An ISO 9001 : 2015 Certified Company

NARESHi technologies

SOFTWARE TRAINING & DEVELOPMENT

040-2374 6666 | 23734842

info@nareshit.com

www.nareshit.com

nareshit nareshitech

nareshit

Find Latest IT Jobs SeshaJobs.com

Opp. Satyam Theatre, Durga Bhavani Plaza, Ameerpet, Hyd-16

# CORE JAVA

**Programming Languages**
- Low Level Programming Language
- High Level Programming Language
- Procedural Vs object oriented programming

**Introduction to Java**
- Java Programming Language
- Java History
- Java Platforms
- Java Features
- Java Versions
- JDK ,JRE, JVM, JIT
- What is OpenJDK/AdaptJDK
- Differences between JDK and OpenJDK
- Which one preferred by industry either JDK/OpenJDK

**Java Features/Character Set of Java**
**Comments & Java Tokens – Identifiers, Keywords, Separators**
- Naming rules for Identifiers
- Naming conventions

**Installation of JDK on all operating systems**
- Download JDK and Install JDK
- Windows 32&64bit
- Linux 32&64bit
- Mac 32&64bit
- AIX (optional)
- Environment Variables
- Version Checking

**Structure of Java Program**
- Documentation Section
- Package Statement
- Import statements
- Class definition

**First Java Program**
- Writing
- Saving
- Compilation
- Execution
- Working with JShell

**Literals**
- Numeric Literals
  - Integer, Floating-point
- Character Literals
- Boolean Literals
- String Literals
- null literal

**Variables & Data Types**
- Primitive Data Types
- Reference Types
- Local Variables

**Type Conversions**
- Auto Conversion/Widening/implicit conversion
- Casting/narrowing/explicit conversion

**Operators**
- Arithmetic Operators
- Increment/ Decrement Operators
- Assignment Operators
- Relational Operators
- Negation Operator
- Bitwise Complement Operator
- Logical Operators
- Bitwise Operators
- Shift Operators
- Conditional Operator
- new Operator
- instance of Operator
- Operators Precedence
- Operators Associativity
- Numeric Type Promotions

**Reading Data from Keyboard**
- Using Scanner class
- Getting different Primitive data

**Control Structures**
- Decision Making Statements – if, if..else, nested if, ladder if, switch
- Repetitive Statements – while, do...while, for, labelled loops
- Branching Statements – break, continue, return

**Methods**
- Method Syntax
- Zero Parameters Method
- Methods with Parameters
- Method Returns Void
- Method Returns Value
- Overloading Methods
- In-out implementation

**Arrays**
- Single Dimensional Arrays
- Enhanced for loop
- Multi Dimensional Arrays
- Jagged Arrays

- Variable Arguments (Var-Args)
- Method Passing Arrays
- Method returns Array
- Case Study

**Command-line Arguments**

**Working with Java IDEs**

- Eclipse
- Netbeans IDE
- Intellij IDE
- Creating Project with IDE
- Debugging a Java Program

**Object Oriented Programming**

- Introduction
- Procedure Oriented Vs Object Oriented Approach
- Class & Object
- Java Modifiers
    - Access Modifiers
    - Non Access Modifiers
- Encapsulation
- Getter & Setters
- Instance & Class Variables
- Instance & Class Methods
- Constructors
- Blocks
- Overloading Constructors
- this keyword
- Constructor Vs Method
- Passing & Returning Object from Method
- Inheritance
- Method Overriding
- super keyword
- Super Constructors
- Polymorphism
- Static Binding Vs Dynamic Binding
- Relationships
- Object Cloning – Shallow & Deep
- Design Patterns
- Case Study

**JVM Architecture**

- Java Runtime Environment
- Memory Areas of JVM
- Class Loaders
- Execution Engine – Interpreter & JIT

**Reflection API**

- Reflection in Java
- Classes in java.lang.reflect
- Methods of java.lang.Class
- Getting Complete information of a Java class
- Access Private constructor from outside of a class

**Static member execution flow**

**Non-Static/Instance member execution flow**

**Abstraction**

- Abstract Methods
    - Abstract class, Class Vs Abstract Class
- Interfaces
    - Multiple Inheritance using Interfaces
    - Interface Vs Abstract class
    - default and static methods in Interface
    - public & private static methods in Interface
- Marker Interface
    - Functional Interface
    - Anonymous Class
- Lambda Expressions
- Method reference
- Reference to a static method
- Reference to a instance method
- Reference to a constructor
- Abstract class with constructor and its importance
- Case Study

**Inner Classes**

- Nested Top Level/ Static Inner Class
- Member Classes
- Local Classes
- Anonymous Class

**Packages**

- Need of Packages
    - Built-in Packages
    - rt.jar and Java Modules
    - Preparation of JAR file and How to run it.
    - Uses of JAR files
- User-defined Packages
    - Package naming rules
    - Importing packages
    - Sub-Packages
    - classpath Environment Variable
    - Static Imports

**String Handling**

- String
- StringBuffer
- StringBuilder
- String Vs StringBuffer
- StringBuffer Vs StringBuilder
- StringJoiner
- Immutablitiy
- Creating Immutable class
- StringTokenizer
- String Pool with memory management

**Wrapper Classes**

- Need of Wrapper Classes
- List of Wrapper Classes
- Wrapper Class Methods
- Boxing & UnBoxing
- Auto Widening
- Widening beats Boxing
- Widening beats Varargs
- Boxing beats Varargs

**Exception Handling**

- What is Exception
    - Exception Propagation
    - Hierarchy of Exception classes
    - Exception Types
    - Exception Vs Error
    - Checked Vs Unchecked Exceptions
    - Keywords used for Exception Handling – try, catch, finally, throw, throws
- try with catch
    - try with multiple catch blocks
    - Nested try blocks
    - finally block
- User-defined / Custom Exceptions
    - throw statement
- throws keyword
- throw Vs throws
- Method Overriding with Exception Handling
- "try" with multiple resources
- Adding logic for try block with parameter
- Chained Exceptions
    - Multicatch in Java
    - try with resources

**IO Streams**

- Streams in Java
- Working with jav.io.File
- Byte Streams
- Input Streams & Output Streams
- Character Streams
- Readers & Writers
- Serialization & De-serialization, SerialCersionUID variable

- Externalization
- transient Modifier
- nio package
- Log files creation

**Multithreading with Executor Framework**

- Multitasking
- What is Thread?
- Multithreading
- Main Thread
- Lifecycle of a Thread
- Creating Child Threads
- implements Runnable Vs extends Thread
- Thread Priorities
- Thread class method to pause thread execution
    - yield()
    - sleep()
    - join()
    - wait() & notify()
    - interrupt() to interrupt thread pause
- Daemon Threads
- Thread Synchronization
- Deadlock & Prevention of Deadlock
- Deadlock Vs Starvation
- Inter Thread Communication
- Thread Pooling
- Thread Group
- Concurrent linked queue
- ShutdownHook
- Executor Framework
- CountdownLatch/cyclic barrier
- Semaphore
- Exchanger
- Performing multiple task by multiple thread
- ThreadDump (Only Overview)

**Synchronization :**

- What and Why?
- synchronized method
- synchronized block
- static synchronization
- Inter-thread Communication
- Interrupting Thread

**Garbage Collection**

- How Automatic Garbage Collection Works
- Old Generation , Eden Space
- Mark,Sweep and Compact Operations
- Different Garbage Collectors
- Which Garbage collector we need to use?
- finalize() of java.lang.Object
- Object class methods
- Usage of equals and hash code method
- System.gc()
- Runnable.gc()
- How to increase java heap size ?
- Runtime class

**Enum**

- Enums
- enum in switch
- values() and valueOf( ) method
- Enum field , methods and constructors

**Generics**

- Generics
- Advantage of Java Generics
- Generic Class
- Type Parameters
- Generic Methods
- Wildcard in Generics

**Collections Framework**

- Limitations of Array
- Introduction to Collections
- Collection Interfaces
- List Interfaces
- Set Interfaces
- Queue Interfaces
- List Implementation Classes – ArrayList, Vector, Stack, LinkedList
- Cursors in Collection – Iterator<E>,listIterator<E>,enumeration<E>,splitIterator<T>
- Set Implementations Classes – HashSet, LinkedHashSet, TreeSet,EnumSet
- Queue Implementaion Classes – ArrayDeque, PriorityQueue
- Comparable Interface
- Comparator Interface
- Comparable Vs Comparator
- Map Interface
    - Map Implementation Classes – HashMap, LinkedHashMap, IdentityHashMap, WeakHashMap, Hashtable, TreeMap,EnumMap
- Internal Implementation of HashMap
- Arrays Class
- Concurrent Collection API
    - Traditional Collections Vs Concurrent Collections
- Collections Factory Methods
- Need to override hashCode() and equals()
- Preparation customized linkedlist/ implementation of LinkedList code?
- Differences between HashMap and ConcurrentHashMap

**Java Stream API**

- Advantages of using Stream API
- Stream Interface methods – forEach, map, filter, limit, sorted
- Parallel Processing – Collectors & Statistics
- Predefined Functional Interfaces – Consumer, Function, Predicate
- Optional Class

**Utility Classes**

- StringTokenizer
- Date, Calendar, Currency classes
- Formatting Date and Time
- Locale, Formatter, Random classes
- Timer & TimerTask Classes
- Date/Time API

**Regular Expressions**

- Use of Regular Expression
- java.util.regex package

**Networking**

- Introduction
- InetAddress
- Socket Programming
- Datagrams
- URL & URLConnection

**Annotations**

- Use of Annotations
- Type of Annotations
- Built-in Java Annotations
- Custom Annotations

**GUI Programming**

- AWT
- Swings
- Java FX
- APPLETS

**Java 8 Features:**

- Lambda Expressions
- Marker and Functional Interfaces
- Pipelines and Streams
- Date and Time API

- Default Methods
- Type Annotations
- Nashorn JavaScript Engine
- Concurrent Accumulators
- Parallel operations
- Predicate
- PermGen Space Removed
- TLS SNI

**Java9 Features:**

- The Java Platform module system
- Dynamic Linking
- JShell: the interactive Java REP
- Improved Javadoc
- Collection factory methodsFactory Methods for Immutable List, Set, Map and Map.Entry
- Stream API improvements
- Private interface methods
- HTTP/2
- Multi-release JARs
- Multi-Resolution Image API
- CompletableFuture API Improvements
- Diamond Operator for Anonymous Inner Class
- Miscallenous
    - GC (Garbage Collector) Improvements
    - Stack-Walking API
    - Filter Incoming Serialization Data
    - Deprecate the Applet API
    - Indify String Concatenation
    - Enhanced Method Handles
    - Java Platform Logging API and Service
    - Compact Strings
    - Parser API for Nashorn

**Java10 Features:**

- The Local Variable Type Inference
- Time-Based Release Versioning
- Garbage-Collector Interface
- Parallel Full GC for G1
- Heap Allocation on Alternative Memory Devices
- JDK Forest into a Single Repository
- Unicode Language-Tag Extensions
- Root Certificates
- Thread-Local Handshakes
- Experimental Java-Based JIT Compiler
- Application Class-Data Sharing
- API Changes(Additions and Removals etc)

**Java 11 Features:**

- **New String methods(**isBlank, lines, repeatetc).
- New File methods (Writestring,readstring,isSameFile)
- **Pattern recognizing methods**
- **Epsilon Garbage Collector**
- **Removed the Java EE / CORBA module and Thread Functions**
- **Local-Variable Syntax for Lambda Parameters**
- **TimeUnit Conversion**
- **Optional.isEmpty()**
- **Adapt JDK and Open JDK vs Oracle JDK Changes Overview**

**Java 12 Features:**

**JVM Changes**

- JEP 189 – Shenandoah: A Low-Pause-Time Garbage Collector (Experimental)
- JEP 346 – Promptly Return Unused Committed Memory from G1
- JEP 344 - Abortable Mixed Collections for G1

**Switch Expressions**

- The new Syntax removes the need for break statement to prevent fallthroughs.
- Switch Expressions don't fall through anymore.
- Furthermore, we can define multiple constants in the same label.
- default case is now compulsory in Switch Expressions.
- break is used in Switch Expressions to return values from a case itself.

**File mismatch() Method**

- If the bytes are not identical. In this case, the position of the first mismatching byte is returned.
- File sizes are not identical. In this case, the size of the smaller file is returned.

**Compact Number Formatting**

**Teeing Collectors in Stream API**

- Teeing Collector is the new collector utility introduced in the Streams API.
- This collector has three arguments – Two collectors and a Bi-function.

**Java Strings New Methods**

- indent(),transform()
- Optional describeConstable()
- String resolveConstantDesc()

**JVM Constants API**

- A new package java.lang.constant is introduced with this JEP.

**JEP 305: Pattern Matching for instanceof**

**Raw String Literals is removed from JDK 12.**

**Java 13 Features:**

- Enhancements for Switch Expressions
- Text Blocks instead of Raw String Literals
- Dynamic CDS Archives
- ZGC returns unused storage
- Renewed Socket API