# Object-Oriented Technologies and Java Programming

Subhash G. Deo

This reference book can be useful for

BBA, MBA, B.Com, BMS, M.Com, BCA, MCA

and many more courses for Various Universities

NEERAJ PUBLICATIONS

www.neerajbooks.com

Published by:

**NEERAJ PUBLICATIONS**

*(Publishers of Educational Books)*

Sales Office : 1507, 1st Floor,
Nai Sarak, Delhi-110 006
E-mail: info@neerajbooks.com
Website: www.neerajbooks.com

Typesetting by: Competent Computers

# CONTENTS

# OBJECT-ORIENTED TECHNOLOGIES AND JAVA PROGRAMMING

***OBJECT-ORIENTED TECHNOLOGY AND JAVA***

## Object-oriented Methodology-1

1

### INTRODUCTION

This chapter gives detailed description of the different programming paradigm evolution and development of different programming languages and it discuss the importance of object-oriented programming methodology. Since the invention of the computer, many approaches of program development have been tested. The program development approach may be categorized as modular programming, top-down programming, bottom-up programming, and structured programming.

In 1980, 'C' language becomes very popular, due to which the structured programming methodology for the development of programs have also becomes popular, but due to undesired performance in terms of maintainability, reusability and reliability, structured programming fails.

After the failures of structured programming, a new programming methodology known as Object-oriented Programming was developed. This approach eliminates some of the pitfalls of conventional programming by incorporating some of the best features of structured programming with some powerful new concepts. This approach speeds up the development of programs and enhance and improves the maintenance, reusability, and modifiability of software.

So, in this chapter, we discuss what are the main features of this approach? How it is better than other approaches? What are the languages which support its various features?

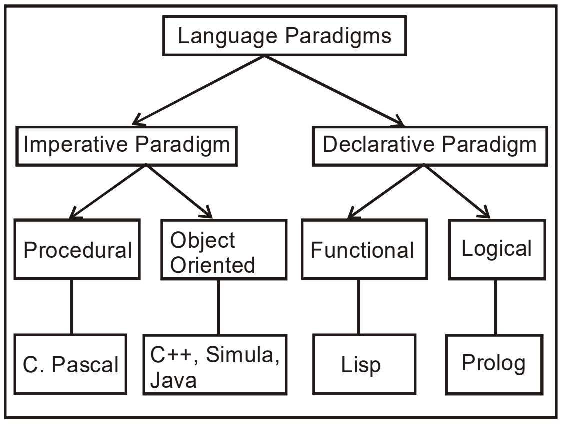
In this chapter, we start with the evolution of different programming languages, and the development of various programming paradigm so as to understand where Object-oriented programming approach fits and why? And, subsequently, we compares the Object-oriented approach with the Procedure-oriented approach. In this chapter, we also introduce some of the basic concepts and terminologies associated with Object-oriented (OO) approach. We also discuss some of the common OO languages and applications of OOP in different problem domain.

### CHAPTER AT A GLANCE

**PARADIGMS OF PROGRAMMING LANGUAGES**

The term paradigm according to [Wegner, 1998] is "patterns of thought for problem solving" or it refers to a set of techniques, methods, theories and standards that together represent a way for solving problem.

The languages paradigms are divided into two parts: *(i)* imperative and *(ii)* declarative paradigms. The imperative languages can be further divided into Procedural and object oriented approach. Declarative languages can be classified into functional languages and logical languages.



Object-oriented programming is one of several programming paradigms. Other programming paradigms include the imperative programming paradigm (as exemplified by languages such as Pascal or C), the logic programming paradigm (Prolog), and the functional programming paradigm (exemplified by languages such as ML, Haskell or Lisp). Logic and functional languages are said to be declarative languages.

We use the word paradigm to mean "any example or model."

This usage of the word was popularized by the science historian Thomas Kuhn. He used the term to describe a set of theories, standards and methods that together represent a way of organizing knowledge, a way of viewing the world.

Thus a programming paradigm is a way of conceptualizing what it means to perform computation and how tasks to be carried out on a computer should be structured and organized.

We can distinguish between two types of programming languages: Imperative languages and declarative languages. Imperative knowledge describes how-to knowledge while declarative knowledge is what-is knowledge?

A program is "declarative" if it describes what something is like, rather than how to create it. This is a different approach from traditional imperative programming languages such as Fortran, and C, which require the programmer to specify an algorithm to be run. In short, imperative programs make the algorithm explicit and leave the goal implicit, while declarative programs make the goal explicit and leave the algorithm implicit.

Imperative languages require you to write down a step-by-step recipe specifying how something is to be done. For example, to calculate the factorial function in an imperative language we would write something like:

```
public int factorial(int n) {
int ans=1;
    for (int i = 2; i <= n; i++){
    ans = ans * i;
    }
    return ans;
}
```

Here, we give a procedure (a set of steps) that when followed will produce the answer.

**Functional Programming**

Functional programming is a programming paradigm that treats computation as the evaluation of mathematical functions. Functional programming emphasizes the definition of functions, in contrast to procedural programming, which emphasizes the execution of sequential commands.

The following is the factorial function written in a functional language called Lisp:

```
(defun factorial (n)
(if (<= n 1) 1 (* n (factorial
                      (- n 1))))
)
```

Notice that it defines the factorial function rather than give the steps to calculate it. The factorial of n is defined as 1 if n <= 1 else it is n * factorial(n - 1)

**Logic Programming**

Prolog (PROgramming in LOGic) is the most widely available language in the logic programming paradigm. It is based on the mathematical ideas of relations and logical inference. Prolog is a declarative language meaning that rather than describing how to compute a solution, a program consists of a data base of facts and logical relationships (rules) which describe the relationships which hold for the given application. Rather than running a program to obtain a solution, the user asks a question. When asked a question, the run time system searches through the data base of facts and rules to determine (by logical deduction) the answer.

Logic programming was an attempt to make a programming language that enabled the expression of logic instead of carefully specified instructions on the computer.

In the logic programming language Prolog you supply a database of facts and rules; you can then perform queries on the database.

This is also an example of a declarative style of programming where we state or define what we know.

In the following example, we declare facts about some domain. We can then query these facts–we can ask, for example, are sally and tom siblings?

```
sibling(X,Y)    :- parent(Z,X), parent(Z,Y).
parent(X,Y)     :- father(X,Y). parent(X,Y) :-
mother(X,Y).  mother(trude, a sally).
father(tom, sally).
father(tom, erica).
father(mike, tom).
```

The factorial function is written in prolog as two rules. Again, notice the declarative nature of the program.

```
fac(0,1).
fac(N,F):-     N > 0,
               M is N - 1,
               fac(M,Fm), F
               is N * Fm.
```

To summarize:

- In procedural languages, everything is a procedure.
- In functional languages, everything is a function.
- In logic programming languages, everything is a logical expression (predicate).
- In object-oriented languages, everything is an object.

## EVOLUTION OF OO METHODOLOGY

When computer system was invented, programming was done using machine language i.e. 0 and 1. Then, to provide convenience to the programmer, assembly language was developed which uses **mnemonic** to **describe** instructions to write programs. But, it was difficult to remember too many mnemonic codes to program in it. Another disadvantage with **assembly language** is that they are machine dependent.

To overcome the difficulties of assembly language, **high-level languages** were developed. High-level programming offers much easy in writing user programs and easy to understand than machine or assembly languages. High-level languages provide capacity and capability to write more complex programs easily. The disadvantage of high-level language is that they have limitations in reusability, flow control, difficulty due to global variables, understanding and maintainability of long programs.

### Structured Programming

In structured programming, the complexity of programming is handled by dividing it into small unit called functions (subroutines, procedures, subprogram) to make program more comprehensible. Each function has a clearly defined purpose and defined interface to the other functions in the program. The number of functions are grouped together into a larger entity called a module, but, the principle remains the same.

Structured programming helps the programmer in writing an **error free** code and **maintain control** over each function. This makes the development and maintainance of the code **faster** and **efficient**.

### Object-oriented Programming

**Object-oriented Programming:** (Oop) represents an attempt to make programs more closely model the way people think about and deal with the world. In the older styles of programming, a programmer who is faced with some problem must identify a computing task that needs to be performed in order to solve the problem. Programming then consists of finding a sequence of instructions that will accomplish that task. But at the heart of object-oriented programming, instead of tasks we find objects entities that have behaviours, that hold information, and that can interact with one another. Programming consists of designing a set of objects that model the problem at hand. Software objects in the program can represent real or abstract entities in the problem domain. This is supposed to make the design of the program more natural and hence easier to get right and easier to understand.

**Important:** An object-oriented programming language such as JAVA includes a number of features that make it very different from a standard language. In order to make effective use of those features, you have to "orient" your thinking correctly.

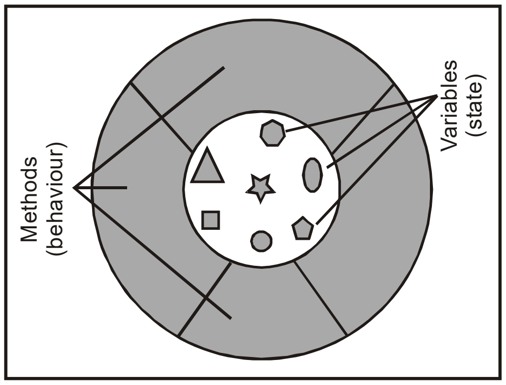
## BASIC CONCEPTS OF OO APPROACH

It is claimed that the problem solving techniques used in object-oriented programming more closely models the way humans solve day-to-day problems.

Object-oriented techniques are popular because experts agree that object oriented techniques are more disciplined than structured programming techniques [Martin and Odell 1992].

The main components of object-oriented technology are Objects and Classes, data abstraction and encapsulation, inheritance, and polymorphism. It is important to understand these concepts to use OOP method.

**Objects**



In object-oriented programming we create software objects that model real world objects. Software objects are modeed after real-world objects in that they too have state and behaviour. A software object maintains its state in one or more variables. A variable is an item of data named by an identifier. A software object implements its behaviour with methods. A method is a function associated with an object.

**Definition:** An object is a software bundle of variables and related methods.

An object is also known as an instance. An instance refers to a particular object. e.g. Karuna's bicycle is an instance of a bicycle, because, it refers to a particular bicycle. Sandile Zuma is an instance of a Student.

The variables of an object are formally known as instance variables because they contain the state for a particular object or instance. In a running program, there may be many instances of an object, e.g. there may be many Student objects. Each of these objects will have their own instance variables and each object may have different values stored in their instance variables. e.g. each Student object will have a different number stored in its Student Number variable.

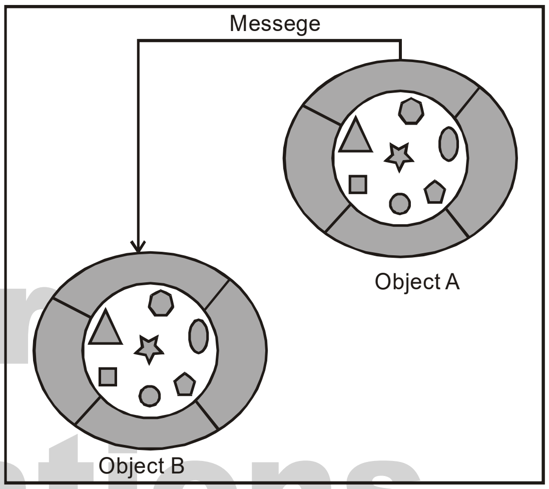
**Encapsulation**

Object diagrams show that an object's variables make up the center, or nucleus, of the object. Methods surround and hide the object's nucleus from other objects in the program. Packaging an object's variables within the protective custody of its methods is called encapsulation.

Encapsulating related variables and methods into a neat software bundle is a simple yet powerful idea that provides two benefits to software developers:

- **Modularity:** The source code for an object can be written and maintained independently of the source code for other objects. Also, an object can be easily passed around in the system. You can give your bicycle to someone else, and it will still work.
- **Information-hiding:** An object has a public interface that other objects can use to communicate with it. The object can maintain private information and methods that can be changed at any time without affecting other objects that depend on it.

**Messages**



Software objects interact and communicate with each other by sending messages to each other. When object A wants object B to perform one of B's methods, object A sends a message to object B.

**There are three parts of a message:** The three parts for the message System.out.println{"Hello World"}; are:

- The object to which the message is addressed (System.out)
- The name of the method to perform (println)
- Any parameters needed by the method ("Hello World!")

**Classes**

In object-oriented software, it's possible to have many objects of the same kind that share characteristics: rectangles, employee records, video clips, and so on. A class is a software blueprint for objects. A class is used to manufacture or create objects.

The class declares the instance variables necessary to contain the state of every object. The class would also declare and provide implementations for the