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. {
3. field;
4. method;
5. }

### 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;
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. }
11. Output:
12. 0
13. 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. }

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. }
20. Output:
21. 101 Sonoo
22. 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. }
20. Output:
21. 111 Karan
22. 222 Aryan

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. }
25. Output:
26. 101 ajeet 45000.0
27. 102 irfan 25000.0
28. 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. }

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. }
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. }}
37. Output:
38. 832345 Ankit 1000.0
39. Balance is: 1000.0
40. 40000.0 deposited
41. Balance is: 41000.0
42. 15000.0 withdrawn
43. Balance is: 26000.0

# this keyword in java

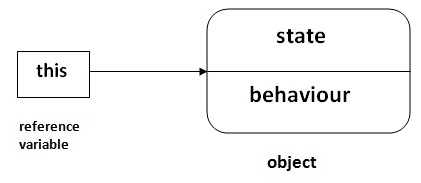
There can be a lot of usage of **java this keyword**. In java, this is a **reference variable** that refers to the current object.

## **Usage of java this keyword**

Here is given the 6 usage of java this keyword.

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this() can be used to invoke current class constructor.
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this can be used to return the current class instance from the method.

**Suggestion:** If you are beginner to java, lookup only three usage of this keyword.



### 1) this: to refer current class instance variable

The this keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.

1. **class** Student{
2. **int** rollno;
3. String name;
4. **float** fee;
5. Student(**int** rollno,String name,**float** fee){
6. **this**.rollno=rollno;
7. **this**.name=name;
8. **this**.fee=fee;
9. }
10. **void** display(){System.out.println(rollno+" "+name+" "+fee);}
11. }
13. **class** TestThis2{
14. **public** **static** **void** main(String args[]){
15. Student s1=**new** Student(111,"ankit",5000f);
16. Student s2=**new** Student(112,"sumit",6000f);
17. s1.display();
18. s2.display();
19. }}

### 2) this: to invoke current class method

You may invoke the method of the current class by using the this keyword. If you don't use the this keyword, compiler automatically adds this keyword while invoking the method.

1. **class** A{
2. **void** m(){System.out.println("hello m");}
3. **void** n(){
4. System.out.println("hello n");
5. //m();//same as this.m()
6. **this**.m();
7. }
8. }
9. **class** TestThis4{
10. **public** **static** **void** main(String args[]){
11. A a=**new** A();
12. a.n();
13. }}

### 3) this() : to invoke current class constructor

The this() constructor call can be used to invoke the current class constructor. It is used to reuse the constructor. In other words, it is used for constructor chaining.

1. **class** A{
2. A(){System.out.println("hello a");}
3. A(**int** x){
4. **this**();
5. System.out.println(x);
6. }
7. }
8. **class** TestThis5{
9. **public** **static** **void** main(String args[]){
10. A a=**new** A(10);
11. }}