# Object and Class in Java

1. [Object in Java](https://www.javatpoint.com/object-and-class-in-java#object)
2. [Class in Java](https://www.javatpoint.com/object-and-class-in-java#class)
3. [Instace Variable in Java](https://www.javatpoint.com/object-and-class-in-java#objectinstancevariable)
4. [Method in Java](https://www.javatpoint.com/object-and-class-in-java#objectmethod)
5. [Example of Object and class that maintains the records of student](https://www.javatpoint.com/object-and-class-in-java#objectex2)
6. [Annonymous Object](https://www.javatpoint.com/object-and-class-in-java#objectannonymous)

In this page, we will learn about java objects and classes. In object-oriented programming technique, we design a program using objects and classes.

Object is the physical as well as logical entity whereas class is the logical entity only.

### Object in Java

An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical (tangible and intangible). The example of intangible object is banking system.

An object has three characteristics:

* state: represents data (value) of an object.
* behavior: represents the behavior (functionality) of an object such as deposit, withdraw etc.
* identity: Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the JVM to identify each object uniquely.

For Example: Pen is an object. Its name is Reynolds, color is white etc. known as its state. It is used to write, so writing is its behavior.

Object is an instance of a class. Class is a template or blueprint from which objects are created. So object is the instance(result) of a class.

Object Definitions:

* Object is a real world entity.
* Object is a run time entity.
* Object is an entity which has state and behavior.
* Object is an instance of a class.

### Class in Java

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

* fields
* methods
* constructors
* blocks
* nested class and interface

### Syntax to declare a class:

1. class <class\_name>{
2. field;
3. method;
4. }

### Instance variable in Java

A variable which is created inside the class but outside the method, is known as instance variable. Instance variable doesn't get memory at compile time. It gets memory at run time when object(instance) is created. That is why, it is known as instance variable.

### Method in Java

In java, a method is like function i.e. used to expose behavior of an object.

#### Advantage of Method

* Code Reusability
* Code Optimization

### new keyword in Java

The new keyword is used to allocate memory at run time. All objects get memory in Heap memory area.

### Object and Class Example: main within class

In this example, we have created a Student class that have two data members id and name. We are creating the object of the Student class by new keyword and printing the objects value.

Here, we are creating main() method inside the class.

File: Student.java

1. class Student{
2. int id;//field or data member or instance variable
3. String name;
4.
5. public static void main(String args[]){
6. Student s1=new Student();//creating an object of Student
7. System.out.println(s1.id);//accessing member through reference variable
8. System.out.println(s1.name);
9. }
10. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=Student)

Output:

0

null

### Object and Class Example: main outside class

In real time development, we create classes and use it from another class. It is a better approach than previous one. Let's see a simple example, where we are having main() method in another class.

We can have multiple classes in different java files or single java file. If you define multiple classes in a single java source file, it is a good idea to save the file name with the class name which has main() method.

File: TestStudent1.java

1. class Student{
2. int id;
3. String name;
4. }
5. class TestStudent1{
6. public static void main(String args[]){
7. Student s1=new Student();
8. System.out.println(s1.id);
9. System.out.println(s1.name);
10. }
11. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestStudent1)

Output:

0

null

## 3 Ways to initialize object

There are 3 ways to initialize object in java.

1. By reference variable
2. By method
3. By constructor

### 1) Object and Class Example: Initialization through reference

Initializing object simply means storing data into object. Let's see a simple example where we are going to initialize object through reference variable.

File: TestStudent2.java

1. class Student{
2. int id;
3. String name;
4. }
5. class TestStudent2{
6. public static void main(String args[]){
7. Student s1=new Student();
8. s1.id=101;
9. s1.name="Sonoo";
10. System.out.println(s1.id+" "+s1.name);//printing members with a white space
11. }
12. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestStudent2)

Output:

101 Sonoo

We can also create multiple objects and store information in it through reference variable.

File: TestStudent3.java

1. class Student{
2. int id;
3. String name;
4. }
5. class TestStudent3{
6. public static void main(String args[]){
7. //Creating objects
8. Student s1=new Student();
9. Student s2=new Student();
10. //Initializing objects
11. s1.id=101;
12. s1.name="Sonoo";
13. s2.id=102;
14. s2.name="Amit";
15. //Printing data
16. System.out.println(s1.id+" "+s1.name);
17. System.out.println(s2.id+" "+s2.name);
18. }
19. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestStudent3)

Output:

101 Sonoo

102 Amit

### 2) Object and Class Example: Initialization through method

In this example, we are creating the two objects of Student class and initializing the value to these objects by invoking the insertRecord method. Here, we are displaying the state (data) of the objects by invoking the displayInformation() method.

File: TestStudent4.java

1. class Student{
2. int rollno;
3. String name;
4. void insertRecord(int r, String n){
5. rollno=r;
6. name=n;
7. }
8. void displayInformation(){System.out.println(rollno+" "+name);}
9. }
10. class TestStudent4{
11. public static void main(String args[]){
12. Student s1=new Student();
13. Student s2=new Student();
14. s1.insertRecord(111,"Karan");
15. s2.insertRecord(222,"Aryan");
16. s1.displayInformation();
17. s2.displayInformation();
18. }
19. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestStudent4)

Output:

111 Karan

222 Aryan



As you can see in the above figure, object gets the memory in heap memory area. The reference variable refers to the object allocated in the heap memory area. Here, s1 and s2 both are reference variables that refer to the objects allocated in memory.

### 3) Object and Class Example: Initialization through constructor

We will learn about constructors in java later.

### Object and Class Example: Employee

Let's see an example where we are maintaining records of employees.

File: TestEmployee.java

1. class Employee{
2. int id;
3. String name;
4. float salary;
5. void insert(int i, String n, float s) {
6. id=i;
7. name=n;
8. salary=s;
9. }
10. void display(){System.out.println(id+" "+name+" "+salary);}
11. }
12. public class TestEmployee {
13. public static void main(String[] args) {
14. Employee e1=new Employee();
15. Employee e2=new Employee();
16. Employee e3=new Employee();
17. e1.insert(101,"ajeet",45000);
18. e2.insert(102,"irfan",25000);
19. e3.insert(103,"nakul",55000);
20. e1.display();
21. e2.display();
22. e3.display();
23. }
24. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestEmployee)

Output:

101 ajeet 45000.0

102 irfan 25000.0

103 nakul 55000.0

### Object and Class Example: Rectangle

There is given another example that maintains the records of Rectangle class.

File: TestRectangle1.java

1. class Rectangle{
2. int length;
3. int width;
4. void insert(int l, int w){
5. length=l;
6. width=w;
7. }
8. void calculateArea(){System.out.println(length\*width);}
9. }
10. class TestRectangle1{
11. public static void main(String args[]){
12. Rectangle r1=new Rectangle();
13. Rectangle r2=new Rectangle();
14. r1.insert(11,5);
15. r2.insert(3,15);
16. r1.calculateArea();
17. r2.calculateArea();
18. }
19. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestRectangle1)

Output:

55

45

## What are the different ways to create an object in Java?

There are many ways to create an object in java. They are:

* By new keyword
* By newInstance() method
* By clone() method
* By deserialization
* By factory method etc.

We will learn these ways to create object later.

## Anonymous object

Anonymous simply means nameless. An object which has no reference is known as anonymous object. It can be used at the time of object creation only.

If you have to use an object only once, anonymous object is a good approach. For example:

1. new Calculation();//anonymous object

Calling method through reference:

1. Calculation c=new Calculation();
2. c.fact(5);

Calling method through anonymous object

1. new Calculation().fact(5);

Let's see the full example of anonymous object in java.

1. class Calculation{
2. void fact(int  n){
3. int fact=1;
4. for(int i=1;i<=n;i++){
5. fact=fact\*i;
6. }
7. System.out.println("factorial is "+fact);
8. }
9. public static void main(String args[]){
10. new Calculation().fact(5);//calling method with anonymous object
11. }
12. }

Output:

Factorial is 120

### Creating multiple objects by one type only

We can create multiple objects by one type only as we do in case of primitives.

Initialization of primitive variables:

1. int a=10, b=20;

Initialization of refernce variables:

1. Rectangle r1=new Rectangle(), r2=new Rectangle();//creating two objects

Let's see the example:

1. class Rectangle{
2. int length;
3. int width;
4. void insert(int l,int w){
5. length=l;
6. width=w;
7. }
8. void calculateArea(){System.out.println(length\*width);}
9. }
10. class TestRectangle2{
11. public static void main(String args[]){
12. Rectangle r1=new Rectangle(),r2=new Rectangle();//creating two objects
13. r1.insert(11,5);
14. r2.insert(3,15);
15. r1.calculateArea();
16. r2.calculateArea();
17. }
18. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestRectangle2)

Output:

55

45

### Real World Example: Account

File: TestAccount.java

1. class Account{
2. int acc\_no;
3. String name;
4. float amount;
5. void insert(int a,String n,float amt){
6. acc\_no=a;
7. name=n;
8. amount=amt;
9. }
10. void deposit(float amt){
11. amount=amount+amt;
12. System.out.println(amt+" deposited");
13. }
14. void withdraw(float amt){
15. if(amount<amt){
16. System.out.println("Insufficient Balance");
17. }else{
18. amount=amount-amt;
19. System.out.println(amt+" withdrawn");
20. }
21. }
22. void checkBalance(){System.out.println("Balance is: "+amount);}
23. void display(){System.out.println(acc\_no+" "+name+" "+amount);}
24. }
25.
26. class TestAccount{
27. public static void main(String[] args){
28. Account a1=new Account();
29. a1.insert(832345,"Ankit",1000);
30. a1.display();
31. a1.checkBalance();
32. a1.deposit(40000);
33. a1.checkBalance();
34. a1.withdraw(15000);
35. a1.checkBalance();
36. }}

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestAccount)

Output:

832345 Ankit 1000.0

Balance is: 1000.0

40000.0 deposited

Balance is: 41000.0

15000.0 withdrawn

Balance is: 26000.0

|  |  |  |
| --- | --- | --- |
| Constructor in Java1. [Types of constructors](https://www.javatpoint.com/constructor#constypes)
	1. [Default Constructor](https://www.javatpoint.com/constructor#consdef)
	2. [Parameterized Constructor](https://www.javatpoint.com/constructor#conspara)
2. [Constructor Overloading](https://www.javatpoint.com/constructor#consoverloading)
3. [Does constructor return any value](https://www.javatpoint.com/constructor#consdoesreturn)
4. [Copying the values of one object into another](https://www.javatpoint.com/constructor#conscopy)
5. [Does constructor perform other task instead initialization](https://www.javatpoint.com/constructor#consothertask)

In Java, constructor is a block of codes similar to method. It is called when an instance of object is created and memory is allocated for the object. It is a special type of method which is used to initialize the object. Note: It is called constructor because it constructs the values at the time of object creation. It is not necessary to write a constructor for a class. It is because java compiler creates a default constructor if your class doesn't have any. Rules for creating java constructorThere are basically two rules defined for the constructor.1. Constructor name must be same as its class name
2. Constructor must have no explicit return type

Types of java constructorsThere are two types of constructors in java:1. Default constructor (no-arg constructor)
2. Parameterized constructor

 Java Default ConstructorA constructor is called "Default Constructor" when it doesn't have any parameter. Syntax of default constructor:1. <class\_name>(){}

Example of default constructor

|  |
| --- |
| In this example, we are creating the no-arg constructor in the Bike class. It will be invoked at the time of object creation.  |

1. class Bike1{
2. Bike1(){System.out.println("Bike is created");}
3. public static void main(String args[]){
4. Bike1 b=new Bike1();
5. }
6. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=Bike1) Output:Bike is createdRule: If there is no constructor in a class, compiler automatically creates a default constructor. Q) What is the purpose of default constructor?Default constructor is used to provide the default values to the object like 0, null etc. depending on the type. Example of default constructor that displays the default values1. class Student3{
2. int id;
3. String name;
4.
5. void display(){System.out.println(id+" "+name);}
6.
7. public static void main(String args[]){
8. Student3 s1=new Student3();
9. Student3 s2=new Student3();
10. s1.display();
11. s2.display();
12. }
13. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=Student3) Output:0 null0 null**Explanation:**In the above class,you are not creating any constructor so compiler provides you a default constructor.Here 0 and null values are provided by default constructor. Java parameterized constructorA constructor which has a specific number of parameters is called parameterized constructor. Why use parameterized constructor?

|  |
| --- |
| Parameterized constructor is used to provide different values to the distinct objects.  |

Example of parameterized constructorIn this example, we have created the constructor of Student class that have two parameters. We can have any number of parameters in the constructor.  |

1. class Student4{
2. int id;
3. String name;
4.
5. Student4(int i,String n){
6. id = i;
7. name = n;
8. }
9. void display(){System.out.println(id+" "+name);}
10.
11. public static void main(String args[]){
12. Student4 s1 = new Student4(111,"Karan");
13. Student4 s2 = new Student4(222,"Aryan");
14. s1.display();
15. s2.display();
16. }
17. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=Student4)

Output:

111 Karan

222 Aryan

## Constructor Overloading in Java

In Java, a constructor is just like a method but without return type. It can also be overloaded like Java methods.

Constructor overloading in Java is a technique of having more than one constructor with different parameter lists. They are arranged in a way that each constructor performs a different task. They are differentiated by the compiler by the number of parameters in the list and their types.

### Example of Constructor Overloading

1. class Student5{
2. int id;
3. String name;
4. int age;
5. Student5(int i,String n){
6. id = i;
7. name = n;
8. }
9. Student5(int i,String n,int a){
10. id = i;
11. name = n;
12. age=a;
13. }
14. void display(){System.out.println(id+" "+name+" "+age);}
15.
16. public static void main(String args[]){
17. Student5 s1 = new Student5(111,"Karan");
18. Student5 s2 = new Student5(222,"Aryan",25);
19. s1.display();
20. s2.display();
21. }
22. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=Student5)

Output:

111 Karan 0

222 Aryan 25

## Difference between constructor and method in java

There are many differences between constructors and methods. They are given below.

|  |  |
| --- | --- |
| Java Constructor | Java Method |
| Constructor is used to initialize the state of an object. | Method is used to expose behaviour of an object. |
| Constructor must not have return type. | Method must have return type. |
| Constructor is invoked implicitly. | Method is invoked explicitly. |
| The java compiler provides a default constructor if you don't have any constructor. | Method is not provided by compiler in any case. |
| Constructor name must be same as the class name. | Method name may or may not be same as class name. |

## Java Copy Constructor

There is no copy constructor in java. But, we can copy the values of one object to another like copy constructor in C++.

There are many ways to copy the values of one object into another in java. They are:

* By constructor
* By assigning the values of one object into another
* By clone() method of Object class

In this example, we are going to copy the values of one object into another using java constructor.

1. class Student6{
2. int id;
3. String name;
4. Student6(int i,String n){
5. id = i;
6. name = n;
7. }
8.
9. Student6(Student6 s){
10. id = s.id;
11. name =s.name;
12. }
13. void display(){System.out.println(id+" "+name);}
14.
15. public static void main(String args[]){
16. Student6 s1 = new Student6(111,"Karan");
17. Student6 s2 = new Student6(s1);
18. s1.display();
19. s2.display();
20. }
21. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=Student6)

Output:

111 Karan

111 Karan

## Copying values without constructor

We can copy the values of one object into another by assigning the objects values to another object. In this case, there is no need to create the constructor.

1. class Student7{
2. int id;
3. String name;
4. Student7(int i,String n){
5. id = i;
6. name = n;
7. }
8. Student7(){}
9. void display(){System.out.println(id+" "+name);}
10.
11. public static void main(String args[]){
12. Student7 s1 = new Student7(111,"Karan");
13. Student7 s2 = new Student7();
14. s2.id=s1.id;
15. s2.name=s1.name;
16. s1.display();
17. s2.display();
18. }
19. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=Student7)

Output:

111 Karan

111 Karan

### Q) Does constructor return any value?

Ans:yes, that is current class instance (You cannot use return type yet it returns a value).

### Can constructor perform other tasks instead of initialization?

Yes, like object creation, starting a thread, calling method etc. You can perform any operation in the constructor as you perform in the method.

# Method Overloading in Java

1. [Different ways to overload the method](https://www.javatpoint.com/method-overloading-in-java#monumberofways)
2. [By changing the no. of arguments](https://www.javatpoint.com/method-overloading-in-java#mobynumber)
3. [By changing the datatype](https://www.javatpoint.com/method-overloading-in-java#mobydatatype)
4. [Why method overloading is not possible by changing the return type](https://www.javatpoint.com/method-overloading-in-java#moreturntype)
5. [Can we overload the main method](https://www.javatpoint.com/method-overloading-in-java#momainmethod)
6. [method overloading with Type Promotion](https://www.javatpoint.com/method-overloading-in-java#motypepromotion)

If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

If we have to perform only one operation, having same name of the methods increases the readability of the program.

Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the method such as a(int,int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you as well as other programmers to understand the behavior of the method because its name differs.

So, we perform method overloading to figure out the program quickly.

## Advantage of method overloading

Method overloading increases the readability of the program.

### Different ways to overload the method

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

#### In java, Method Overloading is not possible by changing the return type of the method only.

### 1) Method Overloading: changing no. of arguments

In this example, we have created two methods, first add() method performs addition of two numbers and second add method performs addition of three numbers.

In this example, we are creating static methods so that we don't need to create instance for calling methods.

1. class Adder{
2. static int add(int a,int b){return a+b;}
3. static int add(int a,int b,int c){return a+b+c;}
4. }
5. class TestOverloading1{
6. public static void main(String[] args){
7. System.out.println(Adder.add(11,11));
8. System.out.println(Adder.add(11,11,11));
9. }}

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestOverloading1)

Output:

22

33

### 2) Method Overloading: changing data type of arguments

In this example, we have created two methods that differs in data type. The first add method receives two integer arguments and second add method receives two double arguments.

1. class Adder{
2. static int add(int a, int b){return a+b;}
3. static double add(double a, double b){return a+b;}
4. }
5. class TestOverloading2{
6. public static void main(String[] args){
7. System.out.println(Adder.add(11,11));
8. System.out.println(Adder.add(12.3,12.6));
9. }}

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestOverloading2)

Output:

22

24.9

### Q) Why Method Overloading is not possible by changing the return type of method only?

In java, method overloading is not possible by changing the return type of the method only because of ambiguity. Let's see how ambiguity may occur:

1. class Adder{
2. static int add(int a,int b){return a+b;}
3. static double add(int a,int b){return a+b;}
4. }
5. class TestOverloading3{
6. public static void main(String[] args){
7. System.out.println(Adder.add(11,11));//ambiguity
8. }}

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestOverloading3)

Output:

Compile Time Error: method add(int,int) is already defined in class Adder

System.out.println(Adder.add(11,11)); //Here, how can java determine which sum() method should be called?

#### Note: Compile Time Error is better than Run Time Error. So, java compiler renders compiler time error if you declare the same method having same parameters.

### Can we overload java main() method?

Yes, by method overloading. You can have any number of main methods in a class by method overloading. But JVM calls main() method which receives string array as arguments only. Let's see the simple example:

1. class TestOverloading4{
2. public static void main(String[] args){System.out.println("main with String[]");}
3. public static void main(String args){System.out.println("main with String");}
4. public static void main(){System.out.println("main without args");}
5. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestOverloading4)

Output:

main with String[]

## Method Overloading and Type Promotion

One type is promoted to another implicitly if no matching datatype is found. Let's understand the concept by the figure given below:



As displayed in the above diagram, byte can be promoted to short, int, long, float or double. The short datatype can be promoted to int,long,float or double. The char datatype can be promoted to int,long,float or double and so on.

### Example of Method Overloading with TypePromotion

1. class OverloadingCalculation1{
2. void sum(int a,long b){System.out.println(a+b);}
3. void sum(int a,int b,int c){System.out.println(a+b+c);}
4.
5. public static void main(String args[]){
6. OverloadingCalculation1 obj=new OverloadingCalculation1();
7. obj.sum(20,20);//now second int literal will be promoted to long
8. obj.sum(20,20,20);
9.
10. }
11. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=OverloadingCalculation1)

Output:40

 60

### Example of Method Overloading with Type Promotion if matching found

If there are matching type arguments in the method, type promotion is not performed.

1. class OverloadingCalculation2{
2. void sum(int a,int b){System.out.println("int arg method invoked");}
3. void sum(long a,long b){System.out.println("long arg method invoked");}
4.
5. public static void main(String args[]){
6. OverloadingCalculation2 obj=new OverloadingCalculation2();
7. obj.sum(20,20);//now int arg sum() method gets invoked
8. }
9. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=OverloadingCalculation2)

Output:int arg method invoked

### Example of Method Overloading with Type Promotion in case of ambiguity

If there are no matching type arguments in the method, and each method promotes similar number of arguments, there will be ambiguity.

1. class OverloadingCalculation3{
2. void sum(int a,long b){System.out.println("a method invoked");}
3. void sum(long a,int b){System.out.println("b method invoked");}
4.
5. public static void main(String args[]){
6. OverloadingCalculation3 obj=new OverloadingCalculation3();
7. obj.sum(20,20);//now ambiguity
8. }
9. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=OverloadingCalculation3)

Output:Compile Time Error

# Inheritance in Java

1. [Inheritance](https://www.javatpoint.com/inheritance-in-java)
2. [Types of Inheritance](https://www.javatpoint.com/inheritance-in-java#inheritancetypes)
3. [Why multiple inheritance is not possible in java in case of class?](https://www.javatpoint.com/inheritance-in-java#inheritancenotmultiple)

Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object.

The idea behind inheritance in java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also.

Inheritance represents the IS-A relationship, also known as parent-child relationship.

### Why use inheritance in java

* For Method Overriding (so runtime polymorphism can be achieved).
* For Code Reusability.

### Syntax of Java Inheritance

1. class Subclass-name extends Superclass-name
2. {
3. //methods and fields
4. }

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called parent or super class and the new class is called child or subclass.

### Java Inheritance Example



As displayed in the above figure, Programmer is the subclass and Employee is the superclass. Relationship between two classes is **Programmer IS-A Employee**.It means that Programmer is a type of Employee.

1. class Employee{
2. float salary=40000;
3. }
4. class Programmer extends Employee{
5. int bonus=10000;
6. public static void main(String args[]){
7. Programmer p=new Programmer();
8. System.out.println("Programmer salary is:"+p.salary);
9. System.out.println("Bonus of Programmer is:"+p.bonus);
10. }
11. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=Programmer)

 Programmer salary is:40000.0

 Bonus of programmer is:10000

In the above example, Programmer object can access the field of own class as well as of Employee class i.e. code reusability.

## Types of inheritance in java

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.



#### Note: Multiple inheritance is not supported in java through class.

When a class extends multiple classes i.e. known as multiple inheritance. For Example:



## Single Inheritance Example

File: TestInheritance.java

1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
5. void bark(){System.out.println("barking...");}
6. }
7. class TestInheritance{
8. public static void main(String args[]){
9. Dog d=new Dog();
10. d.bark();
11. d.eat();
12. }}

Output:

barking...

eating...

## Multilevel Inheritance Example

File: TestInheritance2.java

1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
5. void bark(){System.out.println("barking...");}
6. }
7. class BabyDog extends Dog{
8. void weep(){System.out.println("weeping...");}
9. }
10. class TestInheritance2{
11. public static void main(String args[]){
12. BabyDog d=new BabyDog();
13. d.weep();
14. d.bark();
15. d.eat();
16. }}

Output:

weeping...

barking...

eating...

## Hierarchical Inheritance Example

File: TestInheritance3.java

1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
5. void bark(){System.out.println("barking...");}
6. }
7. class Cat extends Animal{
8. void meow(){System.out.println("meowing...");}
9. }
10. class TestInheritance3{
11. public static void main(String args[]){
12. Cat c=new Cat();
13. c.meow();
14. c.eat();
15. //c.bark();//C.T.Error
16. }}

Output:

meowing...

eating...

## Q) Why multiple inheritance is not supported in java?

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B and C are three classes. The C class inherits A and B classes. If A and B classes have same method and you call it from child class object, there will be ambiguity to call method of A or B class.

Since compile time errors are better than runtime errors, java renders compile time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error now.

1. class A{
2. void msg(){System.out.println("Hello");}
3. }
4. class B{
5. void msg(){System.out.println("Welcome");}
6. }
7. class C extends A,B{//suppose if it were
8.
9. Public Static void main(String args[]){
10. C obj=new C();
11. obj.msg();//Now which msg() method would be invoked?
12. }
13. }

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=C)

 Compile Time Error

|  |
| --- |
|  |

##