INHERITANCE

Introduction to Inheritance

Dict*ion*ary Definition:

"Inheritance is a process which involves the passing-on of property (money or objects) from one generation to another, usually within the family, generally from grandparents to their adult children (heirs)."

Synonym: Legacy, endowment

What is Inheritance?

OOP Definition1:

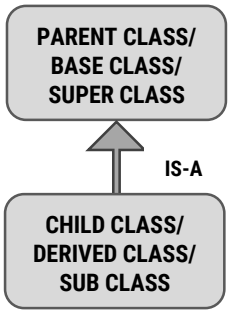
*"Inheritance is the process, by which class can acquire (reuse) the **properties** and **methods** of another class."*

OOP Definition2:

*"The mechanism of a class to derive **properties** and **characteristics** from another class is called Inheritance."*

Base Class Definition: *"The class whose properties are inherited by sub class is called Base Class/Super class/Parent class."*

Derived Class Definition: *"The class that inherits properties from another class is called Sub class/Derived Class/Child class."*



- Inheritance is implemented using **super class** and **sub class** relationship (**IS-A relationship**) in object-oriented languages.
  Example:
  i. **Land Vehicle** class & **Water Vehicle** class **IS-A** child class of **Vehicle** class
  ii. **Doctor** class and **Teacher** class **IS-A** child class of **Person** class
  iii. Engineering College and Medical College **IS-A** child class of College class
- It is the most important feature of Object Oriented Programming.

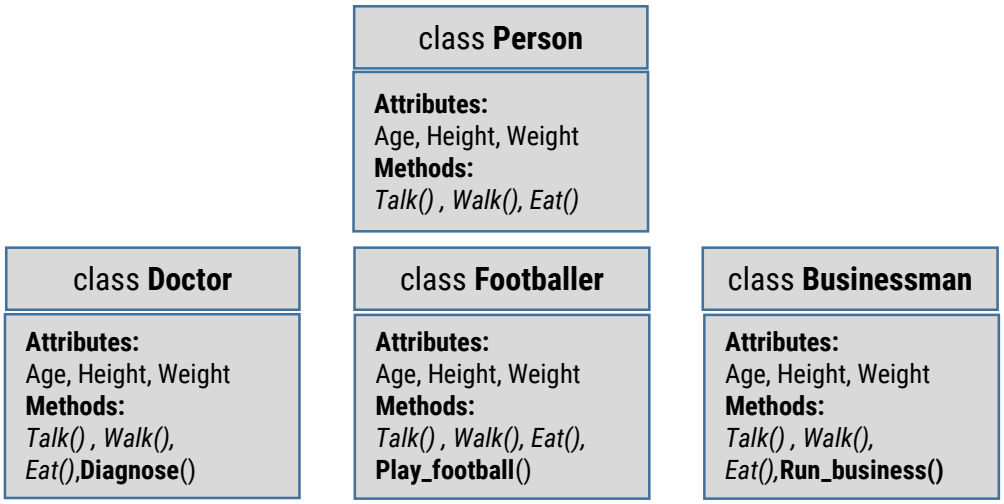


Figure 1: Without Inheritance Relationship

INHERITANCE

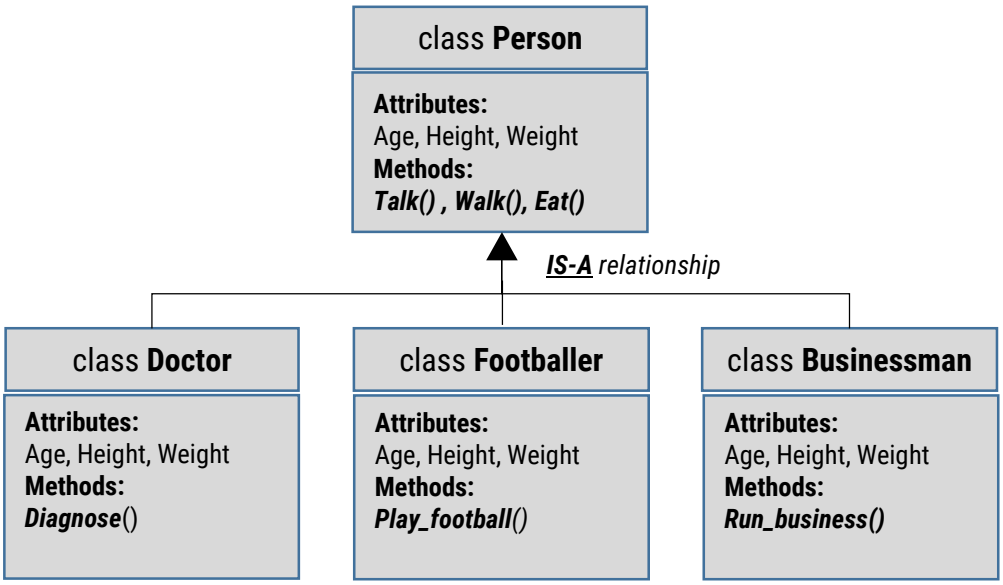


Figure 1: With Inheritance Relationship

Syntax & Sample Program in Java

How to Implement Inheritance in java?

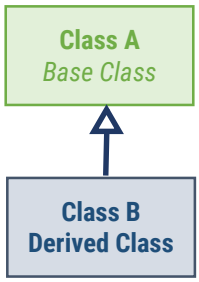
- To inherit a class, simply incorporate the definition of one class into another by using the ***extends*** keyword.

Syntax:

```
class subclass-name extends superclass-name {
     // body of class…
}
```

Example: Here keyword "***extends***" is used to create a subclass of A

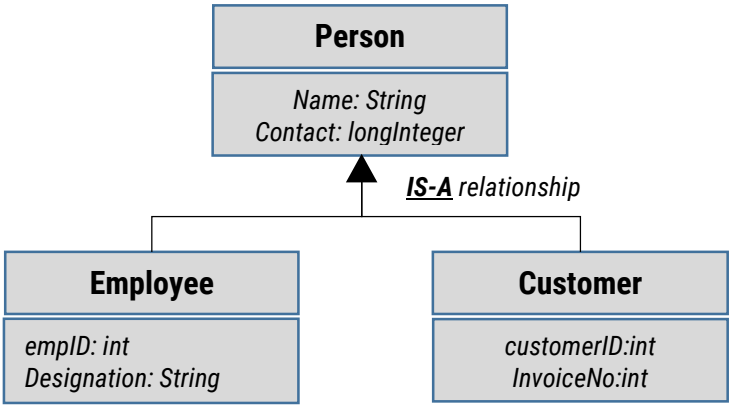
```
class A
{
  //SuperClass or ParentClass or BaseClass
}
class B extends A
{
  //SubClass or ChildClass or DerivedClass
}
```



INHERITANCE

How does the Inheritance works?

- To understand working of Inheritance, Let's implement an example of inheritance with Java language.



Write a java program to implement above inheritance relationship:

```
1.      class Person
2.      {
3.            public String name;
4.            protected long contact;
5.            private int p_id;//can't be accessed by child class
6.      }
7.      class Employee extends Person
8.      {
9.            int empID;
10.           String designation;
11.           name="Emp1";//reusability
12.     }
13.     class Customer extends Person
14.     {
15.           int customerID;
16.           int invoiceNo;
17.           name="Cust1";//reusability
18.     }
```

- The technique of creating a new class by using an existing class functionality is called inheritance in Java.
- In this above example, class Employee and Customer extends the functionality of Person class by deriving name and contact attribute from parent class.
- The child class can access all the attributes of parent class except private members.

INHERITANCE

Types of Inheritance in Java

There are five types of inheritance as listed below.

1. **Single Inheritance**
   In Single Inheritance one class extends another class (one parent class and one child class).
2. **Multiple Inheritance**
   Multiple Inheritance is one of the inheritances in Java types where one class extending more than one class. Java does not support multiple inheritance.
3. **Multilevel Inheritance**
   In Multilevel Inheritance, one class can inherit from a derived class. Hence, the derived class becomes the base class for the new class.
4. **Hierarchical Inheritance**
   In Hierarchical Inheritance, one class is inherited by many sub classes.
5. **Hybrid Inheritance**
   Hybrid inheritance is one of the inheritance types in Java which is a combination of Single and Multiple inheritance.

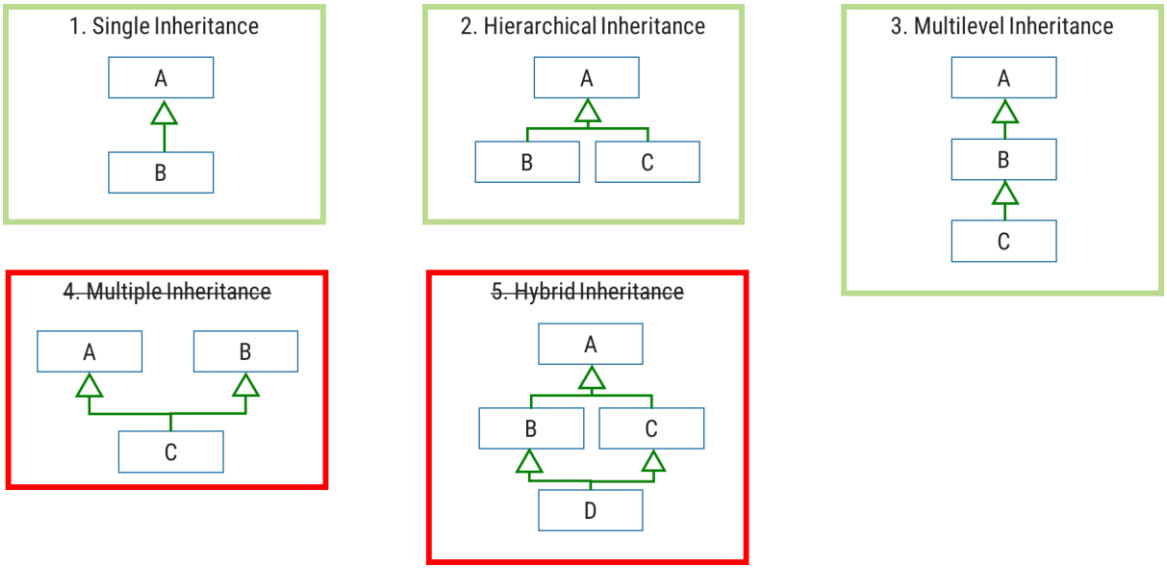


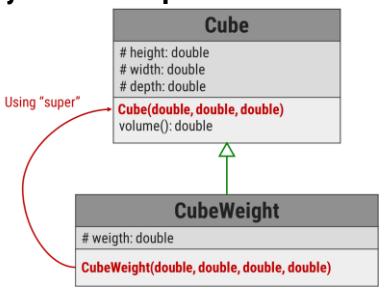
Figure 3: Types of Inheritance in Java

***Note**: Multiple and Hybrid Inheritance is not supported in Java with the Class Inheritance, we can still use those Inheritance with Interface.*

Super keyword in Java

- Whenever a subclass needs to refer to its immediate superclass, it can do so by use of the keyword super.
- Super has two general forms:
  1. Calls the superclass constructor.
  2. Used to access a members (i.e. instance variable or method) of the superclass.

Implement Inheritance using 'Super' keyword to call parent class Constructor



INHERITANCE

```
1. class Cube{
2.       protected double height,width,depth;
3.       Cube(double h,double w,double d){
4.             System.out.println("Constructor: CUBE");
5.             height=h;
6.             width=w;
7.             depth=d;
8.       }
9.       double volume(){
10.            return height*width*depth;
11.                        }
12.      }//class Cube

13.   class CubeWeight extends Cube{
14.            double weigth;
15.            CubeWeight(double h,double w,double d, double m){
16.              super(h,w,d); //call superclassConstructor
17.              System.out.println("Constructor:CUBEWEIGTH");
18.              weigth=m;
19.            }
20.      }

21.   class CubeInheritSuper{
22.      public static void main(String[] args) {
23.      CubeWeight cw1= new CubeWeight(10,10,10,20.5);
24.      CubeWeight cw2= new CubeWeight(100,100,100,200.5);
25.      System.out.println("cw1.volume()="+cw1.volume());
26.      System.out.println("cw1.weigth="+cw1.weigth);
27.      System.out.println("cw2.volume()="+cw2.volume());
28.      System.out.println("cw2.weigth="+cw2.weigth);
29.        }
30.      }
```

Output

```
Constructor:CUBE
Constructor:CUBEWEIGTH
Constructor:CUBE
Constructor:CUBEWEIGTH
cw1.volume()=1000.0
cw1.weigth=20.5
cw2.volume()=1000000.0
cw2.weigth=200.5
```

Using 'super' to access members

- This second form of **super** is most applicable to situations in which member names of a **subclass hide** members by the same name in the **superclass.**

Syntax:

```
super.member; //member can be either a method or an instance variable.
```

INHERITANCE

Example: Java program using **super** to access members

```
1. class A{
2.       int i;
3. }
4. class B extends A{
5.       int i,k;
6.       B(int a,int b){
7.             super.i=a;//refer to member of super class A
8.             this.i=b;
9.       }
10.      void show(){
11.            System.out.println("super.i="+super.i);
12.            System.out.println("this.i="+this.i);
13.       }
14.      }
15. class SuperMemberDemo{
16.      public static void main(String[] args){
17.                  B b= new B(12,56);
18.                  b.show();
19.        }
20. }
```

Output:
super.i=12
this.i=56

Points to remember for Super Keyword

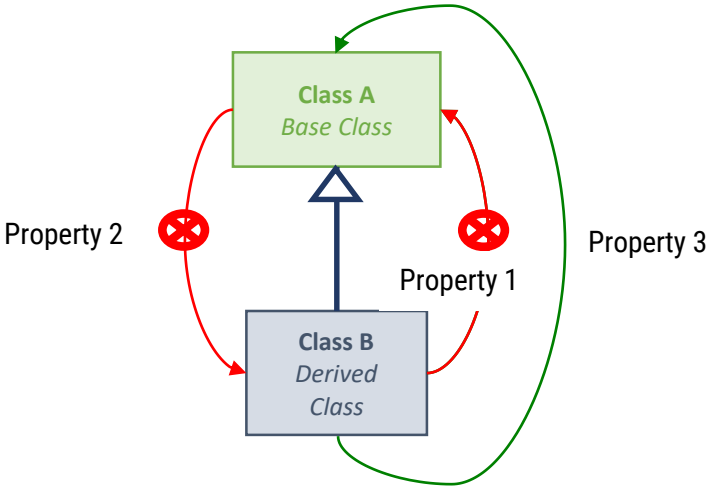
- When a subclass calls super( ), it is calling the constructor of its immediate superclass. This is true even in a multileveled hierarchy.
- super( ) must always be the **first statement** executed inside a subclass constructor.
- If a constructor does not explicitly call a superclass constructor, the Java compiler automatically inserts a call to the no-argument constructor of the superclass.
- The most common application of super keyword is to **eliminate the ambiguity** between members of superclass and sub class.

Property of Inheritance

Property 1: A class member that has been declared as private will remain private to its class. It is not accessible by any code outside its class, including subclasses.
Derived Class B can't access **private members** of Base class A

Property 2: Base class A can't access attributes and methods of Derived Class B.

Property 3: Derived Class B can access attributes and methods of Base class A



INHERITANCE

Advantages of Inheritance

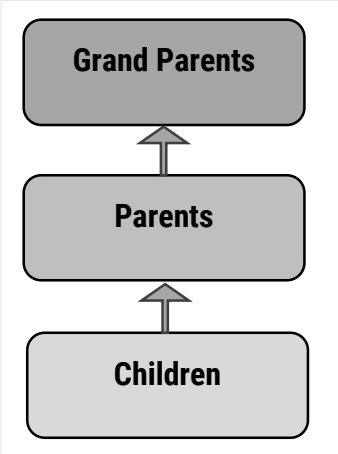
- Promotes reusability
    - When an existing code is reused, it leads to less development and maintenance costs.
- It is used to generate more dominant objects.
- Avoids duplicity and data redundancy.
- Inheritance makes the sub classes follow a standard interface.

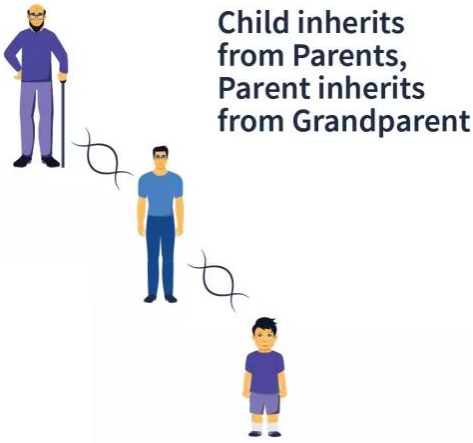
Application of Inheritance

- Reusability of code
- To implement polymorphism at run time (method overriding).
- Method overriding is also known as runtime polymorphism. Hence, we can achieve Polymorphism in Java with the help of inheritance.

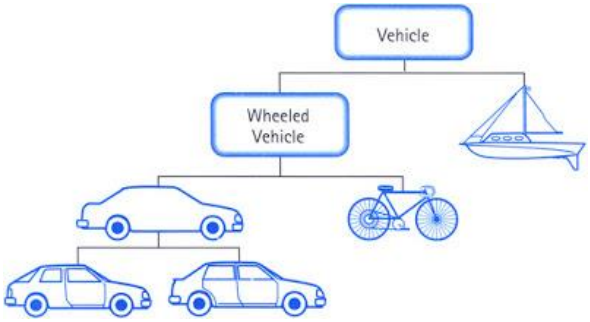
Three Real World Examples of Inheritance

Example-1: Human Family Hierarchy is the best example of Inheritance in real world, where child class may inherits some property/behaviour/ appearance /intelligence etc. from parent class and also contain some of its exclusive properties. This is an example of Multilevel Inheritance.



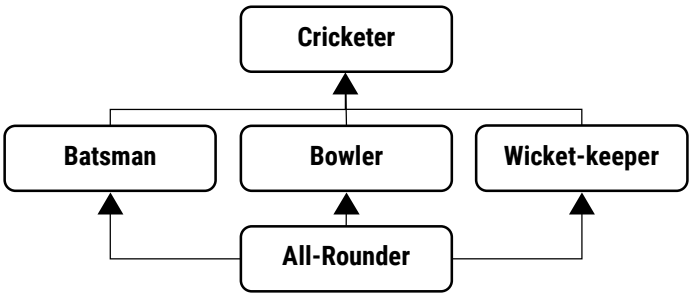


Example-2: This is an example of Hierarchical Inheritance, where Vehicle is the parent class and wheeled-Vehicle and Water-Vehicle are the child classes. This child class is further derived into Car and Bicycle Class.



Example-3: Hybrid Inheritance example, where Cricketer is the parent class of Batsman, Bowler and Wicket-keeper respectively and All-Rounder is the leaf class.

INHERITANCE



Interview Questions: Inheritance

1. **Which class in Java is superclass of every other class?**
   In Java, **Object class** is the superclass of every other java class.
   i.e.: java.lang.Object is the parent class of all the java classes

2. **Can a class in java extends itself?**
   No, a class cannot extend itself

3. **Can a class extend more than one class?**
   No, a class cannot extend more than one class

4. **Differentiate 'extends' and 'implements' keyword in Java.**
   Extends: extends is a keyword that is used for developing the inheritance between two classes and two interfaces.
   Implements: implements keyword is used for developing the inheritance between a class and interface.

5. **Can we assign superclass to subclass?**
   No, we can't assign superclass to subclass.

6. **Can a class extend more than one class?**
   No, one class can extend only a single class.

7. **Are static members inherited to subclass in Java?**
   Static block cannot be inherited to its subclass.
   A static method of superclass is inherited to the subclass as a static member and non-static method is inherited as a non-static member only.

8. **Can we extend (inherit) final class?**
   No, a class declared with final keyword cannot be inherited.

9. **Can a final method be overridden?**
   No, a final method cannot be overridden.

10. **Can we inherit private members of base class to its subclass?**
    No, all the members (public, protected and default) of superclass, except private members are accessible by sub-class.

11. **What are the advantages of inheritance in Java?**
    - Promotes reusability
      When an existing code is reused, it leads to less development and maintenance costs.
    - It is used to generate more dominant objects.
    - Avoids duplicity and data redundancy.
    - Inheritance makes the sub classes follow a standard interface.

INHERITANCE

12. What is Multiple inheritance in Java?

A class that has many super classes is known as multiple inheritance. In other words, when a class extends multiple classes, it is known as multiple inheritance.

13. Why multiple inheritance is not supported in java through class?

Multiple inheritance means that one class extends two or more super classes or base classes but in Java, one class cannot extend more than one class simultaneously. At most, one class can extend only single class. Therefore, to reduce ambiguity, complexity, and confusion, Java does not support multiple inheritance through classes.Multiple inheritance can be achieved in java by implementing interfaces instead of class.

14. How does Multiple inheritance implement in Java?

Multiple inheritance can be implemented in Java by using interfaces. A class cannot extend more than one class but a class can implement more than one interface.

Example:

```
class B extends A
{
        …
}
```

```
class B extends A implements interface1,interface2,…
{
        …
}
```

15. What is Hybrid inheritance in java? How will you achieve it?

A hybrid inheritance in java is a combination of single and multiple inheritance. It can be achieved through interfaces.

16. How will you restrict a member of a class from inheriting its subclass?

We can restrict members of a class by declaring them private because the private members of superclass are not available to the subclass directly. They are only available in their own class.

17. Can we access subclass members if we create an object of superclass?

No, we can access only superclass members but not the subclass members.

18. Can we access both superclass and subclass members if we create an object of subclass?

Yes, we can access both superclass and subclass members.

19. Is interface inherited from the Object class?

No.

20. Can you override a final method?

No, method declared as 'final' cannot be overridden.

21. What does Java's Super Keyword mean?

A reference variable used to refer to the immediate parent class object in Java is the super keyword.
When you create an instance of a subclass, an implicit instance of the parent class is also produced and referred to by the super reference variable.

22. Differentiate inheritance from encapsulation.

Inheritance: hers one class acquires another class to promote reusability.
Encapsulation: Binds code and data together for securing data access.

INHERITANCE

23. Differentiate inheritance from composition.

Inheritance and composition are two programming techniques developers use to establish relationships between classes and objects. Whereas inheritance derives one class from another, composition defines a class as the sum of its parts.

24. super vs. this keyword

'super' keyword

- is used to access methods of the parent class while **'this'** is used to access methods of the current class.
- '**super'** is also used to invoke super-class's method or constructor.
- The most common use of super keyword is that it eliminates the confusion between the superclasses and subclasses that have methods with same name.

'this' keyword

- is used to refer instance variable of current class
- to invoke or initiate current class constructor
- can be passed as an argument in the method call
- can be passed as argument in the constructor call
- can be used to return the current class instance

25. Inheritance vs. Abstraction

Abstraction is an object oriented concept which is used to simplify things by abstracting details. It is used during System design. On the other hand, Inheritance allows code reuse. You can reuse the functionality you have already programmed by using Inheritance.

26. Inheritance vs. Polymorphism

Polymorphism allows the object to decide which form of the function to implement at compile-time as well as run-time, while inheritance performs reusability.
Inheritance is applied by classes, while polymorphism is applied by methods/functions.

27. Can an interface extends more than one interfaces in Java?

Yes, unlike classes, an interface can extend more than one interface in Java.

28. What will happen if a class extends two interfaces and they both have a method with same name and signature?

In this case, a conflict will arise because the compiler will not able to link a method call due to ambiguity. You will get a compile time error in Java.

29. Does a class inherit the constructors of its superclass?

A class does not inherit constructors from any of its superclass's.

30. Can you give a few examples of final classes defined in Java API?

java.lang.String, java.lang.Math are the 'final' classes.