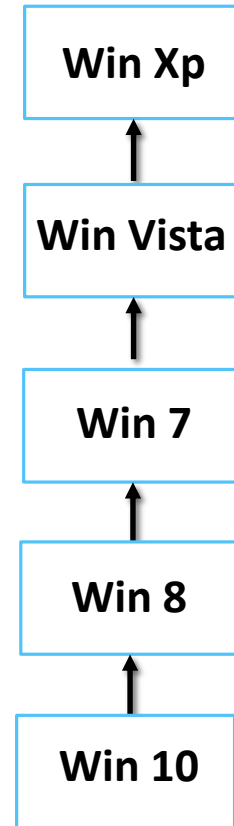
## 2) Multilevel inheritance Example...

```
class Win7 extends WinVista {  
    public void met3()  
    {  
        System.out.println("Windows 7");  
    }  
}
```



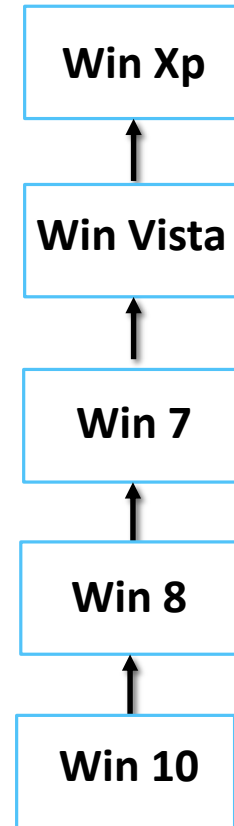
## 2) Multilevel inheritance Example...

```
class Win8 extends Win7{  
    public void met4()  
    {  
        System.out.println("Windows 8");  
    }  
}
```



## 2) Multilevel inheritance Example...

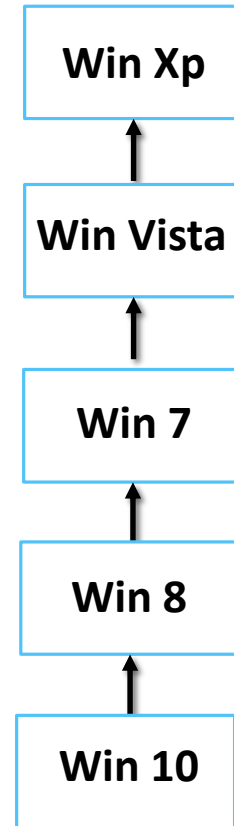
```
class Win10 extends Win8 {  
    public void met5()  
    {  
        System.out.println("Windows 10");  
    }  
}
```



## 2) Multilevel inheritance Example...

```
public class InheritanceDemo2    // Main class
{
    public static void main(String[] args) {
        Win10 object = new Win10();
        object.met5();
        object.met4();
        object.met3();
        object.met2();
        object.met1();
    }
}
```

```
<terminated> Inheritance
Windows 10
Windows 8
Windows 7
Windows Vista
Windows Xp
```





## 2) Multilevel inheritance Example 2

// This program shows the concept of Multilevel Inheritance

```
package inheritance;
```

```
class Student {           // super class
```

```
public int rollNo;
```

```
void getRollNo(int n)
```

```
{
```

```
rollNo=n;
```

```
}
```

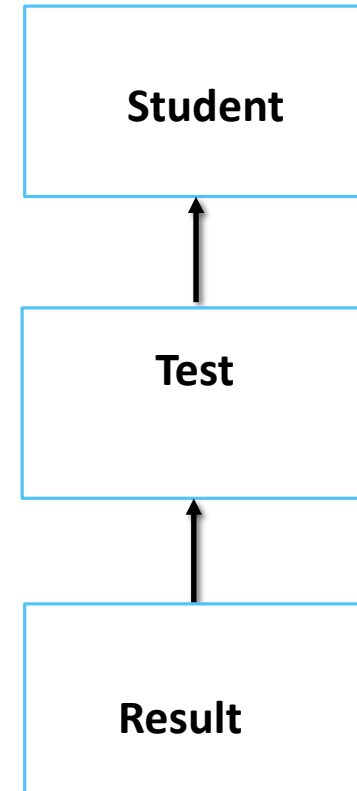
```
public void printRollNo()
```

```
{
```

```
System.out.println("Roll No is: " + rollNo);
```

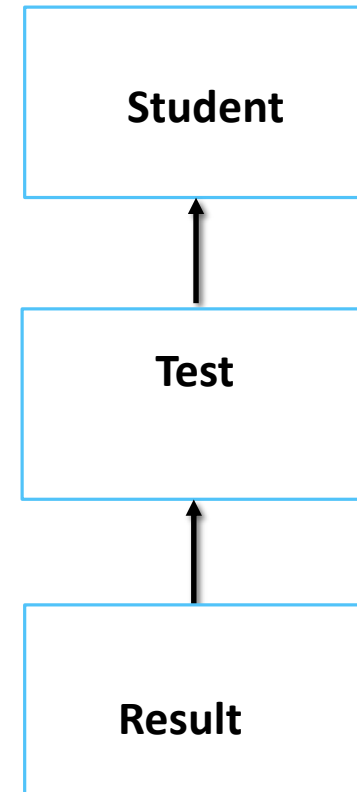
```
}
```

```
}
```



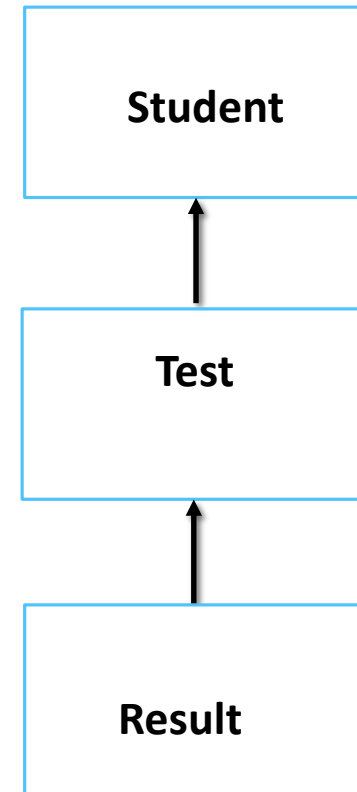
## 2) Multilevel inheritance Example 2...

```
class Test extends Student {    // sub class
int m1, m2;
void getMarks(int x , int y)
{
m1 = x;
m2 = y;
}
void printMarks()
{
System.out.println("Test 1 Marks are = " +m1);
System.out.println("Test 2 Marks are = " +m2);
}
}
```



## 2) Multilevel inheritance Example 2...

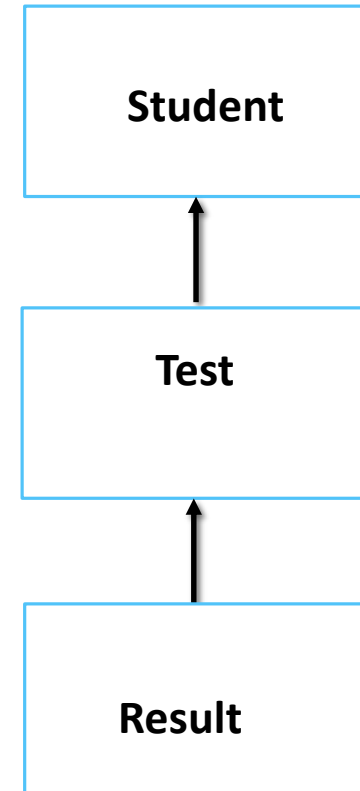
```
class Result extends Test           // Sub class
{
void printResult()
{
    printRollNo();
    printMarks();
    System.out.println("Result is= " + (m1 + m2));
}
}
```



## 2) Multilevel inheritance Example 2...

```
public class MultilevelInheritance           // main class
{
    public static void main(String[] args)
    {
        Result t = new Result();
        t.getRollNo(150517);
        t.getMarks(80 , 90);
        t.printResult();
    }
}
```

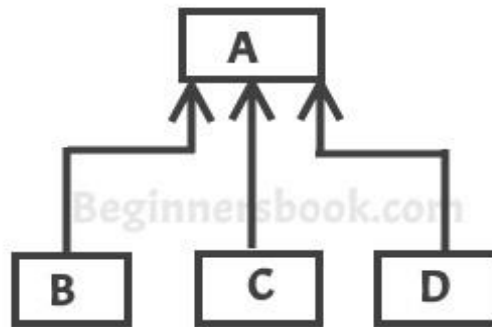
```
Roll No is: 150517
Test 1 Marks are = 80
Test 2 Marks are = 90
Result is= 170
```



### 3) Hierarchical Inheritance

When two or more classes inherit a single class, it is known as *hierarchical inheritance*.

In the example given below, Dog and Cat classes inherit the Animal class, so there is hierarchical inheritance.



Hierarchical Inheritance

### 3) Hierarchical Inheritance...

```
class Animal{  
    void eat(){System.out.println("eating...");}  
}  
  
class Dog extends Animal{  
    void bark(){System.out.println("barking...");}  
}  
  
class Cat extends Animal{  
    void meow(){System.out.println("meowing...");}  
}
```

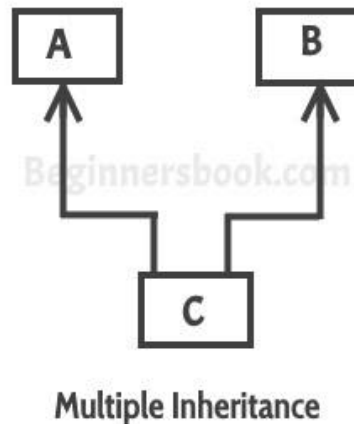
### 3) Hierarchical Inheritance...

```
class TestInheritance3{  
    public static void main(String args[]){  
        Cat c=new Cat();  
        c.meow();  
        c.eat();  
        //c.bark();//C.T.Error  
    }  
}
```

## 4) Multiple Inheritance

Refers to the concept of one class extending more than one classes, which means a child class has two parent classes. For example class C extends both classes A and B.

**Note: Java doesn't support multiple inheritance.**





# Why multiple inheritance is not supported in java?

- ❖ To reduce the complexity and simplify the language, multiple inheritance is not supported in java.
- ❖ Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.
- ❖ Since compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error.

# Why multiple inheritance is not supported in java?...

```
class A{  
    void msg(){System.out.println("Hello");}  
}  
  
class B{  
    void msg(){System.out.println("Welcome");}  
}
```

# Why multiple inheritance is not supported in java?...

```
class C extends A,B{           //suppose if it were
```

```
public static void main(String args[]){
```

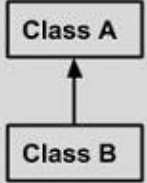
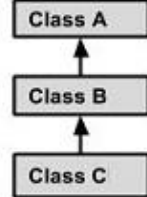
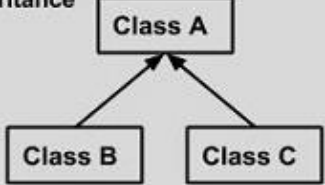
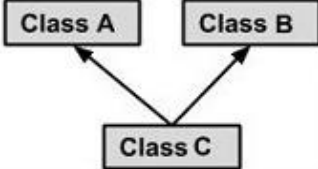
```
    C obj=new C();
```

```
    obj.msg();//Now which msg() method would be invoked?
```

```
}
```

```
}
```

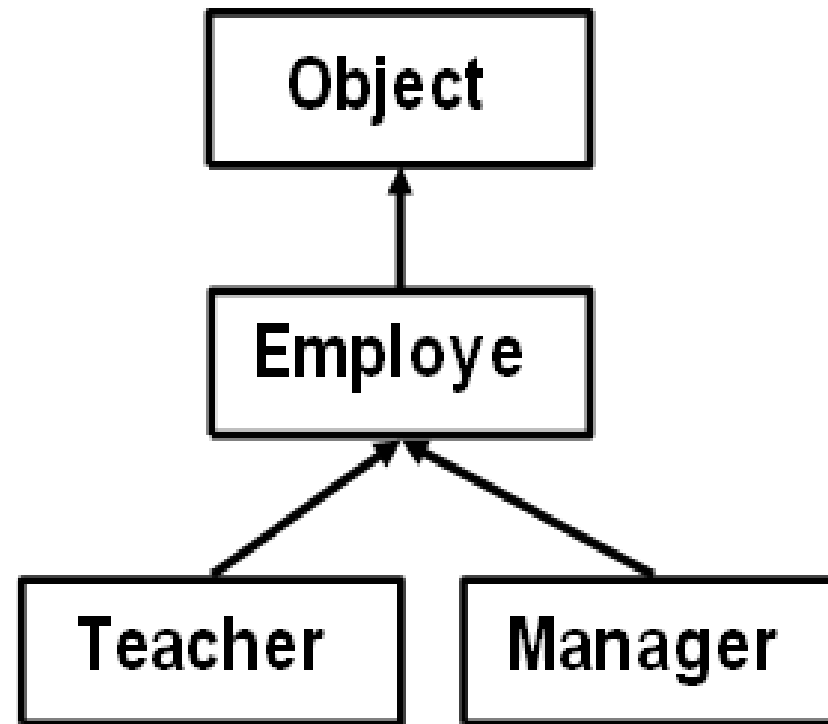
# Types of Inheritance

<b>Single Inheritance</b>  <pre>graph BT; B[Class B] --&gt; A[Class A];</pre>	<pre>public class A {     ..... } public class B extends A {     ..... }</pre>
<b>Multi Level Inheritance</b>  <pre>graph BT; C[Class C] --&gt; B[Class B]; B --&gt; A[Class A];</pre>	<pre>public class A { .....} public class B extends A {.....} public class C extends B {.....}</pre>
<b>Hierarchical Inheritance</b>  <pre>graph BT; B[Class B] --&gt; A[Class A]; C[Class C] --&gt; A;</pre>	<pre>public class A { .....} public class B extends A {.....} public class C extends A {.....}</pre>
<b>Multiple Inheritance</b>  <pre>graph BT; A[Class A] --&gt; C[Class C]; B[Class B] --&gt; C;</pre>	<pre>public class A { .....} public class B {.....} public class C extends A,B {     ..... } // Java does not support multiple Inheritance</pre>

# Object - The Root Class

- ❖ The Object class in Java is a superclass for all other classes defined in Java's class libraries, as well as for user-defined Java classes. For user defined classes, its not necessary to mention the Object class as a super class, java does it automatically for you.
- ❖ The class Hierarchy of Employee class is shown below. Object is the super class of Employee class and Teacher is a subclass of Employee class. We can make another class Manager that can also extends from Employee class.

# Object - The Root Class...



# Method Overriding

- ❖ If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in java**.
- ❖ In other words, If subclass provides the specific implementation of the method that has been provided by one of its parent class, it is known as method overriding.
- ❖ When we have same name methods in super and subclass then such methods are called overridden methods.

# Usage of Java Method Overriding

- ❖ Methods having same signature in different classes i.e. super class and sub class.
- ❖ Method overriding is used to provide specific implementation of a method that is already provided by its super class.
- ❖ Method overriding is used for runtime polymorphism



# Rules for Java Method Overriding

1. Method must have same name as in the parent class
2. Method must have same parameter as in the parent class.
3. Must be IS-A relationship (inheritance).

# Methods that are not overridden

1. Private methods
2. Constructors
3. Static methods
4. Final methods

# Method Overriding Example 1

```
package inheritance;  
class A  
{  
    public void met()          // overridden method  
    {  
        System.out.println("Met Of A");  
    }  
}
```

# Method Overriding Example 1...

```
class B extends A
{
    public void met()          // overriding method
    {
        super.met();          // super keyword is used to call super class overridden method

        System.out.println("Met Of B");
    }
}
```

# Method Overriding Example 1...

```
public class MethodOverriding
{
    public static void main(String[] args)
    {
        B obj=new B();
        obj.met();
    }
}
```

# Method Overriding Example 2

```
class Human{
    //Overridden method
    public void eat()
    {
        System.out.println("Human is eating");
    }
}
class Boy extends Human{
    //Overriding method
    public void eat(){
        System.out.println("Boy is eating");
    }
    public static void main( String args[]) {
        Boy obj = new Boy();
        //This will call the child class version of eat()
        obj.eat();
    }
}
```

Output:

Boy is eating

# Method Overriding Example 3

```
class CarClass
{
    public int speedLimit()
    {
        return 100;
    }
}
class Ford extends CarClass
{
    public int speedLimit()
    {
        return 150;
    }
    public static void main(String args[])
    {
        CarClass obj = new Ford();
        int num= obj.speedLimit();
        System.out.println("Speed Limit is: "+num);
    }
}
```

Output:

Speed Limit is: 150

# THANK YOU

