NEP NEW SYLLABUS

A Book Of

Java Programming

For MCA (Management) : Semester - II

[Course Code IT-21 : Credit - 03]

As per New syllabus of NEP-2020

w.e.f. Academic Year 2024-25

Mr. Satyavan M. Kunjir
M.C.S. NET
Assistant Professor, Computer Science Dept.
Dr. D. Y. Patil ACS College, Pimpri, Pune

Mr. Kamil Ajmal Khan
M.Sc. CS, MCA, PhD Pursuing
Assistant Professor
B.C.A./M.C.A. Department
Abeda Inamadar Senior College, Camp, Pune

Price 180.00

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N3371

JAVA PROGRAMMING (MCA - II) **ISBN 978-93-6109-502-3**

First Edition : January 2025
© : Authors

The text of this publication, or any part thereof, should not be reproduced or transmitted in any form or stored in any computer storage system or device for distribution including photocopy, recording, taping or information retrieval system or reproduced on any disc, tape, perforated media or other information storage device etc., without the written permission of Authors with whom the rights are reserved. Breach of this condition is liable for legal action.

Every effort has been made to avoid errors or omissions in this publication. In spite of this, errors may have crept in. Any mistake, error or discrepancy so noted and shall be brought to our notice shall be taken care of in the next edition. It is notified that neither the publisher nor the authors or seller shall be responsible for any damage or loss of action to any one, of any kind, in any manner, therefrom. The reader must cross check all the facts and contents with original Government notification or publications.

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar,
Off J.M. Road, Pune – 411005
Tel - (020) 25512336/37/39
Email : niralipune@pragationline.com

Polyplate

Printed By :
YOGIRAJ PRINTERS AND BINDERS
Survey No. 10/1A, Ghule Industrial Estate
Nanded Gaon Road
Nanded, Pune - 411041

DISTRIBUTION CENTRES

PUNE

Nirali Prakashan
(For orders outside Pune)
S. No. 28/27, Dhayari Narhe Road, Near Asian College
Pune 411041, Maharashtra
Tel : (020) 24690204; Mobile : 9049672225
Email : bookorder@pragationline.com

Nirali Prakashan
(For orders within Pune)
119, Budhwar Peth, Jogeshwari Mandir Lane
Pune 411002, Maharashtra
Mobile : 9657703145, 9890997937
Email : niralilocal@pragationline.com

MUMBAI

Nirali Prakashan
Rasdhara Co-op. Hsg. Society Ltd., 'D' Wing Ground Floor, 385 S.V.P. Road
Girgaum, Mumbai 400004, Maharashtra
Mobile : 7045821020, Tel : (022) 2385 6339 / 2386 9976
Email : niralimumbai@pragationline.com

DISTRIBUTION BRANCHES

DELHI

Nirali Prakashan
Room No. 2 Ground Floor
4575/15 Omkar Tower, Agarwal Road
Darya Ganj, New Delhi 110002
Mobile : 9555778814/9818561840
Email : delhi@niralibooks.com

BENGALURU

Nirali Prakashan
Maitri Ground Floor, Jaya Apartments,
No. 99, 6th Cross, 6th Main,
Malleswaram, Bengaluru 560003
Karnataka; Mob : 9686821074
Email : bengaluru@niralibooks.com

NAGPUR

Nirali Prakashan
Above Maratha Mandir, Shop No. 3,
First Floor, Rani Jhanshi Square,
Sitabuldi Nagpur 440012 (MAH)
Tel : (0712) 254 7129
Email : nagpur@niralibooks.com

KOLHAPUR

Nirali Prakashan
438/2, Bhosale Plaza, Ground Floor
Khasbag, Opp. Balgopal Talim
Kolhapur 416 012, Maharashtra
Mob : 9850046155
Email : kolhapur@niralibooks.com

JALGAON

Nirali Prakashan
34, V. V. Golani Market, Navi Peth,
Jalgaon 425001, Maharashtra
Tel : (0257) 222 0395
Mob : 94234 91860
Email : jalgaon@niralibooks.com

SOLAPUR

Nirali Prakashan
R-158/2, Avanti Nagar, Near Golden
Gate, Pune Naka Chowk
Solapur 413001, Maharashtra
Mobile 9890918687
Email : solapur@niralibooks.com

marketing@pragationline.com | www.niralibooks.com

Also find us on www.facebook.com/niralibooks

Preface ...

We take this opportunity to present this book entitled as **"Java Programming"** to the students of Second Semester - MCA (Management). The object of this book is to present the subject matter in a most concise and simple manner. The book is written strictly according to the New Syllabus (NEP Pattern 2020).

The book has its own unique features. It brings out the subject in a very simple and lucid manner for easy and comprehensive understanding of the basic concepts, its intricacies, procedures and practices. This book will help the readers to have a broader view on Java Programming. The language used in this book is easy and will help students to improve their vocabulary of technical terms and understand the matter in a better and happier way.

We sincerely thank Shri. Dineshbhai Furia and Shri. Jignesh Furia of Nirali Prakashan, for the confidence reposed in me and giving me this opportunity to reach out to the students of management studies.

We have given our best inputs for this book. Any suggestions towards the improvement of this book and sincere comments are most welcome on niralipune@pragationline.com.

Authors

■■■

Syllabus ...

1. Basics of Java **[Weightage 25]**

- 1.1 Class and Objects
- 1.3 Abstraction, Polymorphism Inheritance and Encapsulation
- 1.4 Abstract Class, Interface
- 1.4 Garbage Collector
- 1.5 Lambda Expression

2. Advanced Java Concepts **[Weightage 20]**

- 2.1 Introduction to Collection Framework
  - 2.1.1 Arraylist, Vector, Set, Map, Hashing
- 2.2 Multithreading
  - 2.2.1 Thread Life-Cycle
  - 2.2.2 Thread Priorities
  - 2.2.3 Synchronizing Threads
  - 2.2.4 Inter Communication of Threads
- 2.3 Exception Handling
  - 2.3.1 Types of Exception
  - 2.3.2 Keywords
  - 2.3.3 User Defined Exception

3. Servlets **[Weightage 20]**

- 3.1 Fundamentals of Java Servlet Programming
- 3.2 A Simple java Servlet
- 3.3 Servlet Life Cycle
- 3.4 Developing and Deploying Servlets
- 3.5 Working with Cookies

4. Java Server Pages **[Weightage 20]**

- 4.1 JSP Overview-Installation
- 4.2 JSP Tags - Components of a JSP Page
- 4.3 Expressions Script Lets - Directives, JSP Object
- 4.4 JDBC Connectivity

5. Spring MVC **[Weightage 15]**

- 5.1 Overview of the Spring Framework
- 5.2 Spring MVC Annotation
- 5.3 Spring MVC Architecture
- 5.4 Spring MVC Flow
- 5.5 Spring Form Handling
- 5.6 Spring Core and Spring Boot Dependency Injection and Inversion of Control (IOC)

■■■

Contents ...

1.	Basics of Java	1.1 – 1.24
2.	Advanced Java Concepts	2.1 – 2.53
3.	Servlets	3.1 – 3.16
4.	Java Server Pages	4.1 – 4.22
5.	Spring MVC	5.1 – 5.27

■■■

1...

Basics of Java

Objectives...

- To familiarize students with the concepts of OOPs.
- To study the Fundamentals of Class.
- To learn how to create an object of a Class.

1.1 INTRODUCTION

- Java is pure Object-Oriented programming Language. It supports to the following object-oriented concepts: Class, Objects, Polymorphism, Inheritance, Encapsulation, Abstraction.

OOPs (Object-Oriented Programming System):

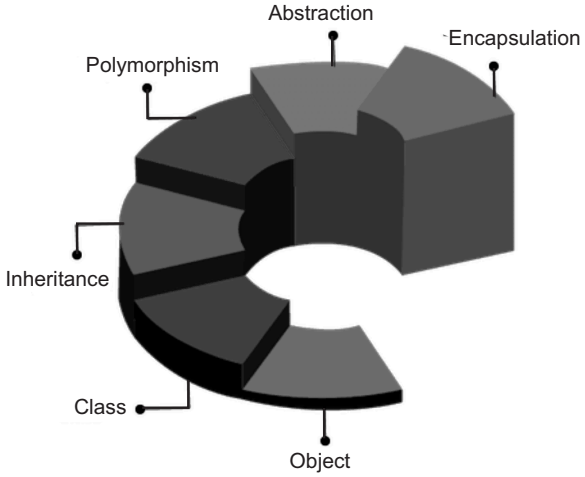


Fig. 1.1: OOPs Concept

1.2 CLASS AND OBJECTS

1. Class:

- Collection of objects is called class. It is a logical entity.
- A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

- Class is most important part of object-oriented programming. It is user-defined data type.
- The Collection of data members and member functions is called class. OR The binding of data members and member functions in a unit that unit is called class.

  Example: Class is Person.

  Syntax:

```
class ClassName[extends SuperClass]
{
   Variable Declarations
   Method Definition /Declaration
}
```

  Example:

```
class Add
{
   int a, b ,c;
   void disp()
   {
      c=a+b;
      System.out.println("Addition Is " + c);
   }
}
```

- The ClassName is user defined name given to the class. When giving name to the class, the naming convention must be followed. The naming convention is nothing but set of rules: Initial character of class name must be capital and name of class must be meaningful.
- The *extends* is a keyword used to inherit super class into the subclass.
- The variables and methods declared in a class without using any keyword are called instance variables and instance methods respectively because for accessing them object of class has to use.

2. Object:

- Any entity that has state and behavior is known as an object. For example, chair, pen, table, keyboard, bike, etc. It can be physical or logical.

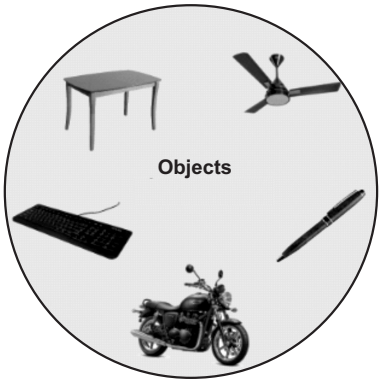


Fig. 1.2: Runtime example of an Objects

- An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

  Example: A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

1.2.1 Creating and Operating Object

- An object of a class can be created by using new operator as follow:
  Syntax: `ClassNameobj=new ClassName ();`
- By using an object of a class, we can access data member and member functions of class.

```
obj.data_member
obj.functionName()
```

Example:

```
class Add
{
   int a, b ,c;
   void disp()
   {
      c=a+b;
      System.out.println("Addition Is" + c);
   }
}
```

- We can create an object of Add class as follow:

```
Add obj=new Add();
obj.a=10;
obj.b=20;
obj.disp();
```

1.2.2 Object Reference

- In Java, there is no pointer because there is reference variable, it works as pointer. It can be declared as follows:

```
ClassNameVar_Name;
```

Example:

```
class Emp
{
   int eno,sal;
   String en;
   void disp()
   {
      System.out.println( eno + " " + en + " " + sal);
   }
}
Emp ob;
```

- Here, ob is a reference variable of class Emp, it always points to the null value, we cannot access data member and member functions of a class by using reference variable, if we try to access then compiler throws NullPointerException.
- If we store reference of an object into the reference variable of same class then by using reference variable, we can access data member and member functions of that class.

```
Emp obj=new Emp();
ob=obj;
ob.eno=10;
ob.en="DPU";
ob.sal=12345;
ob.disp();
```

- If we call method of a class by using reference variable of that class at run time then that concept is called *dynamic method dispatching*.

1.2.3 Object Life Time

- In Java, it has seven states in Object lifecycle. They are:

1. Created state:

- The following are some actions performed when an object is created:
  - New memory is allocated for an object.
  - Once the object has been created, assuming that it is assigned to some variable and then it directly moves to the In Use state.

2. In Use state:

- Objects that are held by at least one strong reference are considered to be “In Use".

3. Invisible state:

- An object is in the “Invisible" state when there are no longer any strong references that are accessible to the program, even though there might still be references.

4. Unreachable state:

- An object enters an “unreachable" state when no more strong references to it exist. When an object is unreachable then it is a state for collection.
- It is important to note that not just any strong reference will hold an object in memory. These must be references that chain from a garbage collection root.
- Garbage collection roots are a special class of variable that includes, Temporary variables on the stack.

5. Collected state:

- An object is in the “collected" state when the garbage collector has recognized an object as unreachable and prepares it for final processing as a precursor to deallocation. If the object has a finalize method, then it is marked for finalization.

6. Finalized state:

- An object is in the "finalized" state if it is still unreachable after the finalize method, if any, has been run. A finalized object is awaiting deallocation. If you are considering using a finalized to ensure that important resources are freed in a timely manner, you might want to reconsider. To lengthening object lifetimes, finalize methods can increase object size.

7. Deallocated state:

- The deallocated state is the final step in Garbage collection. If an object is still unreachable after all the above work has done, then this is the state for deallocation.

1.3 ABSTRACTION, POLYMORPHISM, INHERITANCE, AND ENCAPSULATION

1.3.1 Abstraction

- Hiding internal details and showing functionality is known as **Abstraction**. For example, when doing a phone call, we don't know the internal processing of it.
- In Java, we use abstract class and interface to achieve abstraction.



Capsule

Fig. 1.3: Implementation of Abstraction

1.3.2 Polymorphism

- If one task is performed in different ways, it is known as **Polymorphism**. For example, to convince the customer differently, to draw something, such as shape, triangle, rectangle, etc.
- In Java, we use Method overloading and Method overriding to achieve polymorphism.
- Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.
- Polymorphism means one object many forms. We can implement it in two ways:
  (a) Compile Time (Method overloading)
  (b) Run Time (Method overriding).
- Polymorphism in Java occurs when there are one or more classes or objects related to each other by inheritance.
- In other words, it is the ability of an object to take many forms. Inheritance lets users inherit attributes and methods, and polymorphism uses these methods for performing different tasks. So, the goal is communication but the approach is different.
- For example, you have a Smartphone for communication. The communication mode you choose could be anything. It can be a call, a text message, a picture message, mail, etc. So, the goal is common that is communication, but their approach is different. This is called Polymorphism.
- For implementing polymorphism in inheritance, method overriding is used.

Program 1.1: Write a Java program to calculate area of circle and triangle.

(Use Method Overriding).

```
import java.util.*;
class Triangle
{
   float a,b,h;
   Triangle(float b,float h)
   {
      this.b=b;
      this.h=h;
   }
   void area()
   {
      a=0.5f*b*h;
      System.out.println ("Area of Triangle Is " + a);
   }
}
class Circle extends Triangle
{
   float r;
   Circle(float b, float h, float x)
   {
      super(b,h);
      r=x;
   }
}
```

Program 1.2: Extending Classes and Inheritance.

```
void area()
{
   super.area();
   a=3.14f*r*r;
   System.out.println("Area of Circle Is " + a);
 }
class MethOver
{
   public static void main(String args[])
   {
      float b,h,x,r;
      Scanner ob=new Scanner (System.in);
      System.out.println ("Enter the values for B, H and R");
      b=ob.nextFloat ();
```

```
        h=ob.nextFloat ();
        r=ob.nextFloat ();
        Circle obj=new Circle (5,6,7);
        obj.area();
    }
}
```

Output:

```
Enter the values for B, H and R
5
6
7
Area Of Triangle Is 15.0
Area Of Circle Is 153.86002
```

1.3.3 Inheritance

- When one object obtains all the properties and behaviors of a parent object, it is known as *inheritance*. It provides code reusability. It is used to achieve runtime polymorphism.



Fig. 1.4: Real Time implementation of inheritance

- Inheritance is one of the most important features of Java.
- The process by which one class acquires the properties (data members) and functionalities (methods) of another class are called **Inheritance**.
  - **Child Class:** The class that extends the features of another class is known as Child class, Sub class or Derived class.
  - **Parent Class**: The class whose properties and functionalities are used (inherited) by another class is known as Parent class, Super class or Base class.

- Inheritance is a mechanism in which one object acquires all the properties and behaviors of a parent object. The idea behind inheritance is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class.
- Java does not support to the Multiple inheritance.

Use of Inheritance in OOP:

- The use of inheritance is to provide the reusability of code so that a class has to write only the unique features and rest of the common properties and functionalities can be extended from another class.
- Mostly we use inheritance, so that, we can use specific portions of code and can modify certain features according to our need and this can be done without any complexity. Inheritance provides flexibility in our code to reuse it from base class to required derived class. A child class can override properties of base class without rewriting code in the same class again and again.
- Inheritance is implemented by using *extends* keyword in Java.

  Syntax:

```
class Child-class-name extends Parent-class-name
{
   //methods and fields
}
```

Program 1.3: Write a Java program for the implementation of Inheritance.

```
class Teacher
{
   String designation ="Teacher";
   String collegeName="DPU";
   void does()
   {
      System.out.println ("Teaching");
   }
}
class JavaTeacher extends Teacher
{
   StringmainSubject="Computer";
   public static void main(String args[])
   {
      JavaTeacher obj=new JavaTeacher ();
      System.out.println (obj.collegeName);
      System.out.println (obj.designation);
      System.out.println (obj.mainSubject);
      obj.does ();
   }
}
```