INHERITANCE

Inheritance in Java Mechanism of deriving new class from old class.

The old class is known as- **Base Class / Super class / Parent Class**

The new class is known as- **Derived class/ Sub Class / Child class**

Types:

- Single Inheritance
- Multilevel Inheritance
- Multiple inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

INHERITANCE

What are the two different types of customers you can see in the retail application?

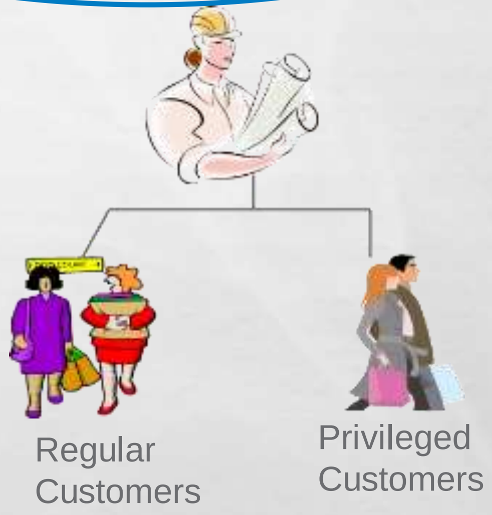
Customers are of two kinds

- Regular
- Privileged

All customers have Customer Id, Name, Telephone Number and Address

The regular customer in addition is given discounts

All customers have some generic features. The different kinds of customers have all generic features in addition to some specific features



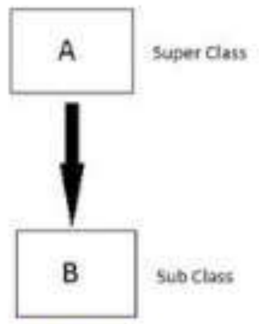
INHERITANCE : Is a mechanism which allows to define generalized characteristics and behavior and also create specialized ones. The specialized ones automatically tend to inherit all the properties of the generic ones

The extends keyword indicates that you are making a new class that derives from an existing class.
A class that is inherited is called a super class.
The new class is called a subclass.

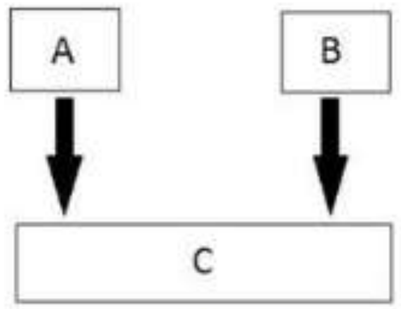
Syntax:

```
class Subclass-name extends Superclass-name
{
  //methods and fields
}
```

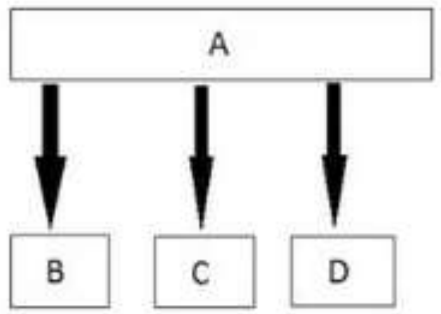
```
class Employee
{
 float salary=40000;
}
class Programmer extends Employee
{
int bonus=10000;
  public static void main(String args[])
{
                  Programmer p=new Programmer();
System.out.println("Programmer salary is:"+p.salary);
                  System.out.println("Bonus of Programmer is:"+p.bonus);

}

}
```



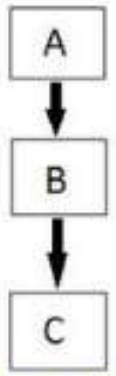
Single



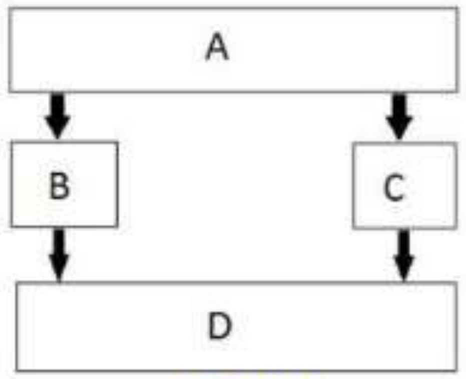
Multiple



Tree/Hierarchical



Multi-level



Hybrid

Multiple And Hybrid Inheritance Is Supported Through Interface Only

Single inheritance -

When a class extends another one class only then we call it a single inheritance.
The below flow diagram shows that class B extends only one class which is A.
Here A is a parent class of B and B would be a child class of A.

```
Class A
{
        public void methodA()
        {
                System.out.println("Base class method");
        }
}
Class B extends A
{
        public void methodB()
        {
                System.out.println("Child class method");
        }
        public static void main(String args[])
        {
                B obj = new B();
                obj.methodA(); //calling super class method
                obj.methodB(); //calling local method
        }
}
```

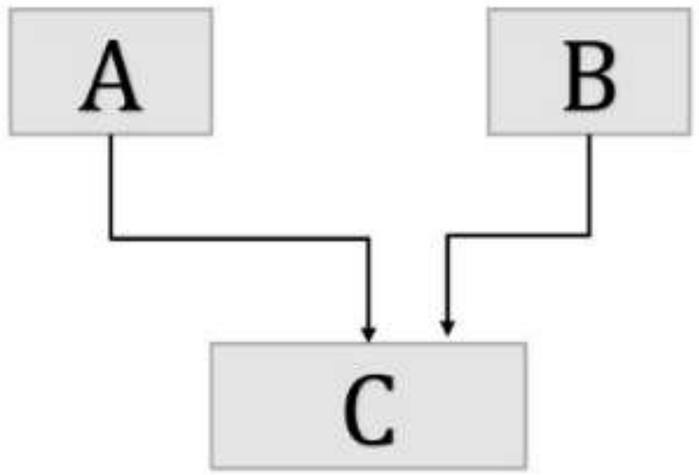
Sub Class Constructor

- A subclass constructor is used to construct the instance variables of both the subclass and the superclass.
- The subclass constructor uses the keyword ***super*** to invoke the constructor method of superclass.
- The **super** Keyword is used with following Conditions
  - **super** may only be used within a subclass constructor.
  - The call to super must appear as first statement.
  - Parameters in the super call must match with declaration in superclass

```
class Vehicle
{
        Vehicle()
        {
        System.out.println("Vehicle is created");
        }
}
class Bike extends Vehicle{
        Bike()
        {
                super();//will invoke parent class constructor
                System.out.println("Bike is created");
        }
public static void main(String args[]){
Bike b=new Bike();
}
}
```

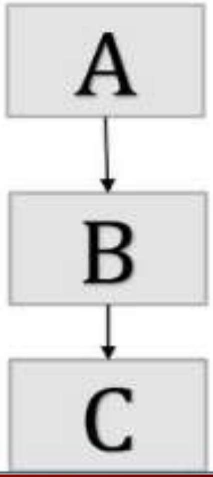
“Multiple Inheritance”

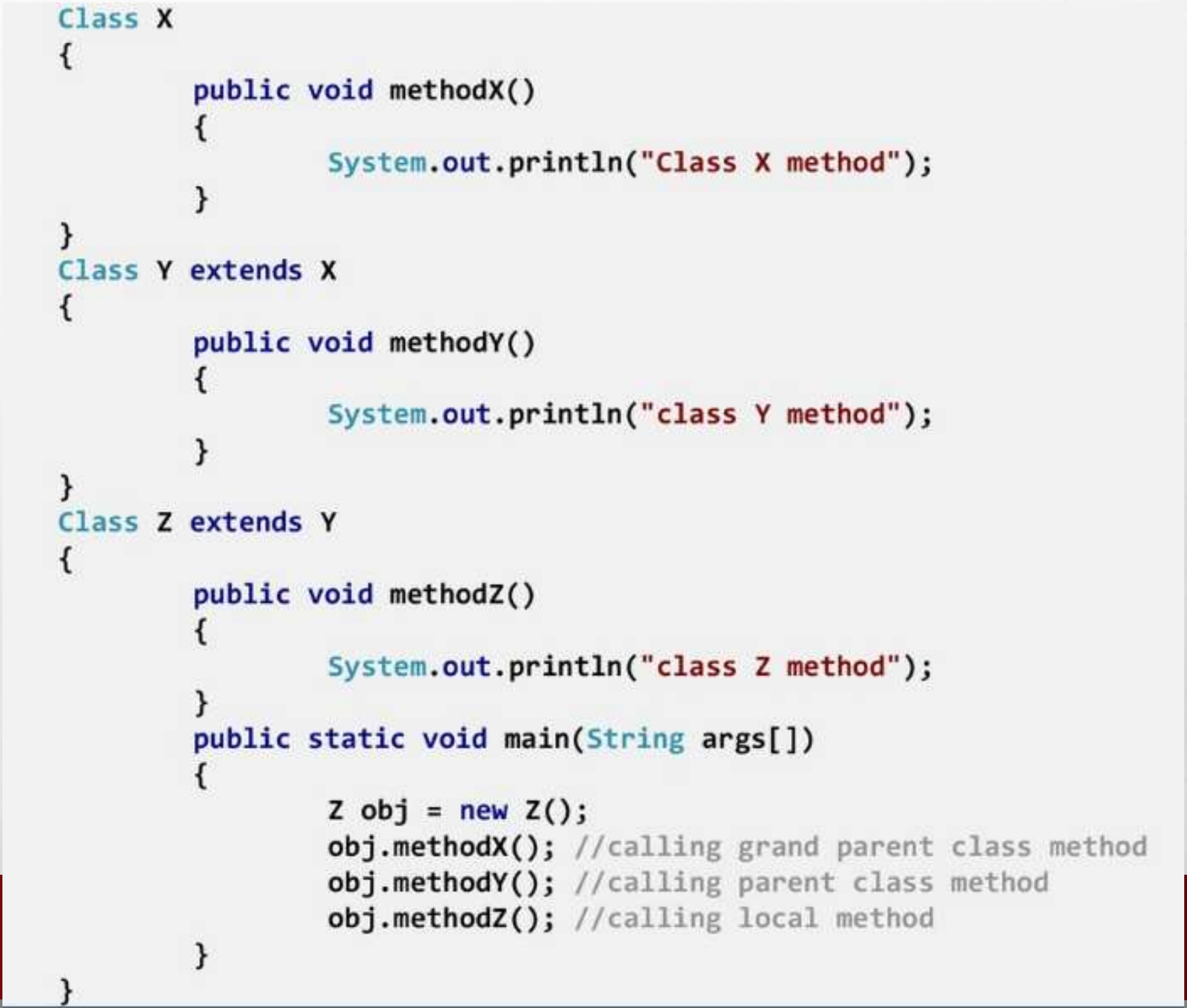
- “**Multiple Inheritance**” refers to the concept of one class extending (Or inherits) **more than one base class**.
- The problem with “multiple inheritance” is that the derived class will have to manage the dependency on two base classes.



Multilevel Inheritance

- **Multilevel inheritance** refers to a mechanism in OO technology where one can inherit from a derived class, thereby making this derived class the base class for the new class.
- As you can see in below flow diagram C is subclass or child class of B and B is a child class of A.

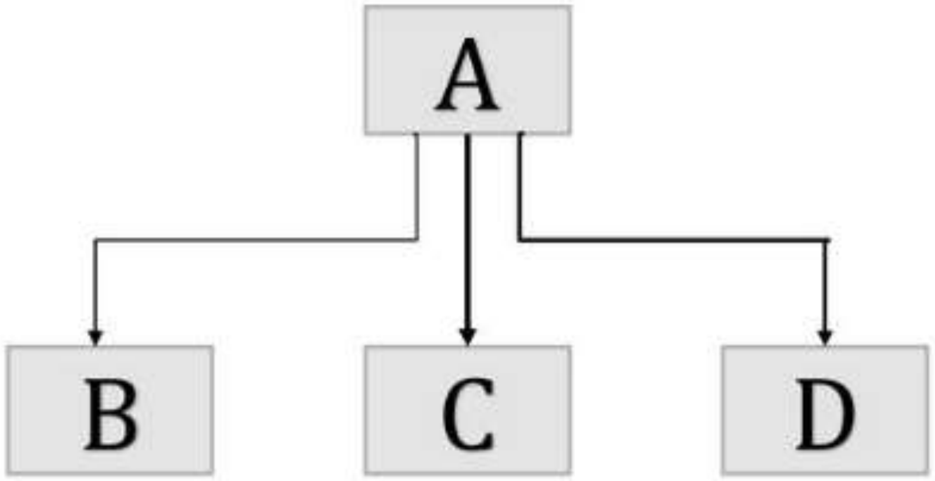

$$\begin{tikzcd}\boxed{A} \arrow[d] \\ \boxed{B} \arrow[d] \\ \boxed{C}\end{tikzcd}$$



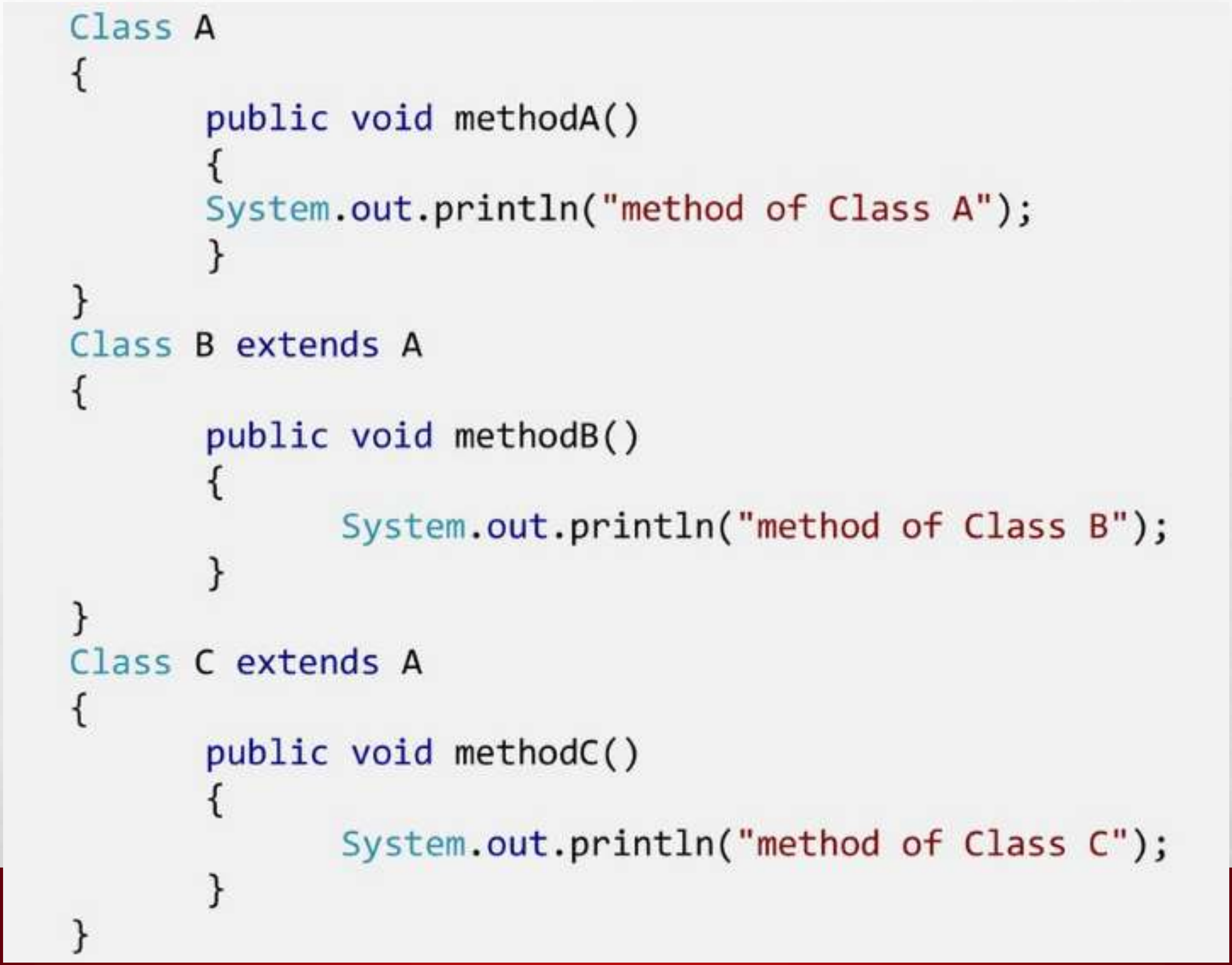
```
Class X
{
        public void methodX()
        {
                System.out.println("Class X method");
        }
}
Class Y extends X
{
        public void methodY()
        {
                System.out.println("class Y method");
        }
}
Class Z extends Y
{
        public void methodZ()
        {
                System.out.println("class Z method");
        }
        public static void main(String args[])
        {
                Z obj = new Z();
                obj.methodX(); //calling grand parent class method
                obj.methodY(); //calling parent class method
                obj.methodZ(); //calling local method
        }
}
```

Hierarchical Inheritance

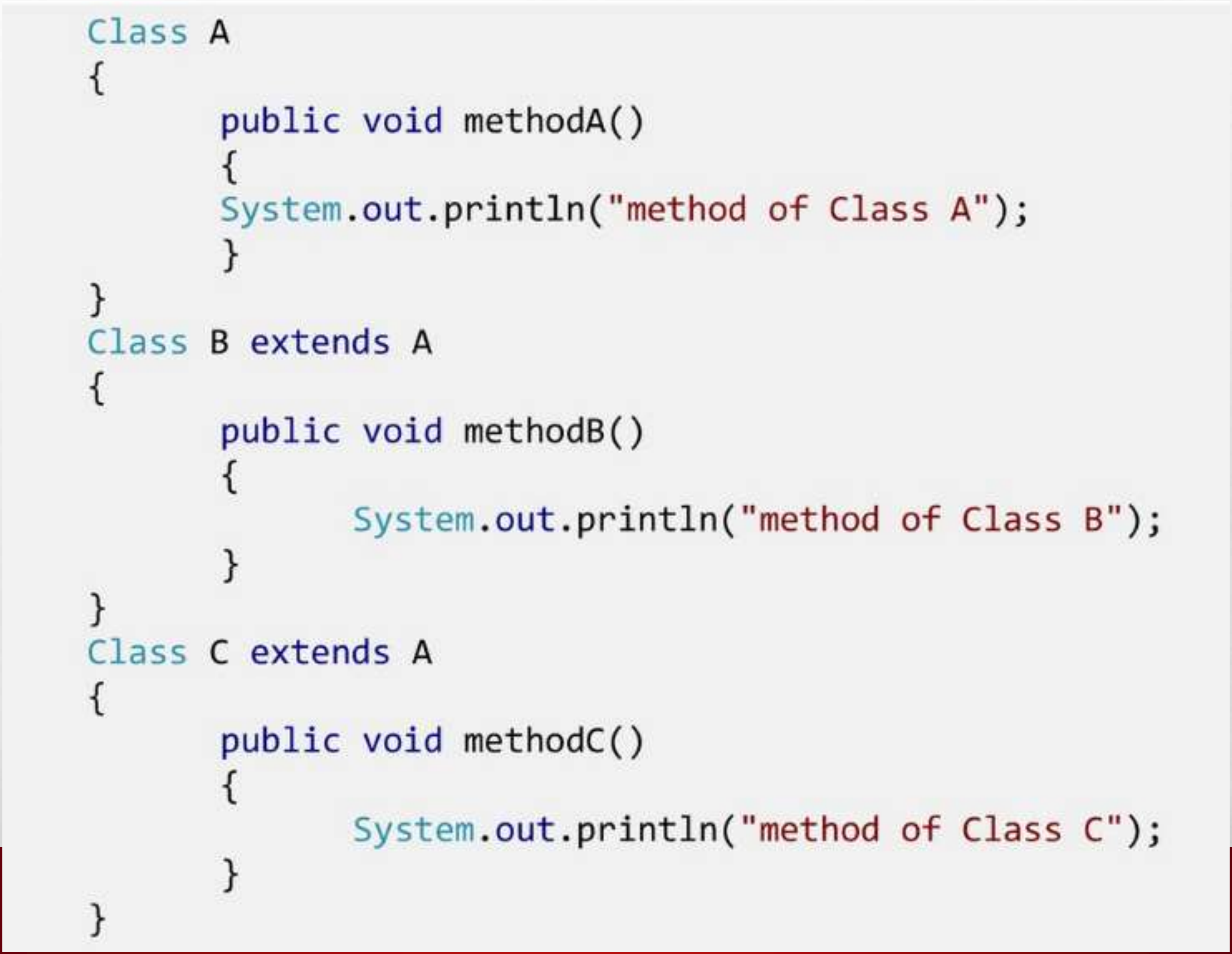
- In such kind of inheritance one class is inherited by many **sub classes**.
- In below example class B,C and D **inherits** the same class A.
- A is **parent class (or base class)** of B,C & D.



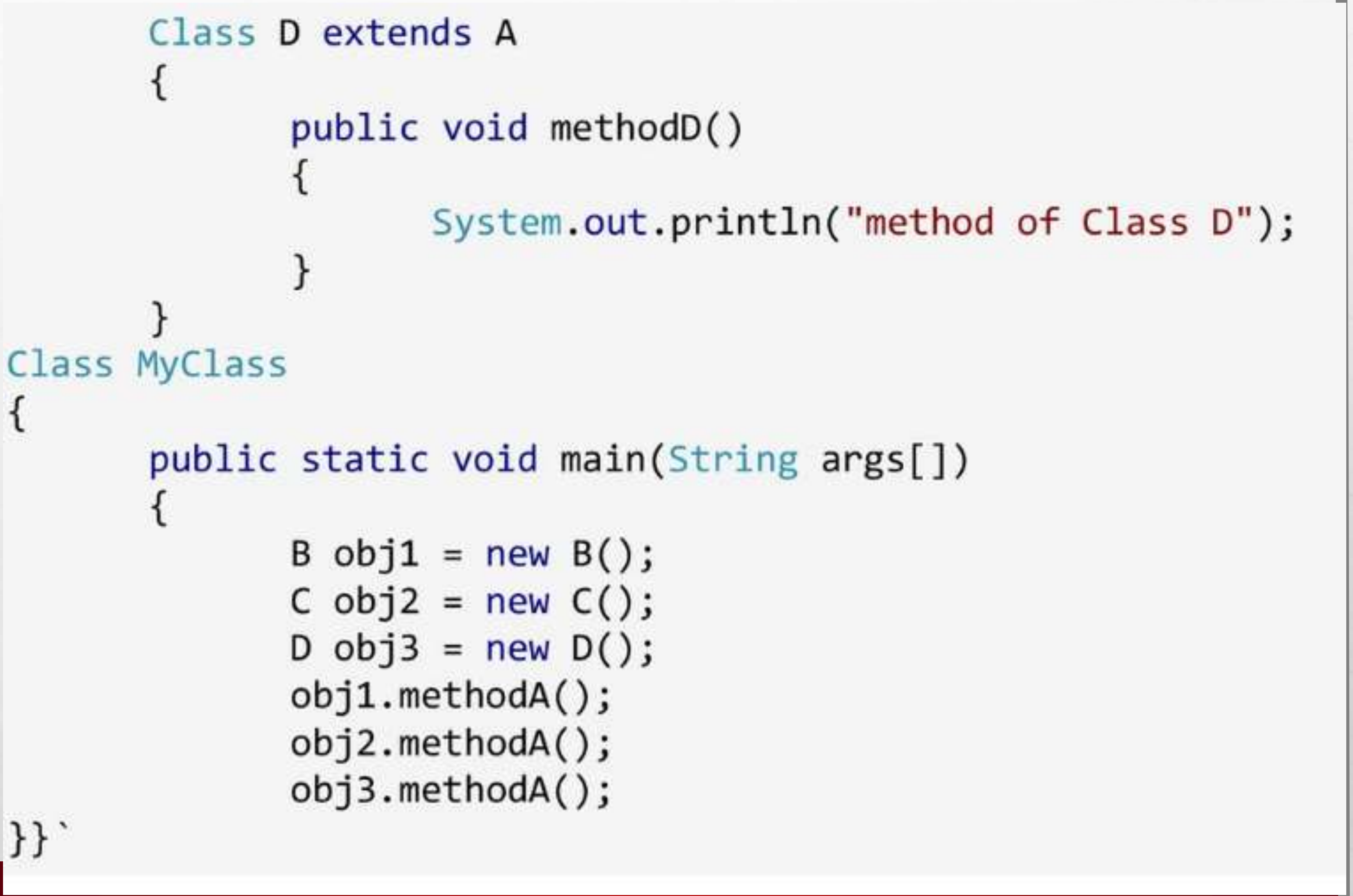
$$\begin{tikzcd}
 & A \arrow[dl] \arrow[d] \arrow[dr] & \\
B & C & D
\end{tikzcd}$$



```
Class A
{
        public void methodA()
        {
        System.out.println("method of Class A");
        }
}
Class B extends A
{
        public void methodB()
        {
                System.out.println("method of Class B");
        }
}
Class C extends A
{
        public void methodC()
        {
                System.out.println("method of Class C");
        }
}
```



```
Class A
{
        public void methodA()
        {
        System.out.println("method of Class A");
        }
}
Class B extends A
{
        public void methodB()
        {
                System.out.println("method of Class B");
        }
}
Class C extends A
{
        public void methodC()
        {
                System.out.println("method of Class C");
        }
}
```



```
      Class D extends A
      {
             public void methodD()
             {
                    System.out.println("method of Class D");
             }
      }
Class MyClass
{
      public static void main(String args[])
      {
             B obj1 = new B();
             C obj2 = new C();
             D obj3 = new D();
             obj1.methodA();
             obj2.methodA();
             obj3.methodA();
}}`
```