



OOP Principles In Java

[become a member](#)[Login](#)

Sandeep Sharma



Updated date, Sep 18, 2019



28.7k



2



0

Post ▼

Ask Question



[Download Free .NET & JAVA Files API](#)

[Try Free File Format APIs for Word/Excel/PDF](#)

Introduction

In this article, we will discuss Object-Oriented Programming (OOP) principles in Java. We will also discuss its features in detail i.e polymorphism, inheritance, encapsulation and abstraction.

OOP Concepts In Java

Object means real-world things such as pen, paper, chair, etc. OOP is a technique that helps in designing a program more

- Class
- Object
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation





The following are the advantages of OOP over procedure-oriented programming languages:

- makes the development of programs easier; in procedure-oriented languages it's difficult to manage the coding of the program when the program increases in size.
- provides a way to interact with real-world data more effectively.
- allows the development of solutions for real-world problems.
- provides hiding of codes but in a procedure-oriented language, the data can be accessed from anywhere.

There are four main features of **OOP**; they are:

1. Encapsulation
2. Inheritance
3. Polymorphism
4. Abstraction

Let's discuss each one in detail.

Encapsulation in Java

Encapsulation means binding all methods and classes in a single class. Encapsulation is the technique of making the fields in a class private and providing access to the fields via public methods. If a field is declared private then it cannot be accessed by anyone outside the class, thereby hiding the fields within the class. The main benefit of encapsulation is the ability to modify our implemented code without breaking the code of others who use our code.

Advantages

The following are a few advantages of using Encapsulation:

1. Provides the ability to change one part of the code without affecting another part of code.
2. Controls the access of the user interface.





5. Encapsulation in Java makes unit testing easy.

6. Reduces the coupling of modules since all pieces of the same type are encapsulated in class.

Post ▼

Ask Question

Inheritance in Java

The main feature of Inheritance is code re-usability. So when making a new class we can use a previously written class and further extend it. In Java, classes may inherit or acquire the properties and methods of other classes. A class derived from another class is called a subclass, whereas the class from which a subclass is derived is called a super class. A subclass can have only one super class, whereas a super class may have one or more sub-classes.

Example

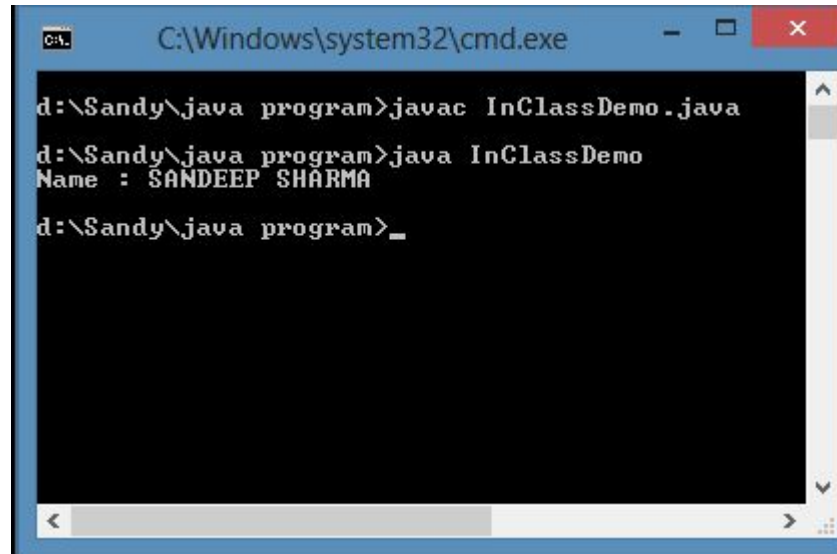
```
01. class StudentRec
02. {
03.     //GET STUDENT RECORD.....
04.     String name;
05.     int rollno;
06.     int get(String n, int r) {
07.         name = n;
08.         rollno = r;
09.         return (0);
10.     }
11.     void showDetails() {
12.         System.out.println("Name : " + name);
13.     }
14. }
```

```
17. {
18.     public static void main(String args[]) {
19.         //CREATE OBJECT OF STUDENT RECORD CLASS
20.         StudentRec studObj = new StudentRec();
21.         studObj.get("SANDEEP SHARMA", 92);
22.         studObj.showDetails();
23.     }
24.     void displayDetails() {
```





Output



```
C:\Windows\system32\cmd.exe

d:\Sandy\java program>javac InClassDemo.java
d:\Sandy\java program>java InClassDemo
Name : SANDEEP SHARMA
d:\Sandy\java program>_
```

Polymorphism in Java

In Core Java, Polymorphism is an easy concept to understand. Polymorphism in Greek is a combination of poly, which means many and morphism which means forms. It refers to the object's ability to be Polymorphic depending on its type.

There are **two** types of **Polymorphism** available in Java.

2. Dynamic Polymorphism

Let's discuss *Static Polymorphism*. It's compile-time Polymorphism. We have two important concepts in Polymorphism, i.e Method Overloading and Method Overriding.

Method Overloading in Java





Example for Method overloading

Post ▼

Ask Question

The following example program will make you understand Method Overloading:

```
01. class Sub {
02.     void add(int tamil, int english) {
03.         System.out.println("The total of tamil and english is " + (tamil + english));
04.     }
05.     void add(int tamil, int english, int maths) {
06.         System.out.println("The total of tamil english and maths is " + (tamil + english + maths));
07.     }
08. }
09.
10. public class MetOvlDemo {
11.     public static void main(String arg[]) {
12.         //create Subjects class object
13.         Sub sb = new Sub();
14.         // we have to call add() method by passing 2 values
15.         sb.add(90, 80);
16.         //here also we are calling add() method by passing 3 values, So the 3 arguments (parameters)
17.         sb.add(95, 85, 100);
18.     }
19. }
```

Output





```
d:\Sandy\java program>javac MetOvlDemo.java
d:\Sandy\java program>java MetOvlDemo
The total of tamil and english is 170
The total of tamil english and maths is 280
d:\Sandy\java program>
```

Method Overriding in Java

Now we will discuss what *dynamic polymorphism* is. It's run time polymorphism. We can also call it Method Overriding. In a class hierarchy, when a method in a sub class has the same name and type signature as a method in its superclass, then the method in the subclass is said to override the method in the superclass. This feature is called method overriding.

Example

```
01. class MathsSqr1 {
02.     void calculate(double price) {
03.         System.out.println("Sqaare value " + (price * price));
04.     }
05. }
06.
07.
08.     void calculate(double price) {
09.         System.out.println("Sqaare value " + (Math.sqrt(price)));
10.     }
11. }
12.
13. public class Metovrr {
14.     public static void main(String arg[]) {
```



```
17.     }  
18. }
```



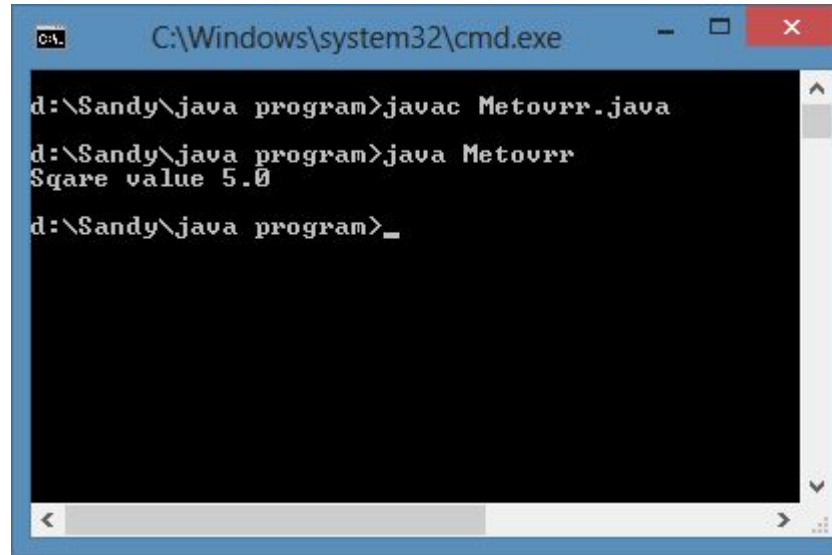
become a member

Login

Post ▼

Ask Question

Output



```
C:\Windows\system32\cmd.exe  
  
d:\Sandy\java program>javac Metovrr.java  
d:\Sandy\java program>java Metovrr  
Square value 5.0  
d:\Sandy\java program>_
```

Abstraction in Java

When we hide the unnecessary detail and defining the useful (relevant) detail, then the procedure is said to be an *abstraction*. An interface or abstract class is something that is not concrete, something that is incomplete. Another way of providing a simple explanation is to use a more complex system as an example. One does not want to understand how an engine works.

Abstraction in Java is done by using an interface and abstract class in Java. In order to use an interface or abstract class, we need to explain the methods of an interface or abstract class in a sub-class.

Abstraction Example: Engine, Driving.



[become a member](#)[Login](#)

Example

[Post ▼](#)[Ask Question](#)

```
01. public class Abstraction {  
02.     private int accNO;  
03.     private String custName;  
04.     private float accBlnc;  
05.     private float profit;  
06.     private float loan;  
07.     public void displayClerkInfo() {  
08.         System.out.println("Account number " + accNo);  
09.         System.out.println("Customer name " + custName);  
10.         System.out.println("Account Balance " + accBlnc);  
11.     }  
12. }
```

Note: This class only defines the structure but not any implementation of them.

Next Recommended Article

[Difference Between An Error And Exception in Java](#)

[encapsulation in java](#)[inheritance in java](#)[oops in java](#)[polymorphism in java](#)

Sandeep Sharma *TOP 500*

Working as a Senior Engineering Analyst having work experience in Core Java, Struts, Spring, Hibernate, Maven, JSP, AJAX, JQ
JSON, Webservice, Jasper-Report, Oracle, MySQL, SQL-Server-2012.



[become a member](#)[Login](#)[Post ▼](#)[Ask Question](#)

Type your comment here and press Enter Key (Minimum 10 characters)



Nice Article

[Ashwani Tyagi](#)

64 24.5k 2.1m

Apr 15, 2013

0 0 Reply



Thank you for explaining this complex topic. I think however that calling some languages procedure-oriented implies that all programs written using an object-oriented language are automatically object-oriented. I think it is possible to write an object-oriented program using a language not designed to be object-oriented and that it is totally possible to write a program that is not object-oriented using an object-oriented language. I think it is critical to understand object-oriented concepts such as described here.

[Sam Hobbs](#)

46 29.5k 1.6m

Apr 15, 2013

0 0 Reply





C# Corner

Join Public Speaking Virtual Conference



Become a member

Login

Post ▼

Ask Question

FEATURED ARTICLES

SQL Query Execution Plan Operations

Insert Data Into Azure Table Storage Using ASP.NET Core Application





C# Corner

Join Public Speaking Virtual Conference

SQL Query Execution Plan

How To Export Data In EXCEL, PDF, CSV, Word, JSON, XML And Text File In MVC Application



Become a member

Login

Post ▼

Ask Question

View All ➔

TRENDING UP



01 Agile Or Scrum - Which One To Choose And Why?

02 Introduction to Network Virtualization

03 What is Docker?

05 Build A FAQ Chatbot With Power Virtual Agents

06 Facade Design Pattern

07 Insert Data Into Azure Table Storage Using ASP.NET Core Application

08 Search Data Between Two Dates Using Web API And Angular 9





C# Corner

Join Public Speaking Virtual Conference

10 Memory Cache in C#



become a member

Login

Post ▼

Ask Question

View All ➔



**C# Corner**

Join Public Speaking Virtual Conference

[About Us](#) [Contact Us](#) [Privacy Policy](#) [Terms](#) [Media Kit](#) [Sitemap](#) [Report a Bug](#)[become a member](#)[Login](#)[C# Tutorials](#) [Common Interview Questions](#) [Stories](#) [Consultants](#) [Ideas](#) [C# Projects](#)[Post](#) ▼[Ask Question](#)

©2020 C# Corner. All contents are copyright of their authors.

