edureka!

MASTERING JAVA WITH EDUREKA

A Guide to Learn Java Programming

Don't just Learn it, Master it!

TABLE OF CONTENTS

TABLE OF CONTENTS

Chapter 1

INTRODUCTION TO JAVA PROGRAMMING

*Java has become an important programming language in today's world with its universal presence in our day-to-day life. Released by Sun Microsystems in 1995, this class-based object-oriented program is often related to the likes of C & C++ due to its similarity in coding. Often termed as the programming language where you can '**Write Once, Run Anywhere**'. This principle of Java makes it an eye-catching language!*

1.1 What is Java Programming?

Java is an object-oriented language with a C/C++-like syntax that is familiar to many programmers. It is dynamically linked, allowing new code to be downloaded and run, but not dynamically typed.

01 Concurrent	02 OOPs	03 WORA
Java is concurrent where you can execute many statements instead of sequentially executing it.	Java is class-based and an object-oriented programming language.	Java is a platform independent programming language that follows the logic of "Write Once, Run Anywhere".

These are the reasons which attract the programmers the most. Java programs can be executed across any machine having JRE. JRE is compatible with all devices, say, mobile phones, PCs as well as any OS like Linux, Windows, Mac, Android, etc. We will learn more about JRE, later in this book.

JAVA FEATURES

Easy to Learn & Use

Free and Open Source

High-Level Language

Highly Portable

Object-Oriented Programming

Comprehensive Set of Libraries

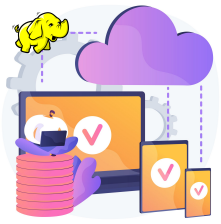
JAVA APPLICATIONS

Web Servers Development

Game Development

Desktop GUI Development

Mobile Application Development



Big Data & Cloud-Based App Development

Scientific & Mathematical Code Development

Chapter 2

JAVA INSTALLATION

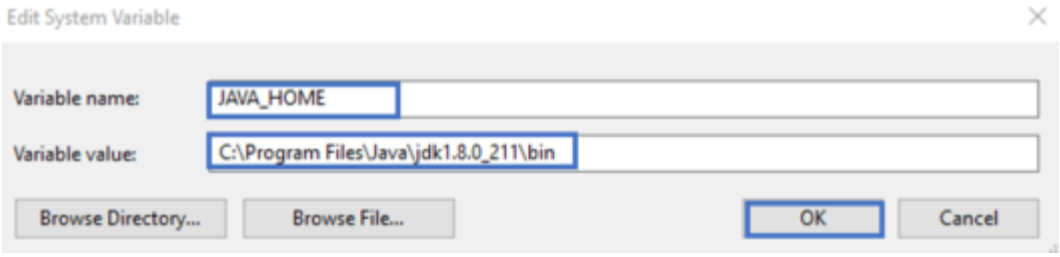
2.1 Downloading Java JDK

STEPS

1. Go to the Java Downloads Page and click on the option of **Download**
2. Choose the download link according to your matching system configuration
3. Once the file is downloaded, run the installer and keep clicking on **Next**, till you finally get a dialog box, which says, you have finished downloading.

2.2 Java Environment Setup on Windows

1. Go to '**Start**' and search for '**System**'. Click on '**System**' and go to '**Advanced System Settings**'
2. Now, click on '**Environment Variables**' under the '**Advanced**' tab
3. Next, under the '**System Variables**' choose '**New**'
4. Enter the variable name as '**JAVA_HOME**' and the full path to the Java installation directory as per your system as shown below:



5. Next, you have to edit the path. For that, select '**path**' under '**System Variable**' and click on '**Edit**'
6. Under '**Variable value**', at the end of the line, enter the following path – **%JAVA_HOME%bin**;
7. Now, you can click '**OK**' and you are done
8. Now to cross-check the installation, just run the following command in **Command Prompt**, it should display the installed version of Java in your system

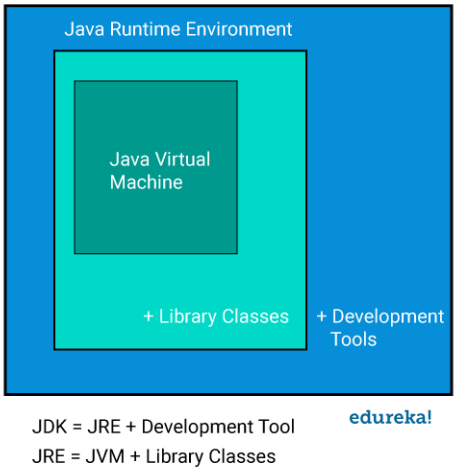
```
java -version
```

Chapter 3

JAVA FUNDAMENTALS

3.1 JVM, JRE and JDK

Java applications are called WORA (Write once Run Anywhere) because of their ability to run a code on any platform. This is done only because of **JVM (Java Virtual Machine)**. The JVM is a Java platform component that provides an environment for executing Java programs. JVM interprets the bytecode into machine code which is executed in the machine in which the Java program runs.



JDK = JRE + Development Tool
JRE = JVM + Library Classes

The **JRE** software builds a runtime environment in which Java programs can be executed. The JRE is the on-disk system that takes your Java code, combines it with the needed libraries, and starts the JVM to execute it. The JRE contains libraries and software needed by your Java programs to run. JRE is a part of JDK but can be downloaded separately.

The **Java Development Kit (JDK)** is a software development environment used to develop Java applications and applets. It contains JRE and several development tools, an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) accompanied with another tool.

3.2 Data Types in Java

A data type is an attribute of a variable that tells the compiler or interpreter how the programmer intends to use the variable. It defines the operations that can be done on the data and what type of values can be stored. According to the properties they possess, they are divided into two groups:

01 Primitive Data Types

- Boolean Data Type
- Byte Data Type
- Char Data Type
- Short Data Type
- Int Data Type
- Long Data Type
- Float Data Type
- Double Data Type

02 Non-Primitive Data Types

- Strings
- Arrays
- Classes
- Interface
- Enumerations

3.3 Variables and Keywords

Variable in Java is the basic unit of storage. It acts as a container and is used to hold data values. The values held by the variable can be changed during the execution of the program. Every variable is assigned a data type. Variable is a name given to a memory location. A variable is declared by specifying the following parameters:

1. **Datatype:** Type of data stored in the variable
2. **Variable name:** A unique name of the variable
3. **Value:** The initial value stored in the variable

```
int age = 50 ;
float weight = 50.60;
```

Keywords are predefined which have a unique meaning and functionality in the Java programming language. These keywords are also known as reserved keywords which means they cannot be used as a variable name, class, method, or any other identifier. There are 57 reserved keywords in Java, some which are listed below:

- abstract
- continue
- for
- new
- switch
- package
- private
- this
- break
- implements
- throw
- import
- public
- throws
- enum
- return
- instanceof
- short
- interface
- static
- void

3.4 Operators in Java

Operators in Java are used for operations between two values or variables. The output varies according to the type of operator used in the operation. We can call operators as special symbols or constructs to manipulate the values of the operands. Consider the expression 2 + 3 = 5, here 2 and 3 are operands and + is called operator.

Type	Operators
Arithmetic	+, -, *, /, %, **, //
Assignment	=, +=, -=, *=, %=, **=, //=, |=, ^=, &=
Relational	==, !=, >, <, <=,>=
Logical	&&, ||, !
Unary	++, --, !
Bitwise	&, |, ^, ~
Ternary	(Condition) ? (Statement1) : (Statement2);
Shift	<<,>>,>>>

3.5 Methods in Java

A method is basically a set of code which is referred to by name and can be called or invoked at any point in a program, just by utilizing the method's name. Each method is given its own name. When that name is in a program, the execution branches to the body of that method.

How to create a method in Java?

A method in Java must be declared within a specific class. It is defined with the name of the method, followed by parentheses "()". Java provides some pre-defined methods, such as System.out.println(), etc.

```
public static dataType methodName
(dataType x, dataType y)
{
      // body
}
```

Every Java program must have the '**main**' method. It is the entry point for the Java Compiler, from where it starts the execution and follows the order specified in the rest of the program.

public static void main (String args[])

- **public**
- **static**
- **void**
- **main**
- **String args[]**

3.6 Access Modifiers in Java

Access modifiers in Java are used to specify the access levels for classes, variable methods, and constructors. It helps in updating the value of a variable. They are also known as access/visibility modifiers. There are four access modifiers keywords in Java and they are:

1

DEFAULT ACCESS MODIFIER

When no access modifier is specified for a particular class, method or a data member, it is said to be having the default access modifier. The data members, class or methods which are not declared utilizing any entrance modifiers, will have default modifier which is accessible only inside a similar bundle. It means you do not explicitly declare an access modifier for a class, field, method, etc.

2

PRIVATE ACCESS MODIFIER

The methods or data members that are declared as private are only accessible within the class in which they are declared. Top-level classes or interfaces cannot be declared as private in light of the fact that 'private' signifies "just visible inside the enclosing class". If a class has a private constructor then you cannot create the object of that class from outside the class.

3

PUBLIC ACCESS MODIFIER

The public access modifier is specified using the keyword public. It has a broadest scope among all other access modifiers. Classes, methods or data members which are declared as public are accessible anywhere throughout the program. There is no restriction on the scope of public data members.

4

PROTECTED ACCESS MODIFIER

The protected access modifier is specified using the keyword protected. The methods or data members declared as protected are accessible within the same package or subclasses in a different package. Protected members can be accessed only in the child or derived classes.

Chapter 4

JAVA FLOW CONTROL

A control statement in Java determines whether the other statements will be executed or not.

4.1 Decision Making Statements

Statements that determine which statement to execute and when are known as decision-making statements. The flow of the execution of the program is controlled by the control flow statement. There are four decision-making statements available in Java.

1 if Statement

The 'if' statement determines whether a code should be executed based on the specified condition.

```
if (condition) {
Statement 1; //executed if condition is true
} Statement 2; //executed irrespective of the condition
```

2 if..else Statement

In this statement, in case the condition specified is true, then 'if' block is executed. Otherwise, the 'else' block is executed.

```
if (condition) {
Statement 1; //executed if condition is correct
} else Statement 2; /*executed if condition is false*/
```

3 Nested if Statement

An 'if' present inside another 'if' block is known as a nested 'if' block. It is similar to an if..else statement, except they are defined inside another if..else statement.

```
if (condition1) {
Statement 1;
if (condition2) {
Statement 2; /*executed 'if' 2nd condition is correct*/
} else {
Statement 3; /*executed if second condition is false*/
}}
```

4 switch-case Statement

A 'switch' statement in Java is used to execute a single statement from multiple conditions. The switch statement can be used with short, byte, int, long, enum types.

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

4.2 Iterative Statements

Statements that execute a block of code in a loop until a specified condition is met are known as Iterative statements. Java provides the user with three types of loops:

1 while Loop

A while loop in Java is used to iterate over a block of code or statements as long as the test expression is true. You can use this, in case number of iterations are not specified.

```
while (condition) {
/*code block to be executed*/
}
```

2 do-while Loop

It is similar to a while loop, but in a while loop, the condition is evaluated before the execution of the loop's body but in a do-while loop, the condition is evaluated after the execution of the loop's body.

```
do { /*code block to be executed*/
} while (condition)
```

3 for Loop

The for loop in Java is used to iterate and evaluate a code multiple times. When the number of iterations is known by the user, it is recommended to use the for loop.

```
for (initialization; condition; increment/decrement)
{
statement;
}
```

4 for-each Loop

The traversal of elements in an array can be done by the for-each loop. The elements present in the array are returned one by one.

```
for (type var : array)
{
    statements using var;
}
```

Example

```
public class Edureka {
public static void
pyramidPattern(int n) {
for (int i=0; i<n; i++) {
for (int j=n-i; j>1; j--)
  { System.out.print(" "); }
for (int j=0; j<=i; j++ )
  { System.out.print("* "); }
System.out.println();
 }
}

public static void main(String
args[]) //driver function
{ int n = 5;
        pyramidPattern(n);
    }
}
```

Chapter 5

OBJECT-ORIENTED PROGRAMMING IN JAVA

Object-Oriented Programming (OOP) refers to a type of programming in which programmers define the data type of a data structure and the type of operations that can be applied to the data structure. An object-based application in Java is based on declaring classes, creating objects from them and interacting between these objects. Let's take a look at the building blocks of object-oriented programming:

5.1 Classes & Objects

Class in Java is a blueprint from which an object is created. It is a logical entity that helps in defining the behavior and properties of an object. A class can only be accessed from outside via its instance.

01 Built-in Classes

Built-in classes in Java are the classes that comes bundled within predefined packages in Java. E.g:

- Java.Lang.String
- Java.Lang.System
- Java.Lang.Exception
- Java.Lang.Object

02 User Defined Classes

As the name suggests, a custom or user-defined class is a class that is created by a user. It will contain the class members as defined by the user.

Class Elements

A Java class generally consists of the following:

1. Fields
2. Methods
3. Constructors
4. Blocks
5. Nested Classes

Class Syntax

```
<access specifier> class <classname>{
//classbody
}
```

Object in Java is a real-world entity that has its own property and behavior. These are considered to be the fundamental concepts of Java and uses classes as its blueprint. A Java program can have as many objects as required. An object in Java typically consists of the following:

1. **State:** This is represented by the attributes and properties of an object.
2. **Behavior:** This is defined by the methods of an object.
3. **Identity:** This provides a unique name to an object and enables the communication between them.

5.2 Abstraction in Java

Abstraction refers to the quality of dealing with ideas rather than events. It basically deals with hiding the details and showing the essential things to the user. In Java, you can achieve abstraction in two ways:

1. **Abstract Class:** Abstract class contains the 'abstract' keyword and cannot be instantiated. It can contain abstract as well as concrete methods.
2. **Interface:** An interface in Java is a collection of abstract methods and static constants. In an interface, each method is public and abstract but it does not contain any constructor.

```
public interface EduInterface{
    public void show();
    public void run();
}

public class eduDemo
implements EduInterface{
    public void show(){
        //implementation
    }
    public void run(){
        //implementation
    }
}
```

5.3 Encapsulation in Java

Encapsulation refers to the process of wrapping up of data under a single unit. It is the mechanism that binds code and the data it manipulates. Another way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield. In this, the variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared.

Encapsulation in Java can be achieved by:

1. Declaring the variables of a class as private.
2. Providing public setter and getter methods to modify and view the values of the variables.

```
class Student {
private String name;
public String getName() {
return name;
}
public void setName(String name)
{
this.name = name;
}
}
public class Main {
public static void main(String[]
args) {
Student s=new Student();
s.setName("Harry Potter");
System.out.println(s.getName());
}
}
```

5.4 Inheritance in Java

Inheritance is an integral part of Java OOPs which lets the properties of one class to be inherited by the other. It basically, helps in reusing the code and establish a relationship between different classes.

TYPES OF INHERITANCE

Single Inheritance

Multilevel Inheritance

Hierarchical Inheritance

Hybrid Inheritance

```
class Animal {
void eat() {
System.out.println("eating…"); }
}
class Dog extends Animal {
void bark() {
System.out.println("barking…"); }
}
class Cat extends Animal {
void meow() {
System.out.println("meowing…"); }
}
class TestInheritance3 {
public static void main(String
args[]){
Cat c=new Cat();
c.meow();
c.eat();
}
}
```

5.5 Polymorphism in Java

Polymorphism in OOP is the ability of an entity to take several forms. In other words, it refers to the ability of an object (or a reference to an object) to take different forms of objects. It allows a common data-gathering message to be sent to each class. Polymorphism encourages a concept called 'extendibility' which means an object or a class can have its uses extended.

TYPES OF POLYMORPHISM

Static Polymorphism

Dynamic Polymorphism

```
class Calculator
{
int add(int x, int y)
{
return x+y;
}
int add(int x, int y, int z)
{
return x+y+z;
}
}
public class Test
{
public static void main(String args[])
{
Calculator obj = new Calculator();
System.out.println(obj.add(100, 200));
System.out.println(obj.add(100, 200,
300));
}
}
```

Chapter 6

DATA STRUCTURES IN JAVA

A Data Structure is a way of storing and organizing data in a computer so that it can be used efficiently. It provides a means to manage large amounts of data efficiently. These are categorized into two types:

6.1 Linear Data Structures

Linear data structures in Java are those whose elements are sequential and ordered in a way so that there is only one first element and has only one next element, there is only one last element and has only one previous element, while all other elements have a next and a previous element.

1 ARRAYS

An array represents a group of similar data type elements, accessed by an index. The size of an array must be provided before storing data.

2 LINKED LIST

A linked list is a linear data structure with the collection of multiple nodes, where each element stores its own data and a pointer to the location of the next element. The last link in a linked list points to null, indicating the end of the chain. An element in a linked list is called a node. The first node is called the head. The last node is called the tail.

3 STACKS

Stack, an abstract data structure, is a collection of objects that are inserted and removed according to the last-in-first-out (LIFO) principle. Objects can be inserted into a stack at any point of time, but only the most recently inserted (that is, “last”) object can be removed at any time.

4 QUEUES

Queues are also another type of abstract data structure. Unlike a stack, the queue is a collection of objects that are inserted and removed according to the first-in-first-out (FIFO) principle. That is, elements can be inserted at any point of time, but only the element that has been in the queue the longest can be removed at any time.

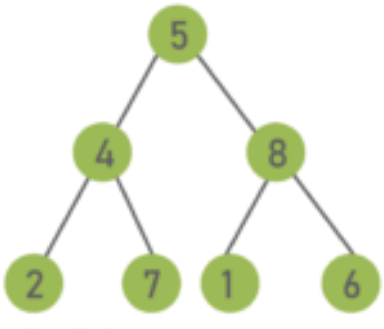
6.2 Hierarchical Data Structures

Hierarchical data structures in Java store elements on the basis of their hierarchy. These types of data structures are efficient for visualizing and retrieving the data.

1 BINARY TREE

Binary Tree is a hierarchical tree data structure in which each node has at most two children, which are referred to as the left child and the right child. Each binary tree has the following group of nodes:

- **Root Node:** It is the topmost node and often referred to as the main node because all other nodes can be reached from the root
- **Left Sub-Tree**, a sub- binary tree on the LHS of the Root Node
- **Right Sub-Tree**, a sub-binary tree on the RHS of the Root Node



2 BINARY HEAP

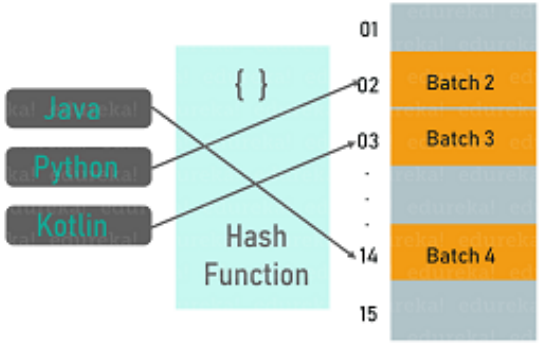
Binary Heap is a complete binary tree, which answers to the heap property. In simple terms, it is a variation of a binary tree with the following properties:

- **Heap is a complete binary tree:** A tree is said to be complete if all its levels, except possibly the deepest, are complete. This property of Binary Heap makes it suitable to be stored in an array.
- **Follows heap property:** A Binary Heap is either a Min-Heap or a Max-Heap.
- **Min Binary Heap:** For every node in a heap, node's value is lesser than or equal to values of the children.
- **Max Binary Heap:** For every node in a heap, the node's value is greater than or equal to values of the children.

3 HASH TABLES

Imagine that you have an object and you want to assign a key to it to make searching very easy. To store that key/value pair, you can use a simple array like a data structure where keys (integers) can be used directly as an index to store data values. However, in cases where the keys are too large and cannot be used directly as an index, a technique called hashing is used.

In hashing, the large keys are converted into small keys by using hash functions. The values are then stored in a data structure called a hash table. A hash table is a data structure that implements a dictionary ADT, a structure that can map unique keys to values.



In general, a hash table has two major components:

1. Bucket Array
2. Hash Function

Chapter 7

ADVANCE JAVA CONCEPTS

This chapter will introduce you to some of the advanced concepts of Java which serves major functionalities in making a program efficient.

7.1 Multithreading in Java

A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread and each thread defines a separate path of execution. Thus, multithreading is a specialized form of multitasking. Java's multithreading system is built upon the Thread class, its methods, and its companion interface i.e., Runnable.

HOW TO CREATE A JAVA THREAD?

Java lets you create a thread in the following two ways:-

1. By implementing the Runnable interface
2. By extending the Thread

CREATING MULTIPLE THREADS

```
class MyThread implements Runnable {
String name;
Thread t;
MyThread (String threadname){
name = threadname;
t = new Thread(this, name);
System.out.println("New thread: " + t);
t.start();
}
public void run() {
try { for(int i = 5; i > 0; i--)
{ System.out.println(name + ": " + i);
Thread.sleep(1000);
}
}catch (InterruptedException e) {
System.out.println(name + "Interrupted");
}
System.out.println(name + " exiting.");
} }

class Main {
public static void main(String args[]) {
new MyThread("One");
new MyThread("Two");
new MyThread("Three");
try { Thread.sleep(10000);
} catch (InterruptedException e) {
System.out.println("Main                thread
Interrupted");
}
System.out.println("Main thread exiting.");
} }
```

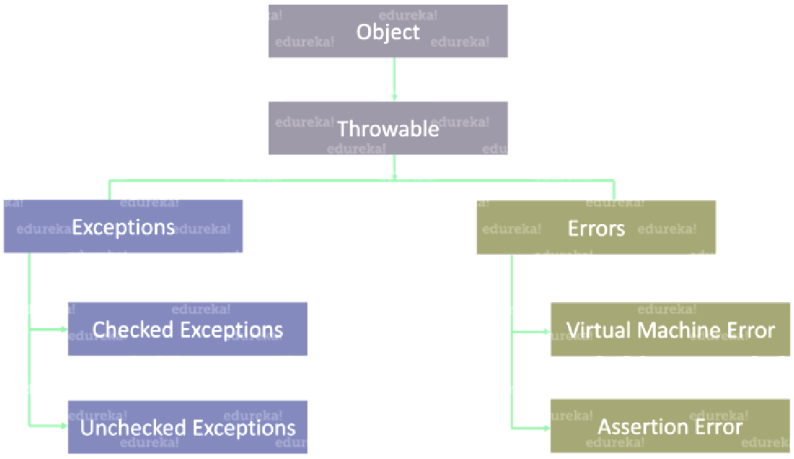
OUTPUT

```
New thread: Thread[One,5,main]
New thread: Thread[Two,5,main]
New thread: Thread[Three,5,main]
One: 5
Two: 5
Three: 5
One: 4
Two: 4
Three: 4
One: 3
Three: 3
Two: 3
One: 2
Three: 2
Two: 2
One: 1
Three: 1
Two: 1
One exiting.
Two exiting.
Three exiting.
Main thread exiting.
```

7.2 Exception Handling in Java

Errors arise unexpectedly and can result in disrupting the normal flow of execution. This is something that every programmer faces at one point or the other while coding.

Java, being the most prominent object-oriented language, provides a powerful mechanism to handle these errors/exceptions called Exception Handling. All exception and error types are subclasses of class **Throwable**, which is the base class of the hierarchy. One branch is headed by **Error** which occurs at run-time and the other by **Exception** that can happen either at compile-time or run-time.



7.3 Java DataBase Connectivity (JDBC)

JDBC is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases. This application program interface lets you encode the access request statements, in SQL. They are then passed to the program that manages the database. It mainly involves opening a connection, creating a SQL Database, executing SQL queries, and then arriving at the output.

The JDBC API supports both two-tier and three-tier processing models for database access but in general, JDBC Architecture consists of two layers:

1. **JDBC API**: This provides the application-to-JDBC Manager connection.
2. **JDBC Driver API**: This supports the JDBC Manager-to-Driver Connection.

Chapter 8

JAVA PRACTICE PROGRAMS

8.1 Palindrome Program using For Loop

```
public class PalindromeProgram {
public static void main(String[] args) {
int n=1234521, rev=0, rem, temp;
temp = n;
for( ;n != 0; n /= 10 )
{
  rem = n % 10;
  rev= rev* 10 + rem;
}
if (temp== rev)
System.out.println(temp + " is a
palindrome.");
else
System.out.println(temp + " is not a
palindrome.");
    }
}
```

8.2 Calculate Permutation & Combination

```
package Edureka;
import java.util.Scanner;
public class nprandncr {
//calculating a factorial of a number
public static int fact(int num)
{
int fact=1, i;
for(i=1; i<=num; i++)
{
fact = fact*i;
}
return fact;
}
public static void main(String args[])
{
int n, r;
Scanner scan = new Scanner(System.in);
System.out.print("Enter Value of n : ");
n = scan.nextInt();
System.out.print("Enter Value of r : ");
r = scan.nextInt();
// NCR and NPR of a number
System.out.print("NCR = " +(fact(n)/(fact(n-
r)*fact(r))));
System.out.print("nNPR = " +(fact(n)/(fact(n-
r))));
}
}
```

8.3 Recursive Binary Search

```
public class BinarySearch {
// Java implementation of recursive Binary Search
// Returns index of x if it is present in
arr[l..h], else return -1
int binarySearch(int a[], int l, int h, int x)
{
if (h >= l) {
int mid = l + (h - l) / 2;
// If the element is present at the middle itself
if (a[mid] == x)
return mid;
// If element is smaller than mid, then it can
only be present in left subarray
if (a[mid] >x)
return binarySearch(arr, l, mid - 1, x);
// Else the element can only be present in right
subarray
return binarySearch(arr, mid + 1, h, x);
}
// We reach here when element is not present in
array
return -1;
}
public static void main(String args[])
{
BinarySearch ob = new BinarySearch();
int a[] = { 20, 30, 40, 10, 50 };
int n = a.length;
int x = 40;
int res = ob.binarySearch(a, 0, n - 1, x);
if (res == -1)

System.out.println("Element not present");
else
System.out.println("Element found at index " +
res);
}
}
```

JAVA PRACTICE PROGRAM EXAMPLES

Chapter 9

FREQUENTLY ASKED INTERVIEW QUESTIONS

Java has dominated the programming world since the early 2000s and has managed to keep its magic intact with its platform independence till date. This chapter covers the questions that will help you in your Java Interviews and open up various career opportunities as a **Java aspirant**.

1. Explain JDK, JRE and JVM.
2. Explain public static void main(String args[]) in Java.
3. Why Java is platform-independent?
4. Why Java is not 100% Object-oriented?
5. What are Wrapper classes in Java?
6. What are constructors in Java?
7. What is a singleton class in Java?
8. Differentiate between Arraylist and Vector in Java.
9. What is the difference between equals() and == in Java?
10. Differentiate between Heap and Stack Memory in Java.
11. What is Java String Pool?
12. What is Constructor Chaining in Java?
13. What is a classloader in Java?
14. What is a Map in Java?
15. What is runtime Polymorphism?
16. Differentiate between Abstract classes and Interfaces.
17. What is an Association?
18. What is a Marker interface?
19. What is a Servlet?
20. What is the life-cycle of a Servlet?
21. Explain what is MVC?
22. Describe the Collection hierarchy in Java.
23. What is Arraylist in Java?
24. How will you reverse a List?
25. What is a Priority Queue in Java?
26. What is the HashSet class in Java and how does it store elements?
27. What is the difference between Error and Exception?
28. Differentiate between final, finally, and finalize fulfill.
29. What is Synchronization?
30. What is OutOfMemoryError in Java?
31. What are the different types of Garbage Collectors in Java?
32. Can we write multiple catch blocks under single try block?
33. What is JDBC Driver?
34. What is JDBC Connection interface?
35. Differentiate between execute & executeQuery.
36. What do you understand by JDBC Statements?
37. What is Composition in Java?
38. What is the difference between break and continue statements?

CAREER GUIDANCE

WHO IS A JAVA PROFESSIONAL?
A Java Professional works mostly with Java to design, develop and build applications and websites that have dynamic elements.

Java Developer

A **Java Developer** is a specialized type of programmer who may collaborate with web developers and software engineers to integrate Java into business applications, software and websites. A Java Developer is responsible for the design, development, and management of Java-based applications. Because Java is used so widely, particularly by large organizations, the daily roles vary widely.

Android / Mobile App Developer

An **Android Developer** is responsible for developing applications for devices powered by the Android operating system. Due to the fragmentation of this ecosystem, an Android developer must pay special attention to the application's compatibility with multiple versions of Android and device types.

Java Web Developer

Java Web Developers write, test, and debug back-end code that supports websites and web applications. This role requires a high level of technical expertise and familiarity with both the Java programming language and related technologies, such as relational databases and servers. Java developers work closely with front-end developers to determine a project's specifications and scope and translate this information into functional, reliable, and scalable code.

Java Web Programmer

Java Web Programmers are responsible for designing, coding and modifying websites, from layout to function and according to a client's specifications. Strive to create visually appealing sites that feature user-friendly design and clear navigation.

Java Application Developer

Java Application Developer is responsible for developing software solutions to meet customer needs. Creating and implementing the source code of new applications. Testing source code and debugging code. Evaluating existing applications and performing updates and modifications.

EJB Programmer

The **EJB Developer** is a Java applications programmer, and is familiar with both SQL and with database access using SQLJ or JDBC. The EJB deployer installs and publishes the EJBs. This involves interaction with the EJB developer, so that the transactional nature of the EJBs are understood.

EDUREKA JAVA TRAINING PROGRAMS

JAVA CERTIFICATION TRAINING

Weekend | Live Class | 24 x 7 Technical Assistance

https://www.edureka.co/java-j2ee-training-course

SPRING FRAMEWORK TRAINING

Weekend | Live Class | 24 x 7 Technical Assistance

https://www.edureka.co/spring-certification-course

MICROSERVICES CERTIFICATION TRAINING

Weekend | Live Class/SP | 24 x 7 Technical Assistance

https://www.edureka.co/microservices-architecture-training

SELENIUM CERTIFICATION TRAINING

Weekend/Weekday | Live Class | 24 x 7 Technical Assistance

https://www.edureka.co/selenium-certification-training

LEARNER'S REVIEWS

EA — Elena AF

★★★★★

Clear and detailed instructions, great examples of problem solving mostly based on real-world cases, and **Java programming codes** were made available quickly for review right after class. Instructor takes time to listen and answer students' questions.

AK — Avinash Kulkarni

★★★★★

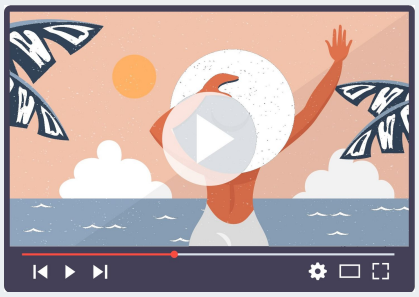
My journey started with Edureka last year, when i was in search of "**Micro-service Architecture Training**" course. My search ended at Edureka. It was the best learning experience. Edureka provides very well trained professionals. I would like to thank Edureka team for all the support.

AS — Ankit Sharma

★★★★★

Its very **simple in learning and interesting** too. The way our instructor is teaching us is simply awesome. The thing which I like the most about Edureka is its Support service,as I have got all my queries answered by them on time. Thank you Edureka :)

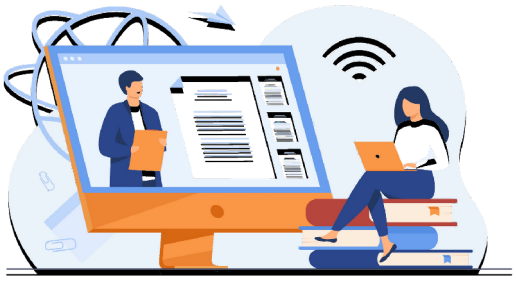
Free Resources



3000+
Video Tutorials on
YouTube

2500+ Technical
Blogs

Active
Community



30+
Free Monthly
Webinars

About Us

There are countless online education marketplaces on the internet. And there's us. We are not the biggest. We are not the cheapest. But we are the fastest growing. We have the highest course completion rate in the industry. We aim to become the largest online learning ecosystem for continuing education, in partnership with corporates and academia. To achieve that we remain ridiculously committed to our students. Be it constant reminders, relentless masters or 24 x 7 online technical support - we will absolutely make sure that you run out of excuses to not complete the course.

Contact Us

IndiQube ETA, 3rd Floor,
No.38/4,
Adjacent to Dell EMC2,
Dodanekundi,
Outer Ring Road, Bengaluru,
Karnataka - 560048

IN: 08035068109 | US: +1415 915 9044
www.instagram.com/edureka.co/
www.facebook.com/edurekaIN
www.linkedin.com/company/edureka/
www.youtube.com/user/edurekaIN
t.me/s/edurekaupdates
twitter.com/edurekaIN
in.pinterest.com/edurekaco/

News & Media

INDIA TODAY

Edureka partners with **NIT Warangal** to upskill IT professionals in **AI and Machine Learning**

Deloitte.

Edureka (Brain4ce Education Solutions) tops Deloitte Tech Fast 50 2014 rankings

edureka!