Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

SAIRAM DIGITAL RESOURCES

YEAR II
SEM III

CS8392

OBJECT ORIENTED PROGRAMMING
(Common to CSE, EEE, EIE, ICE, IT)

UNIT NO 2

INHERITANCE AND INTERFACES

2.1 Inheritance-Super classes-Sub classes

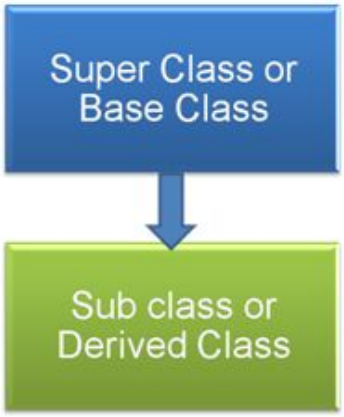
COMPUTER SCIENCE & ENGINEERING

Inheritance-Super classes-Sub classes

Inheritance Introduction:

- An object acquires all the properties and behaviors of a parent object, it is known as inheritance.
- One class is allowed to acquire (inherit) the features (fields and methods) of another class.
- It represents the IS-A relationship which is also known as Parent-Child relationship.
- Super Class: The class whose features are inherited is known as superclass (or a base class or a parent class).
- Sub Class: The class that inherits the other class is known as subclass (or a derived class, extended class, or child class).

Inheritance-Super classes-Sub classes



Syntax for inheritance:

```
class derived-class extends base-class
{
   //methods and fields
}
```

extends keyword – we are making a new class that derives from an existing class.

Example :

```
class Parent
{
   //methods and fields
}

class Child extends Parent
{
   //methods and fields  of Parent class acquired here
//methods and fields  Child
}
```

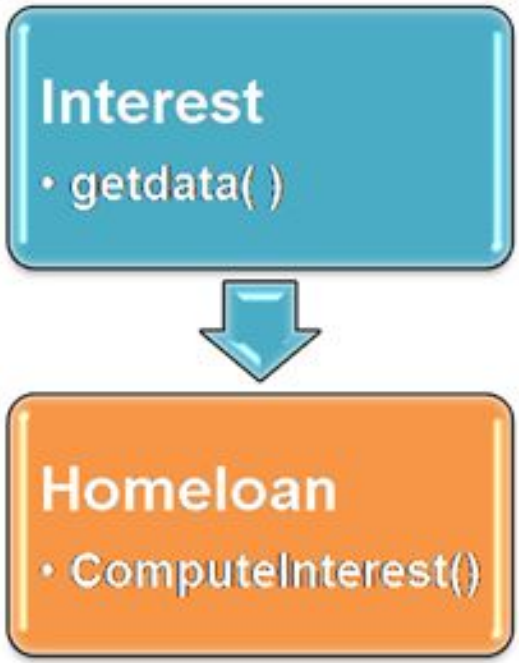
Inheritance example

```
// base class
class Interest
{
public void getdata()
  {
  //get customer details
  }
}

// Subclass
class Homeloan extends Interest
{
  public void ComputeInterest()
  {
        System.out.println("Homeloan: interest");
    }
}
```

```
class Bank  //main method class
{
  public static void main(String[] args)
        {
Homeloan m= new Homeloan();
m.getdata();
m.ComputeInterest();

        }
}
```



Inheritance IS-A Relationship

Let us see how the extends keyword is used to achieve inheritance. **IS-A is used to represent this object is a type of that object.**

For ex, consider the below program.

```
public class Animal {
}

public class Mammal extends Animal {
}

public class Reptile extends Animal {
}

public class Dog extends Mammal {
}
```

With the help of extends keyword, subclasses will be able to inherit all the properties of superclass except for the private properties of the superclass.

In this example,

- Animal is the superclass of Mammal and Reptile
- Mammal and Reptile are subclasses of Animal class
- Dog is the subclasses of both Mammal and Animal classes

In IS-A relationship we can say

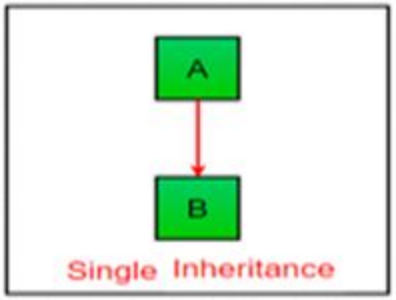
- Mammal IS-A Animal
- Reptile IS-A Animal
- Dog IS-A Mammal
- Hence Dog IS-A Animal as well
-

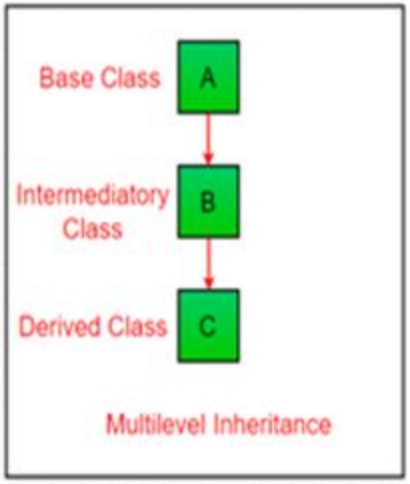
Inheritance-Super classes-Sub classes

Types of inheritance :

- Single inheritance
- Multilevel inheritance
- Hierarchical inheritance
- Multiple inheritance (through interface)
- Hybrid inheritance (through interface)

Inheritance-Super classes-Sub classes: Types of inheritance

Single inheritance
 	
One class extends another class (one class only).
	public class A{
	# variable and methods
	}
	public class B extends A{
	# variable and methods of A
	# variable and methods of B
	}
		

Multilevel inheritance
 	
In Multilevel Inheritance, one class can inherit from a derived class.
	public class A{
	# variable and methods
	}
	public class B extends A{
	# variable and methods of A
	# variable and methods of B
	}
	public class C extends B{
	# variable and methods of A
	# variable and methods of B
	# variable and methods of C
	}
		

Inheritance Example : Single Inheritance

Example : Single Inheritance

```
class A{
public void displayA()
  {
System.out.println("display() method of Class A");
  }
}
 class B extends A{
public void displayB()
  {
System.out.println("display() method of Class B");
  }
}
public class single
  {
public static void main(String args[])
  {
  B obj = new B();
  obj.displayA();
  obj.displayB();   }     }
```

Inheritance Example : Multilevel Inheritance

Example : Multilevel Inheritance

```
class A
{
public void displayA()
{
System.out.println("This is displayA()
method of Class A");
}
}

class B extends A
{
public void displayB()
{
System.out.println("This is displayB()
method of Class B");
}
}
```

```
class C extends B
{
public void displayC()
{
System.out.println("This is displayC()
method of Class C");
}
}
public class multilevel
{
public static void main(String args[])
{

C obj = new C();

obj.displayA();
obj.displayB();
obj.displayC();
}
}
```

Inheritance-Super classes-Sub classes: Types of inheritance

Hierarchical Inheritance

In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one sub class.

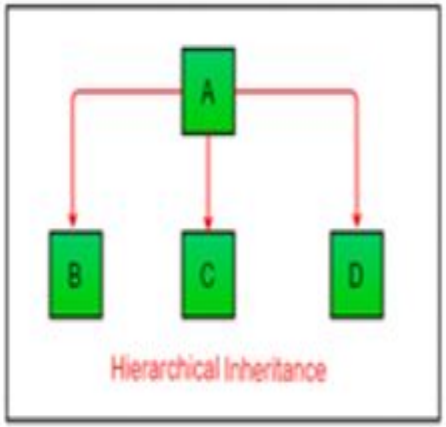
One class is inherited by many sub classes.

```
public  class A{
# variable and methods
}

public class B extends A{
# variable and methods of A
# variable and methods of B
}

public class C extends A{
# variable and methods of A
# variable and methods of C
}

public class D extends A{
# variable and methods of A
# variable and methods of D
}
```



Inheritance Example : Hierarchical Inheritance

Example : Hierarchical Inheritance

```
class A{
public void displayA()
{
System.out.println("This is displayA() method of
Class A");
}
}
 class B extends A{
public void displayB()
{
System.out.println("This is displayB() method of
Class B");
}
}
 class C extends A{
public void displayC()
{
System.out.println("This is displayC() method of
Class C");
}
}
```

```
class D extends A{
public void displayD()
{
System.out.println("This is displayC() method of
Class C");
}
}
public class hierarchy {
public static void main(String args[])
{
B obj1 = new B();
C obj2 = new C();
D obj3 = new D();
obj1.displayA();
obj1.displayB();
      obj2.displayA();
      obj2.displayC();
            obj3.displayA();
            obj3.displayD();
}
}
```

Inheritance-Super classes-Sub classes: Types of inheritance

Multiple Inheritance (Through Interfaces) :

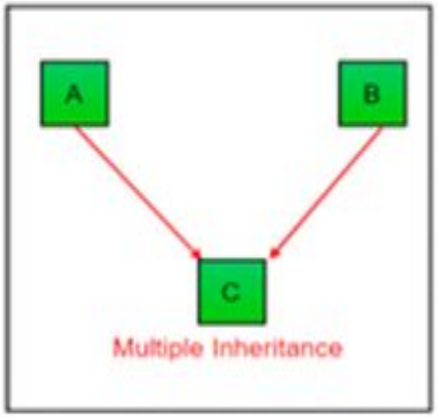
In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes.

Java does not support multiple inheritances with classes.

```
interface A
{
# variable and methods of A
}

interface B
{
# variable and methods of B
}

public class C implements A,B
{
# variable and implemented methods of A
# variable and implemented methods of B
# variable and methods of C
}
```



Inheritance Example : Multiple Inheritance

Example : Multiple Inheritance (Through Interfaces)

```
interface A
{
public void display();
}
interface B
{
public void show();
}
class C implements A,B
{
//overriding display
public void display()
{
System.out.println("display() method
implementation");
}
//Overriding show
public void show()
{
System.out.println("show() method
implementation");
}
}   // end of class C
public class multiple
{
public static void main(String args[])
{
C m = new C();
m.display();
m.show();
}
}
```

Inheritance-Super classes-Sub classes: Types of inheritance

Hybrid Inheritance (Through Interfaces) :

Hybrid Inheritance is the combination of Single and /or Hierarchical and /or Multiple Inheritance.

It is a mix of two or more of the above said types of inheritance.

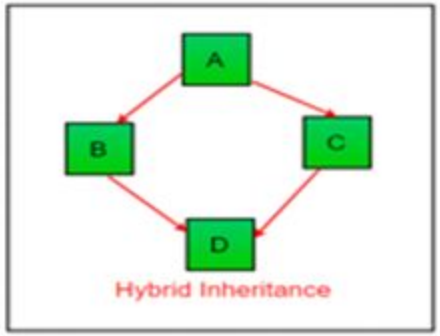
```
interface A
{
// variable and methods of A

}

interface B extends A
{
// variable and methods of A
// variable and methods of B
}

interface C extends A
{
// variable and methods of A
// variable and methods of C
}

public class D implements B,C
{
// variable and implemented methods of
A,,B and C
}
```



Inheritance Example : Hybrid Inheritance

Example : Hybrid Inheritance (Through Interfaces)

```
interface A
{
public void display();
}
interface B extends A
{
public void show();
}
interface C extends A
{
public void print();
}
public class D implements B,C
{
//overriding display
public void display()
{
System.out.println("display() method
implementation");
}
//Overriding show
public void show()
{
System.out.println("show() method
implementation");
}
//Overriding print
public void print()
{
System.out.println("print() method
implementation");
}
public static void main(String args[])
{
D m = new D();
m.display();
m.show();
m.print();
}
}
```

Inheritance Summary

- It allows **coding reusability**.
- It forms the **backbone of** object oriented programming and **Java**.
- It forms a **IS-A relationship** between superclass and subclass using inheritance.
  ie. we can use any subclass object in the place of superclass object.
- **One subclass can extend only one superclass** in Java but it can implement the multiple interfaces.
- A **private member of the superclass can not be inherited** in subclass.
- The **constructor in Java is not inherited by the subclass**.
- **Multiple inheritance is not supported** in Java but it can be achieved using Interface. One **class can implement multiple interfaces.**
- Java **class only implements the interface** and it never extends the interface.
- Inheritance in Java is achieved through two keywords:
  - **extends** – one class extends another class when a class wants to use the property of that class.
  - **implements** – used to implement an interface. This keyword is used to describe a special type of class that only contains abstract methods.

Inheritance Video Link

https://youtu.be/nixQyPIAnOQ

https://youtu.be/jJ8L3SeFy_E