SNS COLLEGE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
COIMBATORE – 35
DEPARTMENT OF COMPUTER SIENCE AND ENGINEERING

UNIT 3

Types of Inheritance in Java

Inheritance is the most powerful feature of object-oriented programming. It allows us to inherit the properties of one class into another class. In this section, we will discuss **types of inheritance in Java** in-depth with real-life examples. Also, we will create Java programs to implement the concept of different types of inheritance.

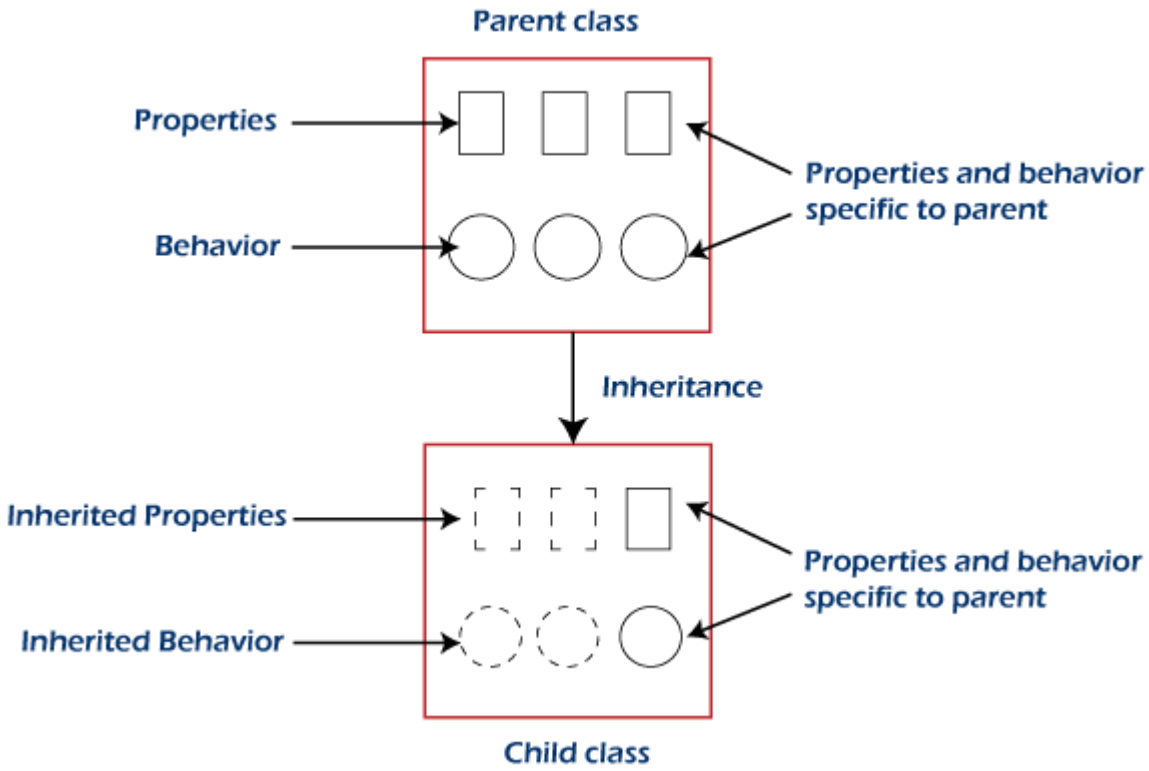
Inheritance

Inheritance is a mechanism of driving a new class from an existing class. The existing (old) class is known as **base class** or **super class** or **parent class**. The new class is known as a **derived class** or **sub class** or **child class**. It allows us to use the properties and behavior of one class (parent) in another class (child).

A class whose properties are inherited is known as **parent class** and a class that inherits the properties of the parent class is known as **child class**. Thus, it establishes a relationship between parent and child class that is known as parent-child or **Is-a** relationship.

Suppose, there are two classes named **Father** and **Child** and we want to inherit the properties of the Father class in the Child class. We can achieve this by using the **extends** keyword.

```
1.  //inherits the properties of the Father class
2.  class Child extends Father
3.  {
4.  //functionality
5.  }
```



When we should use inheritance?

Inheritance provides the **reusability** of code especially when there is a large scale of code to reuse. It also establishes the relationship between different classes that is known as a **Is-a** relationship. We can also use it if we want to achieve **method overriding**.

Points to Remember

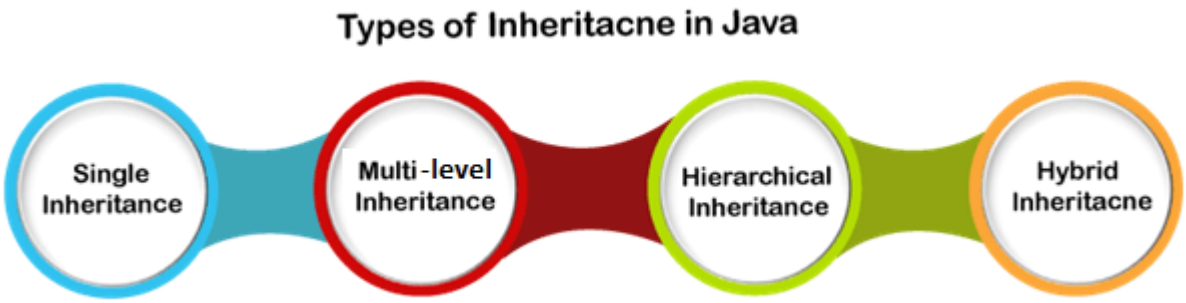
- Constructor cannot be inherited in Java.
- Private members do not get inherited in Java.
- Cyclic inheritance is not permitted in Java.
- Assign parent reference to child objects.
- Constructors get executed because of super() present in the constructor.

Types of Inheritance

Java supports the following four types of inheritance:

- Single Inheritance
- Multi-level Inheritance

- Hierarchical Inheritance
- Hybrid Inheritance

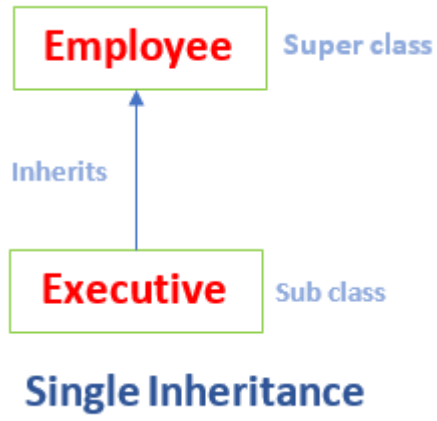


> Note: Multiple inheritance is not supported in Java.

Let's discuss each with proper example.

Single Inheritance

In single inheritance, a sub-class is derived from only one super class. It inherits the properties and behavior of a single-parent class. Sometimes it is also known as **simple inheritance**.



In the above figure, Employee is a parent class and Executive is a child class. The Executive class inherits all the properties of the Employee class.

Let's implement the single inheritance mechanism in a Java program.

Executive.java

```
1.  class Employee
2.  {
3.  float salary=34534*12;
```

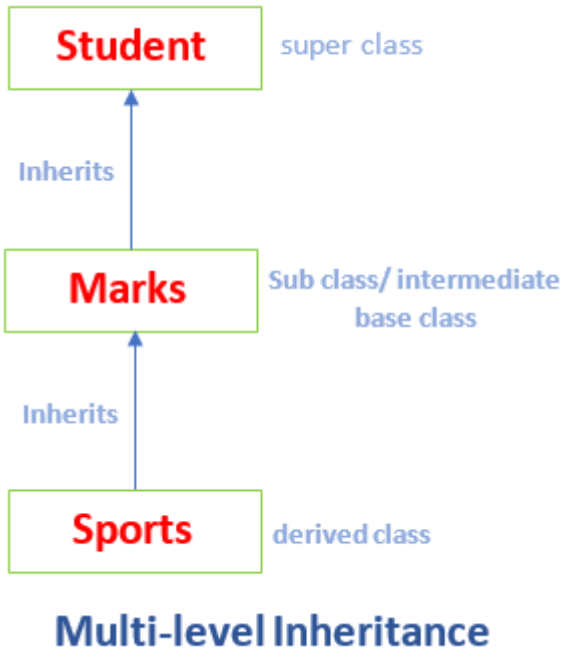
```
4.      }
5.      public class Executive extends Employee
6.      {
7.      float bonus=3000*6;
8.      public static void main(String args[])
9.      {
10.     Executive obj=new Executive();
11.     System.out.println("Total salary credited: "+obj.salary);
12.     System.out.println("Bonus of six months: "+obj.bonus);
13.     }
14.     }
```

Output:

```
Total salary credited: 414408.0
Bonus of six months: 18000.0
```

Multi-level Inheritance

In **multi-level inheritance**, a class is derived from a class which is also derived from another class is called multi-level inheritance. In simple words, we can say that a class that has more than one parent class is called multi-level inheritance. Note that the classes must be at different levels. Hence, there exists a single base class and single derived class but multiple intermediate base classes.



In the above figure, the class Marks inherits the members or methods of the class Students. The class Sports inherits the members of the class Marks. Therefore, the Student class is the parent class of the class Marks and the class Marks is the parent of the class Sports. Hence, the class Sports implicitly inherits the properties of the Student along with the class Marks.

Let's implement the multi-level inheritance mechanism in a Java program.

MultilevelInheritanceExample.java

```
1.  //super class
2.  class Student
3.  {
4.  int reg_no;
5.  void getNo(int no)
6.  {
7.  reg_no=no;
8.  }
9.  void putNo()
10. {
11. System.out.println("registration number= "+reg_no);
12. }
13. }
14. //intermediate sub class
15. class Marks extends Student
16. {
17. float marks;
18. void getMarks(float m)
19. {
20. marks=m;
21. }
22. void putMarks()
23. {
24. System.out.println("marks= "+marks);
25. }
26. }
27. //derived class
28. class Sports extends Marks
29. {
```

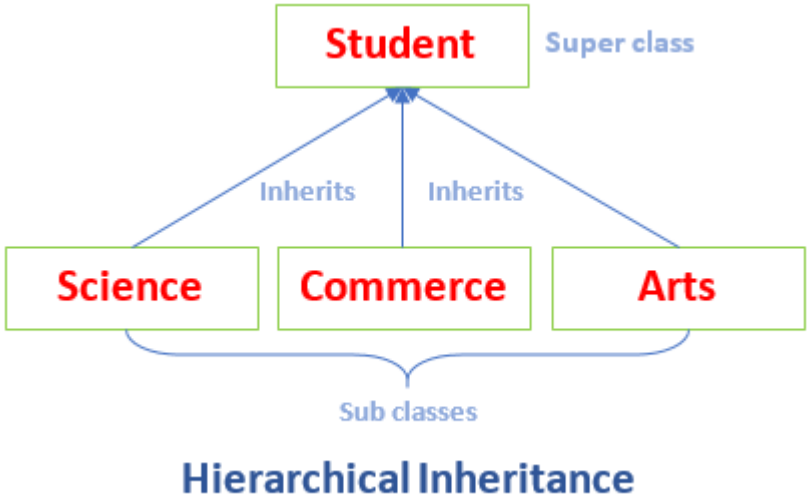
```
30.     float score;
31.     void getScore(float scr)
32.     {
33.     score=scr;
34.     }
35.     void putScore()
36.     {
37.     System.out.println("score= "+score);
38.     }
39.     }
40.     public class MultilevelInheritanceExample
41.     {
42.     public static void main(String args[])
43.     {
44.     Sports ob=new Sports();
45.     ob.getNo(0987);
46.     ob.putNo();
47.     ob.getMarks(78);
48.     ob.putMarks();
49.     ob.getScore(68.7);
50.     ob.putScore();
51.     }
52.     }
```

Output:

```
registration number= 0987
marks= 78.0
score= 68.7
```

Hierarchical Inheritance

If a number of classes are derived from a single base class, it is called **hierarchical inheritance**.



Hierarchical Inheritance

In the above figure, the classes Science, Commerce, and Arts inherit a single parent class named Student.

Let's implement the hierarchical inheritance mechanism in a Java program.

HierarchicalInheritanceExample.java

```
1.  //parent class
2.  class Student
3.  {
4.  public void methodStudent()
5.  {
6.  System.out.println("The method of the class Student invoked.");
7.  }
8.  }
9.  class Science extends Student
10. {
11. public void methodScience()
12. {
13. System.out.println("The method of the class Science invoked.");
14. }
15. }
16. class Commerce extends Student
17. {
18. public void methodCommerce()
19. {
20. System.out.println("The method of the class Commerce invoked.");
```

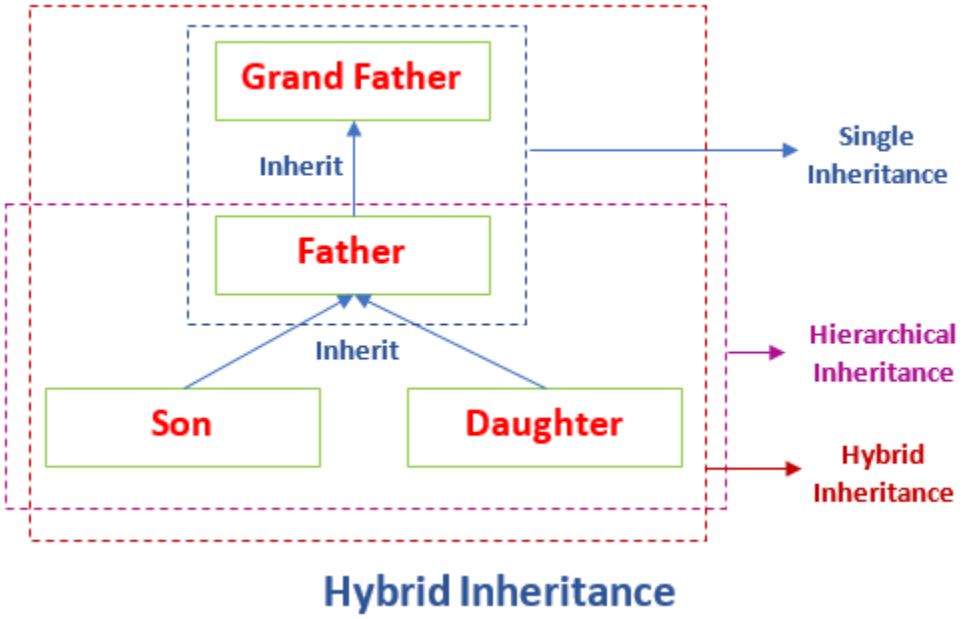
```
21.         }
22.         }
23.         class Arts extends Student
24.         {
25.         public void methodArts()
26.         {
27.         System.out.println("The method of the class Arts invoked.");
28.         }
29.         }
30.         public class HierarchicalInheritanceExample
31.         {
32.         public static void main(String args[])
33.         {
34.         Science sci = new Science();
35.         Commerce comm = new Commerce();
36.         Arts art = new Arts();
37.         //all the sub classes can access the method of super class
38.         sci.methodStudent();
39.         comm.methodStudent();
40.         art.methodStudent();
41.         }
42.         }
```

Output:

```
The method of the class Student invoked.
The method of the class Student invoked.
The method of the class Student invoked.
```

Hybrid Inheritance

Hybrid means consist of more than one. Hybrid inheritance is the combination of two or more types of inheritance.



Hybrid Inheritance

In the above figure, GrandFather is a super class. The Father class inherits the properties of the GrandFather class. Since Father and GrandFather represents single inheritance. Further, the Father class is inherited by the Son and Daughter class. Thus, the Father becomes the parent class for Son and Daughter. These classes represent the hierarchical inheritance. Combinedly, it denotes the hybrid inheritance.

Let's implement the hybrid inheritance mechanism in a Java program.

Daughter.java

```
1.  //parent class
2.  class GrandFather
3.  {
4.  public void show()
5.  {
6.  System.out.println("I am grandfather.");
7.  }
8.  }
9.  //inherits GrandFather properties
10. class Father extends GrandFather
11. {
12. public void show()
13. {
14. System.out.println("I am father.");
15. }
```

```
16.     }
17.     //inherits Father properties
18.     class Son extends Father
19.     {
20.     public void show()
21.     {
22.     System.out.println("I am son.");
23.     }
24.     }
25.     //inherits Father properties
26.     public class Daughter extends Father
27.     {
28.     public void show()
29.     {
30.     System.out.println("I am a daughter.");
31.     }
32.     public static void main(String args[])
33.     {
34.     Daughter obj = new Daughter();
35.     obj.show();
36.     }
37.     }
```

Output:

```
I am daughter.
```

Multiple Inheritance (not supported)

Java does not support multiple inheritances due to ambiguity. For example, consider the following Java program.

Demo.java

```
1.      class Wishes
2.      {
3.      void message()
4.      {
5.      System.out.println("Best of Luck!!");
```

```
6.      }
7.      }
8.      class Birthday
9.      {
10.     void message()
11.     {
12.     System.out.println("Happy Birthday!!");
13.     }
14.     }
15.     public class Demo extends Wishes, Birthday  //considering a scenario
16.     {
17.     public static void main(String args[])
18.     {
19.     Demo obj=new Demo();
20.     //can't decide which classes' message() method will be invoked
21.     obj.message();
22.     }
23.     }
```

The above code gives error because the compiler cannot decide which message() method is to be invoked. Due to this reason, Java does not support multiple inheritances at the class level but can be achieved through an **interface**.