


Introduction to Object-oriented Programming (OOP)

Posted by [potty](#) ^[1] on January 21, 2014 at 1:52 AM CST

 [ide_icon.png](#) ^[2]

Introduction to object-oriented programming concepts: class, objects, polymorphism, abstraction, inheritance.

Why do we need to program?

Technology is part of our life. The world is changing daily and everything is getting automated. Learn to program is the creative way we can take our ideas to the next level and express solutions to society. By designing programs, we learn several abilities like critical reading, analytical thinking and create synthesis. The programmer defines the problem, plans a solution, codes the program, test the proposal and documents the features. But we can't program all the solutions with the same method, that's why programming paradigms appears.

Overview of Programming Paradigms

According to Vasappanavara, a programming paradigm "it is the manner in which programming elements such as functions, objects and variables are exploited to produce the desired output". It is important to understand that programming paradigms are not programming languages.

The following are typical examples of programming paradigms (according to Bhawe):

Name	Features	Building Block	Usage	Examples
1 Imperative	Viewed as a sequence of instructions.	Procedural Abstraction	Scientific Computation	C
2 Functional	Considered as a function from inputs to outputs.	Interaction among Functions	Mathematical Logic	LISP
3 Logic	Attempt to find values that satisfy goals.	Non-deterministic Expressions	Artificial Intelligence	Prolog
4 Object-oriented	Interacting objects of different types.	Object and Classes Modeling	Simulation	Java, C++
5 Event-driven	Continuous loop which responds to events.	Monitoring the Events	Web Applications	Java

What is Object-oriented Programming (OOP)?

The object-oriented is a programming paradigm where the program logic and data are weaved. As stated by Phil Ballard, it is a way of conceptualizing a program's data into discrete "things" referred to as objects, each having its own properties and methods.

Let's see an example. Suppose your friend is a bank manager and he wants you to help

improving their system. The first object you might design is the general-purpose **Account**. The **Account** object has properties and methods. For each client your friend's bank have, you would have to create an **Account** object.

Characteristics

As follows the most important features of object-oriented programming:

- **Encapsulation.** Capability to hide data and instructions inside an object.
- **Inheritance.** Ability to create one object from another.
- **Polymorphism.** The design of new classes is based on a single class.
- **Message Passing.** It is the way objects communicate with each other.
- **Garbage Collection.** Automatic memory management that destroys objects that are no longer in use by the program.

Benefits

As follows some benefits of using object-oriented programming:

- **Re-usability.** You can write a program using a previous developed code.
- **Code Sharing.** You are able to standardize the way your are programming with your colleagues.
- **Rapid Modeling.** You can prototype the classes and their interaction through a diagram.

Drawbacks

As follows the disadvantages of using object-oriented programming:

- **Size.** Are larger than other programs, consuming more memory and disk space.
- **Effort.** Require a lot of work to create, including the diagramming on the planning phase and the coding on the execution phase.

Basic Concepts

Class

Basic template that specifies the properties and behaviours of something (real life or abstract).

Object

Particular instance of a class that responds consequently to events.

Attribute

Characteristics of the class. Often called instance variables.

Method

Algorithm associate to an class that represent a thing that the object does.

Subclass

Class based on another class.

Inheritance

Process where the subclass gets the attributes and methods from its parent class.

Interface

Specific template that enforces certain attributes and methods of a class.

Package

Namespace that organizes a set of related classes and interfaces.

Event

Alert the application when there is a state change of the object.

References

1. Vasappanavara, R. **Object-oriented Programming Using C++ and Java. Chapter 1. Object-oriented Programming Basics. Section 1.3. Programming Paradigms.** Pearson Education. India. May, 2011.
2. Bhawe, M. **Object Oriented Programming with C++, Second Edition. Chapter 4. Object Orientation: An Introduction. Section 4.1. Programming Paradigms.** Pearson Education. India. May, 2012.
3. Ballard, P. **Sams Teach Yourself JavaScript in 24 Hours, Fifth Edition. Hour 7. What is Object Oriented Programming (OOP)?.** Pearson Education, Inc. United States. November, 2012.

Attachment Size

[table_1.png](#) [3] 31.4 KB

Related Topics >> [Java User Groups](#) [4] [Open Source](#) [5] [Java Desktop](#) [6] [J2SE](#) [7]
[Community](#) [8] [Blogs](#) [9]

Your use of this web site or any of its content or software indicates your agreement to be bound by these [Terms of Participation](#).

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.



Powered by Oracle, Project Kenai and Cognisync

Source URL: <http://www.java.net/blog/potty/archive/2014/01/20/introduction-object-oriented-programming-oop-part-i>

Links:

- [1] <http://www.java.net/blog/potty>
- [2] http://www.java.net/sites/default/files/potty/ide_icon.png
- [3] http://www.java.net/sites/default/files/table_1_0.png
- [4] <http://www.java.net/blog-community/java-user-groups>
- [5] <http://www.java.net/related-topics/open-source>
- [6] <http://www.java.net/blog-community/java-desktop>
- [7] <http://www.java.net/topic/j2se>
- [8] <http://www.java.net/topic/community>
- [9] <http://www.java.net/feed-category/blogs>