Object and Class in Java

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. {
3. **int** id;//field or data member or instance variable
4. String name;
6. **public** **static** **void** main(String args[])
7. {
8. Student s1=**new** Student();//creating an object of Student
9. System.out.println(s1.id);//accessing member through reference variable
10. System.out.println(s1.name);
11. }
12. }

[**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. {
7. **public** **static** **void** main(String args[]){
8. Student s1=**new** Student();
9. System.out.println(s1.id);
10. System.out.println(s1.name);
11. }
12. }

[**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. {
3. **int** rollno; **global var**
4. String name;
5. **void** insertRecord(**int** r, String n)…. **Local var**
6. {
7. rollno=r;
8. name=n;
9. }
10. **void** displayInformation()
11. {
12. System.out.println(rollno+" "+name);
13. }
14. }
15. **class** TestStudent4{
16. **public** **static** **void** main(String args[]){
17. Student s1=**new** Student();
18. Student s2=**new** Student();
19. s1.insertRecord(111,"Karan");
20. s2.insertRecord(222,"Aryan");
21. s1.displayInformation();
22. s2.displayInformation();
23. }
24. }

[**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

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. }
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