

# Core Java

---

## Chapter 7 Inheritance in Java



### ICT Academy of Tamil Nadu

ELCOT Complex, 2-7 Developed Plots, Industrial Estate, Perungudi, Chennai 600 096.

Website : [www.ictact.in](http://www.ictact.in) , Email : [contact@ictact.in](mailto:contact@ictact.in) ,

Phone : 044 4290 6800 , Fax : 044 4290 6820

---

## Table of Contents

---

|                                     |          |
|-------------------------------------|----------|
| <b>1. Inheritance in Java .....</b> | <b>4</b> |
| 1.1 Inheritance .....               | 4        |
| 1.2 Inheritance in java.....        | 5        |
| 1.3 Casting.....                    | 11       |
| 1.4 Method Overriding .....         | 12       |
| 1.5 Polymorphism .....              | 15       |
| 1.6 Super .....                     | 17       |
| 1.7 The Object Class.....           | 19       |

ICTACT

---





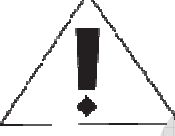


**Chapter 7**

Inheritance in Java

---

## Documentation Conventions

The following conventions are used in this guide:

| When you see this...  | This is...      |
|---|-----------------|
|    | Recall learning |
|    | Case study      |
|    | Did you know?   |
|   | Class session   |
|  | Activity        |
|  | Quiz            |
|  | Reference       |
| Indented text in different font   | Code snippet    |

## Chapter 7: Inheritance in Java

Inheritance is one of the important concepts of object-oriented programming, as it allows the creation of hierarchical classifications. By using inheritance, we can create a general class that inherits the other. This chapter deals with inheritance concepts, Polymorphism, overriding & the object classes.

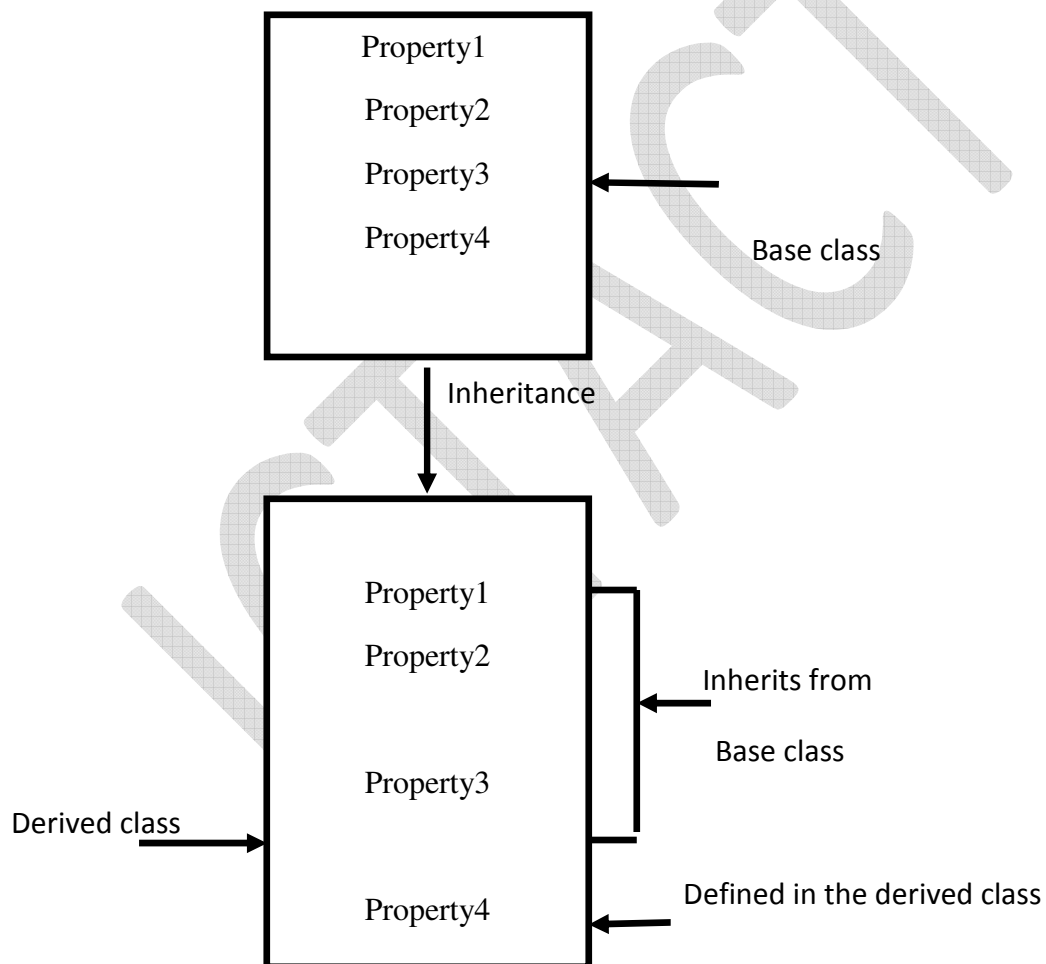
ICTACT

## 1. Inheritance in Java

### 1.1 Inheritance

Inheritance is the process of creating new classes from the existing classes. The new classes are called derived classes & the existing classes are called base classes.

The derived classes inherit all the properties of the base classes plus its own properties. The process of inheritance does not affect the base classes. The figure given below shows the process of inheritance.



### Advantages

- New classes can be derived by the user from the existing classes without any modification
- It saves time & money
- It reduces program coding time
- It increases the reliability of the program.

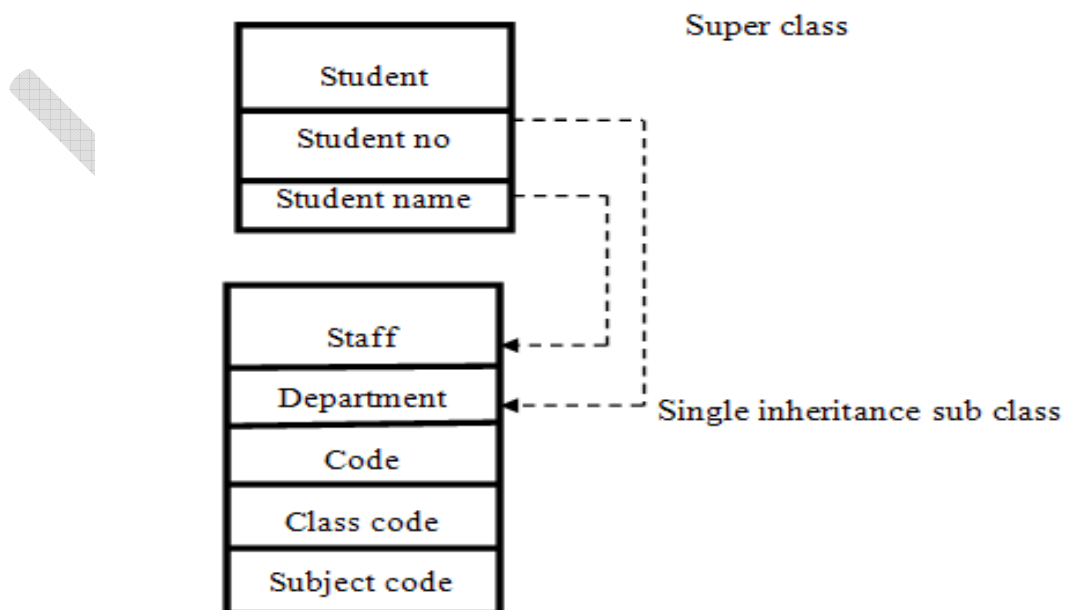
### 1.2 Inheritance in java

Java supports the following types of inheritance. Java uses the **extends** keyword, to set the relationship between a parent class and a child class. The following are the different types of inheritance available in java. They are

- Single inheritance
- Multilevel inheritance
- Hierarchical inheritance

#### i) Single inheritance

A class derived from a super class is called single inheritance. The figure given below shows the single inheritance.



Here a subclass called *staff* is derived from a superclass *student*. The sub class *staff* contains all the properties (student no, student name) of the superclass *employee* plus its own properties (department, code, class code).

### Deriving a sub class

A class derived from an existing class is called subclass. The general form is

```
class name1 extends name2
{
  declaration of variables
  definition of methods
}
```

Where

Class.extends- keyword

Name1 –name of the subclass to be derived

Name2 – name of the class already existing



### Example Program

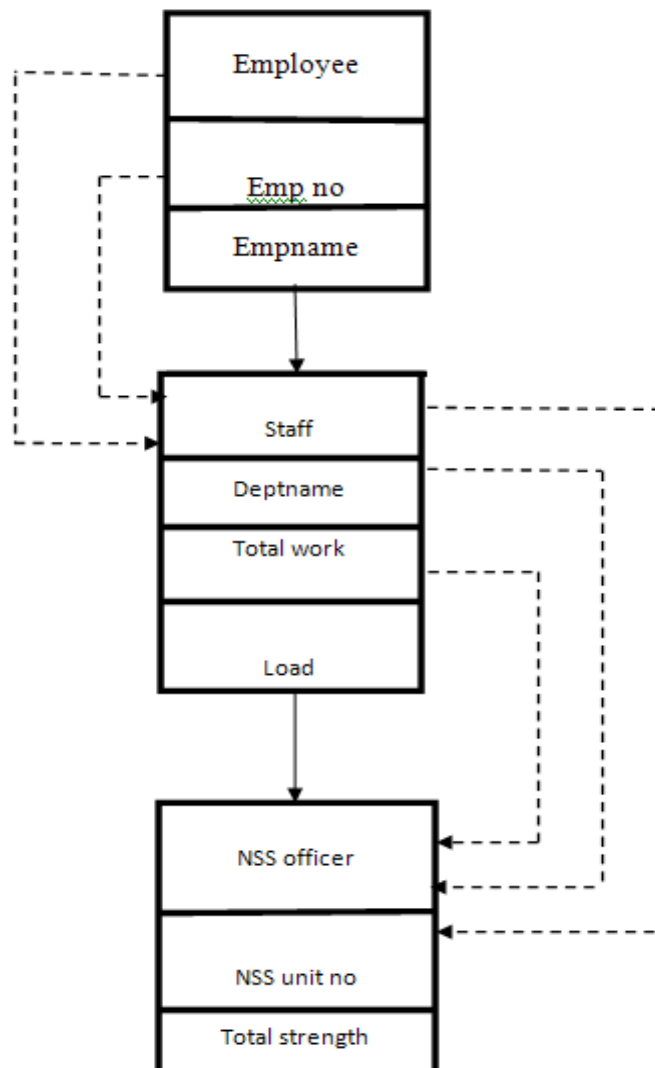
```
class A
{
int a,b;
A()
{
a=10;
b=25;
}
};
class B extends A
{
int c;
void show()
{
c=a+b;
System.out.println(c);
}}
class Const
{
public static void main(String args[])
{
B x1=new B();
x1.show();
}}
```

## Output

```
C:\WINDOWS\system32\cmd.exe  
  
C:\Program Files\Java\jdk1.6.0\bin>javac Const.java  
C:\Program Files\Java\jdk1.6.0\bin>java Const  
35  
C:\Program Files\Java\jdk1.6.0\bin>_
```

## Multilevel inheritance

A class which is derived from other derived class is called multilevel inheritance. The figure given below shows the process of multilevel inheritance.



In the above example a subclass called *staff* is derived from a super class *employee*. This subclass inherits all the properties of the super class *employee*(empno & emp name) plus its own properties( deptname, total work and class). From this subclass another subclass called NSS officer is derived. This class contains all the properties of teacher ( that is properties of employee and teacher) plus its own process(NSS unit no and total strength.)

### Example program

```
class A {
    int i=10,j=20;
    void xyz()
    {
        System.out.println(i+j);
    }
    void lmn()
    {
        System.out.println("lmn");
    }
}
class B extends A{
    int k=40,l=45;
    void abc()      {
        System.out.println(i+j+k);
        lmn();      }}
class C extends B{ }
class Inherit{
    public static void main(String args[])      {
        C a1= new C();
        a1.abc();}}}
```

## Output

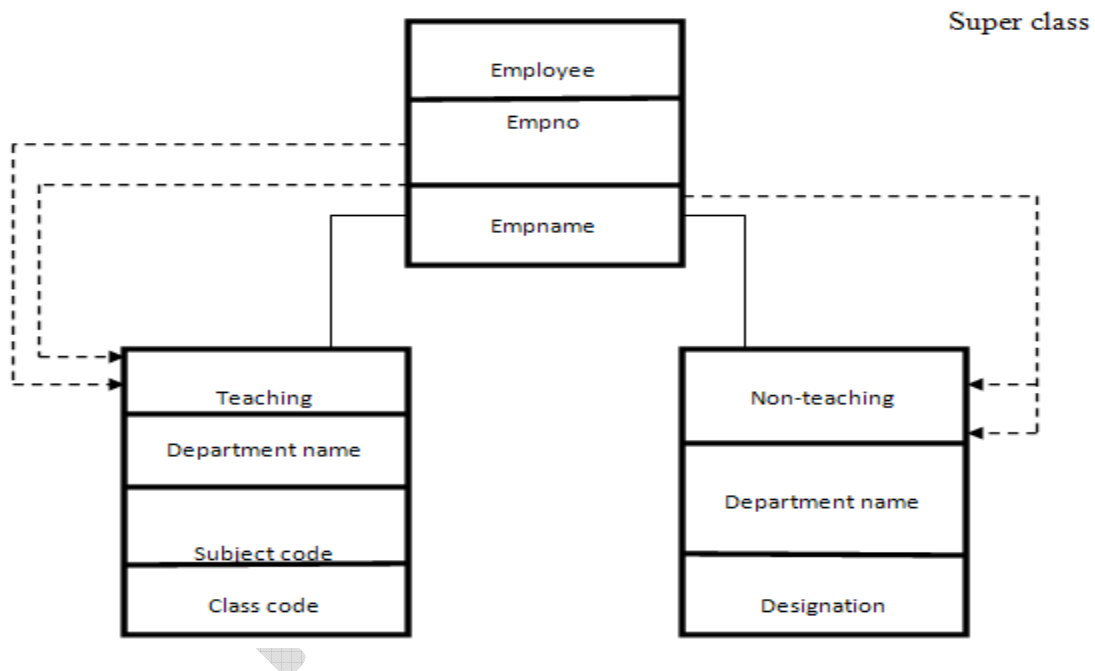
```

C:\WINDOWS\system32\cmd.exe

C:\Program Files\Java\jdk1.6.0\bin>javac Inherit.java
C:\Program Files\Java\jdk1.6.0\bin>java Inherit
70
1mn
C:\Program Files\Java\jdk1.6.0\bin>_
  
```

## Hierarchical Inheritance

More classes derived from one super class is called hierarchical inheritance. The figure given below shows the hierarchical inheritance.



In the above example two sub classes namely teaching and non-teaching are derived from a super class employee. The subclasses contain all the properties of super class employee plus its own properties.

The base class is treated as level1 class. From this class we can derive new classes that are treated as level2 classes. From level2 classes we can derive new classes and are treated as level3 classes and so on.

### 1.3 Casting

The process of converting one data type in to another is called casting. There are two types of conversion. They are

- i) Automatic type conversion
- ii) Explicit type conversion

#### i) Automatic type conversion

If one data type variable is assigned to another data type variable, the lower datatype variable is automatically converted to higher data type before assigning. This process of conversion is called automatic type conversion.

*Eg:*

```
float a;  
int b=20;  
a=b;
```

Here the value of b is automatically converted to float and it is assigned to a. Therefore a=20.0

#### ii) Explicit type conversion

The process of converting one data type variable in to another locally, without affecting its data type in the declaration is called explicit conversion. The general form is

```
(datatype)variable;
```

Where

Datatype - valid data type

Variable - valid user defined name

Eg:

```
x=(int)5.25;  
5.25 is converted to 5
```

Here the value of x is converted in to integer.

```
float=a;  
int b=15;  
a=(float)b;
```

#### 1.4 Method Overriding

Overriding is a process of redefining the already defined method in a super class with all its arguments in the subclass.

If an already defined method in a super class is redefined with all arguments in its sub class, then the redefined method overrides the method in the super class. This means the sub class method is active and the super class method is hidden.

If we want to call the method in the super class, we have to follow the syntax given below.

```
super.name of super class method();
```

**Example program**

```
class One {
    int a,b;
    One(int x,int y){
        a=x;
        b=y;}
    void print(){
        System.out.println("a="+a);
        System.out.println("b="+b);
    }
}
class Two extends One{
    int c;
    Two(int x,int y,int z){
        super(x,y);
        c=z;}
    void print(String msg){
        System.out.println(msg);}
    void print(){
        System.out.println("c="+c);}}
class Override {
    public static void main(String args[]){
        Two s1=new Two(10,20,35);
        s1.print("the numbers");
        s1.print();
    }
}
```

**Output**

```

C:\WINDOWS\system32\cmd.exe

C:\Program Files\Java\jdk1.6.0\bin>javac Override.java
C:\Program Files\Java\jdk1.6.0\bin>java Override
the numbers
c=35
C:\Program Files\Java\jdk1.6.0\bin>_

```

### Final variable and methods

Final is a modifier (keyword) used to prevent variables and methods from overriding. If final is added in front of any variable or method this cannot be overridden in the sub-classes. That is the values cannot be changed. The general forms are

- i) final datatype varname;
- ii) final returntype function name()

```

{
.....
}

```

Where

Final -keyword

Datatype, return type-valid datatype such as int, float etc.

varname, function name-valid user defined name



## 1.5 Polymorphism

The word polymorphism means many forms. "poly" means "many". Polymorphism is a technique used to write more than one function with same function name. It is the ability of the object to take many forms. The functions may be in the same class or in different derived classes.

Let us look in to the following example

```
public interface birds{}  
public class parrot{}  
public class pigeon extends parrot implements birds{}
```

Here the pigeon is said to be polymorphic

When the reference variable is applied to pigeon the declarations can be done as follows,

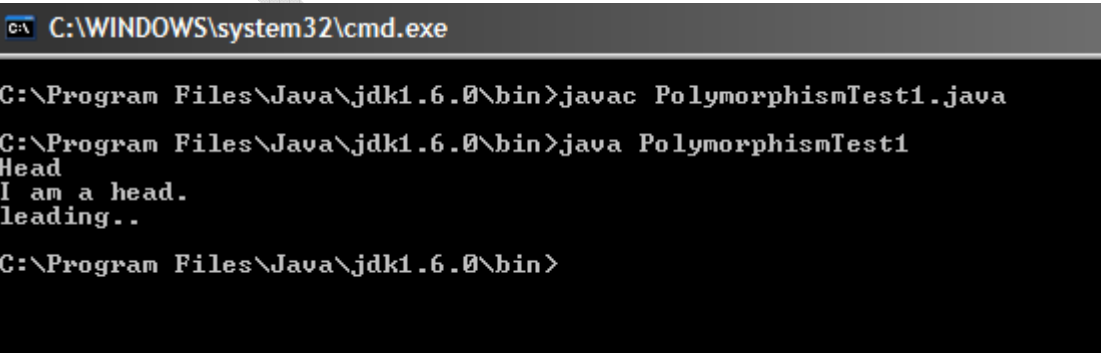
```
pigeon d = new pigeon();  
parrot a = d;  
birds v = d;  
Object o = d;
```

**Example program**

```
class Labour {
    public void work() {
        System.out.println("I am labour");
    }
}
class Head extends Labour {
    public void work() {
        System.out.println("I am a head.");
    }

    public void manage() {
        System.out.println("leading..");
    }
}

public class PolymorphismTest1 {
    public static void main(String[] args) {
        Labour labour;
        labour = new Head();
        System.out.println(labour.getClass().getName());
        labour.work();
        Head head=(Head) labour;
        head.manage();
    }
}
```

**Output**

```
C:\WINDOWS\system32\cmd.exe
C:\Program Files\Java\jdk1.6.0\bin>javac PolymorphismTest1.java
C:\Program Files\Java\jdk1.6.0\bin>java PolymorphismTest1
Head
I am a head.
leading..
C:\Program Files\Java\jdk1.6.0\bin>
```

## 1.6 Super

While writing programs using inheritance, we normally don't create objects for super class. But we create objects only for the subclasses. This is because all the properties except the constructors of the super class are inherited in the subclass. These can be accessed using the sub class objects. But super class constructor cannot be accessed.

Super is a keyword used to call the super class constructor from sub class constructor. The general form is

```
super(argument list);
```

where,

super- keyword

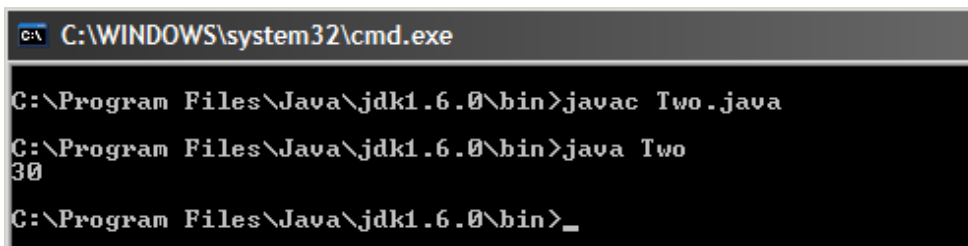
argument list – list of variables to be initialized including superclass variables.

### Rules to be followed

- i) super keyword is used only within a subclass constructor.
- ii) Super() must be the first statement executed inside a subclass constructor
- iii) The arguments in the super() must match the arguments in the constructor defined in the superclass.

**Example Program**

```
class Super1
{
int a,b;
Super1()
{
a=10;
b=20;
}}
class Sub extends Super1
{
int a;
void show()
{
a=super.a+b;
System.out.println(a);
}}
class Two
{
public static void main(String args[])
{
Sub sub1=new Sub();
sub1.show();
}}
```

**Output**

```
C:\WINDOWS\system32\cmd.exe
C:\Program Files\Java\jdk1.6.0\bin>javac Two.java
C:\Program Files\Java\jdk1.6.0\bin>java Two
30
C:\Program Files\Java\jdk1.6.0\bin>_
```

## 1.7 The Object Class

The object class stands at the top of the class hierarchy. Every class in the Java is a descendent of the Object class. The Object class defines the basic state and behavior that all objects must have, such as the ability to compare oneself to another object, to convert to a string, to wait on a condition variable, to notify other objects that a condition variable has changed, and to return the object's class. Every class has Object as a super class.

### The equals method

To compare equality of two strings equals() method is used. It returns true if the method is true & false otherwise.

```
Integer first = new Integer(1), Secondone = new  
Integer(1);  
if (first.equals(Secondone))  
    System.out.println("objects are equal");
```

### The toString method

The toString method returns a string representation of the object. We can display a string representation of a current thread in the following way,

```
System.out.println(Thread.currentThread().toString());
```

The string representation of integer is the integer value displayed as text.

The following are the class methods in java:

equals()    wait()

toString()    hashCode()

clone()    getClass()

notify()    finalize()

getName()

### Exercise

1. Write a program to create a subclass box from the class rectangle.
2. Write a program to create vehicle class. Inherit the classes road vehicle, water vehicle from vehicle class.
3. Write a program to explain overriding method.
4. Write a program to illustrate super keyword
5. Write a program using the following methods
  - a) toString()
  - b) getName()
  - c) wait()

### Summary:

From the above chapter we are familiar with the following concepts

- Inheritance
- Casting methods
- Method Overriding
- Polymorphism
- Object classes in java