



**BITS Pilani**  
Pilani Campus

# Object Oriented Programming

## CS F213

### Amit Dua

Slides Taken from the slides prepared by Dr. Jennifer



# Inheritance

**BITS Pilani**  
Pilani Campus

# What is Inheritance?



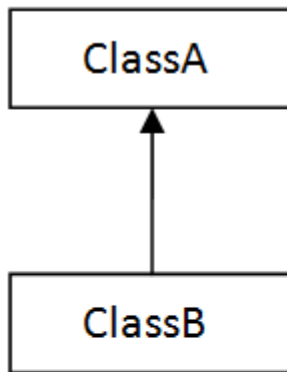
- An Object acquires all the properties and behavior of a parent
- New classes can be built upon the existing classes
- Methods and fields of the parent class can be reused
- Runtime polymorphism can be achieved

# Rules

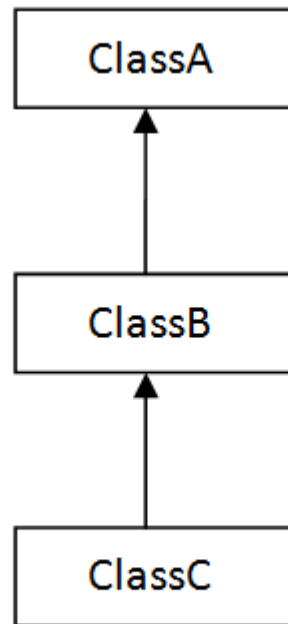


- All data members of the superclass are also data members of the subclass. Similarly, the methods of the superclass are also the methods of the subclass.
- The private members of the superclass cannot be accessed by the members of the subclass directly.
- The subclass can directly access the public members of the superclass.

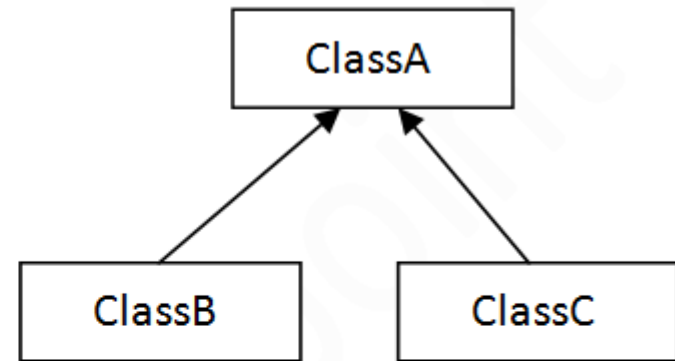
# Types of Inheritance



1) Single

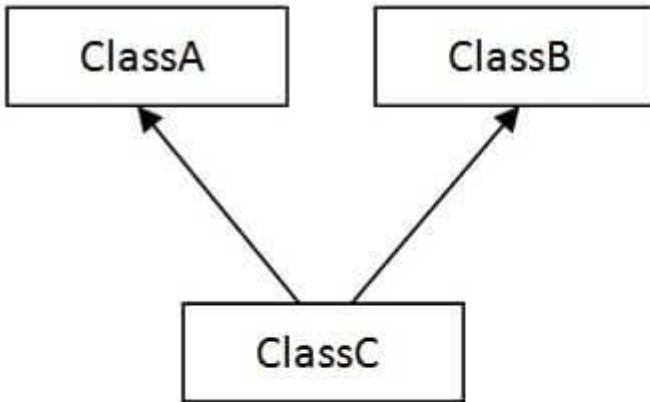


2) Multilevel

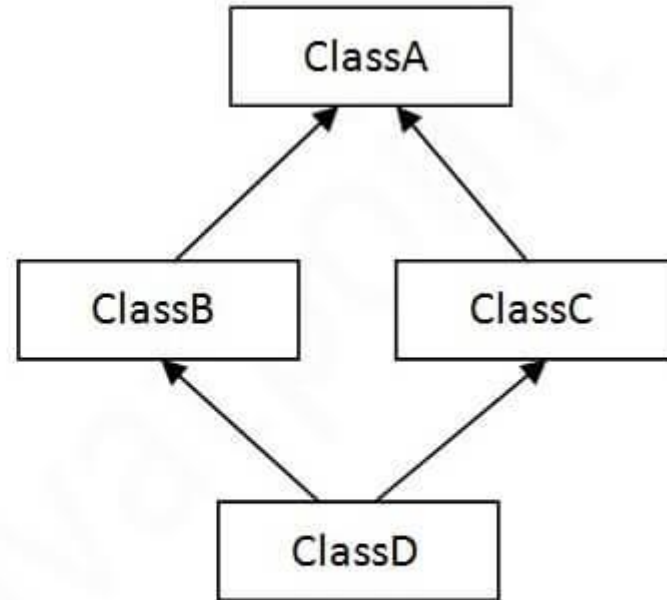


3) Hierarchical

# Types of Inheritance



4) Multiple



5) Hybrid

# Why is Multiple Inheritance not Supported?



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



# **‘Super’ Keyword**



# Immediate parent class instance variable



```
class Animal{
String color="white";
}
class Dog extends Animal{
String color="black";
void printColor(){
System.out.println(color);//prints color of Dog class
System.out.println(super.color);//prints color of Animal class
}
}
class TestSuper1{
public static void main(String args[]){
Dog d=new Dog();
d.printColor();
}}
```

black  
white

# Invoke parent class method

```
class Animal{
    void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
    void eat(){System.out.println("eating bread...");}
    void bark(){System.out.println("barking...");}
    void work(){
        super.eat();
        bark();
    }
}
class TestSuper2{
    public static void main(String args[]){
        Dog d=new Dog();
        d.work();
    }
}
```

eating...  
barking...

# Invoke parent class constructor

```
class Animal{  
    Animal(){System.out.println("animal is created");}  
}  
class Dog extends Animal{  
    Dog(){  
        super();  
        System.out.println("dog is created");  
    }  
}  
class TestSuper3{  
    public static void main(String args[]){  
        Dog d=new Dog();  
    }  
}
```

animal is created  
dog is created

# Multi Level Inheritance



```
class Car{
    public Car()    {
        System.out.println("Class Car");    }
    public void vehicleType()    {
        System.out.println("Vehicle Type:
        Car");    } }

```

```
class Maruti extends Car{
    public Maruti()    {
        System.out.println("Class Maruti"); }
    public void brand()    {
        System.out.println("Brand: Maruti");}
    public void speed()    {
        System.out.println("Max: 90Kmph");
    } }

```

```
public class Maruti800 extends Maruti{
    public Maruti800()    {
        System.out.println("Maruti Model:
        800");    }
    public void speed()    {
        System.out.println("Max: 80Kmph");}

```

```
public static void main(String args[])
{
    Maruti800 obj=new Maruti800();
    obj.vehicleType();
    obj.brand();
    obj.speed();
}

```

Class Car  
Class Maruti  
Maruti Model: 800  
Vehicle Type: Car  
Brand: Maruti  
Max: 80Kmph

# Multi Level Inheritance



```
class Grandparent {  
    public void Print() {  
        System.out.println("Grandparent's  
Print()");  
    }  
}
```

```
class Parent extends Grandparent {  
    public void Print() {  
        System.out.println("Parent's  
Print()");  
    }  
}
```

```
class Child extends Parent {  
    public void Print() {  
        super.super.Print(); //Error  
        System.out.println("Child's  
Print()");  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Child c = new Child();  
        c.Print();  
    }  
}
```

# Multi Level Inheritance



```
class Grandparent {  
    public void Print() {  
        System.out.println("Grandparent's  
Print()");  
    }  
}
```

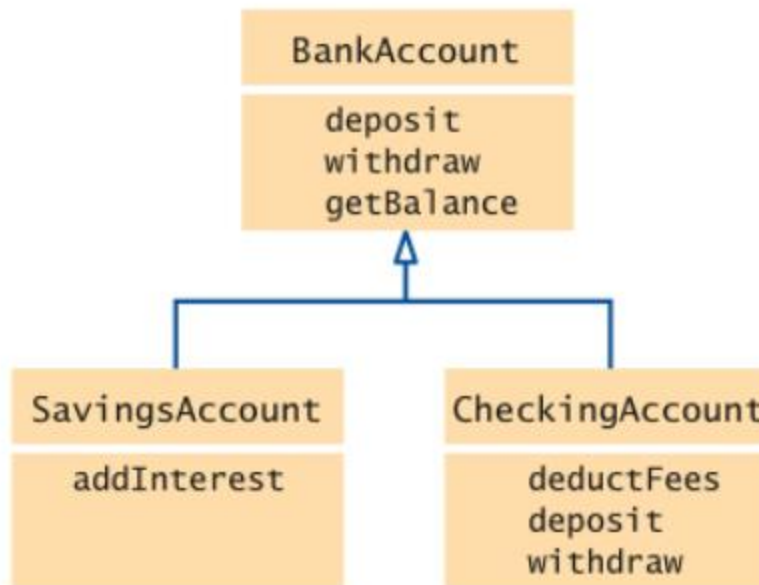
```
class Parent extends Grandparent {  
    public void Print() {  
        super.Print();  
        System.out.println("Parent's  
Print()");  
    }  
}
```

```
class Child extends Parent {  
    public void Print() {  
        super.Print();  
        System.out.println("Child's  
Print()");  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Child c = new Child();  
        c.Print();  
    }  
}
```

Grandparent's Print()  
Parent's Print()  
Child's Print()

# Bank Inheritance Scenario



# Single Inheritance - Example



```
class BankAccount{  
    private int acc;  
    private String name;  
    private float amount;
```

```
    BankAccount(int acc,String name,float amt)  
    {  
        this.acc = acc;  
        this.name = name;  
        this.amount = amt; }  
  
    void setAcc(int acc) {  
        this.acc = acc; }  
  
    void setName(String name) {  
        this.name = name; }
```

```
        float getBalance(){  
            return amount;}
```

```
        void deposit(float amount) {  
            this.amount = this.amount+amount; }
```

```
        void withdraw(float amount) {  
            if (this.amount < amount)  
                System.out.println("Insufficient  
                Funds. Withdrawal Failed");  
            else  
                this.amount=this.amount-amount; }  
        }
```



# Single Inheritance - Example



```
class SavingsAccount extends BankAccount
```

```
{
```

```
private float interest;
```

```
SavingsAccount(int acc,String name,float amt,float interest) {
```

```
super(acc,name,amt);
```

```
this.interest = interest; }
```

```
void addInterest() {
```

```
float interest = getBalance()*this.interest /100;
```

```
deposit(interest);
```

```
}
```

```
}
```

# Single Inheritance - Example



```
class TestAccount{  
    public static void main(String[] args) {  
  
        SavingsAccount sa= new SavingsAccount(111,"Ankit",5000,9);  
  
        System.out.println("Initial: "+sa.getBalance());  
  
        sa.deposit(1000);  
        System.out.println("After Deposit: " + sa.getBalance());  
  
        sa.addInterest();  
        System.out.println("Deposit+Interest: " + sa.getBalance());  
  
        sa.withdraw(6000);  
        System.out.println("After Withdraw: " + sa.getBalance());    }}
```

Initial: 5000.0

After Deposit: 6000.0

Deposit+Interest: 6540.0

After Withdraw: 540.0

# Ques



- Does a subclass object creation always lead to parent class object creation? What happens at the time of child class creation?
- Does constructor execution always lead to object creation?
- If `super()` is not defined in child class constructor, and parent class has overloaded constructor, which constructor of parent is called?
- If `super()` is not defined in child class, and parent class also does not have a constructor with no arguments, what happens?
- How can we access an overridden method() from its grand parent class.
- What is by run time polymorphism and how is it different from compile time polymorphism?