Java

[Java](https://www.simplilearn.com/mobile-and-software-development/java-javaee-soa-development-training) is one of the most common, in-demand computer programming languages in use today. Owned by the Oracle Corporation, this general-purpose programming language with its object-oriented structure has become a standard for applications that can be used regardless of platform (e.g., Mac, Window, Android, iOS, etc.) because of its Write Once, Run Anywhere (WORA) capabilities. As a result of this capability, Java is recognized for its portability across platforms from mainframe data centers to smartphones. Today there are more than 3 billion devices running applications built with Java.

Java is widely used in web and application development as well as Big Data. Java is also used in the backend of several popular websites, including Google, Amazon, Twitter, and YouTube. It is also extensively used in hundreds of applications. New Java frameworks like Spring, Struts, and Hibernate are also very popular. With millions of Java developers worldwide, there are hundreds of texts available for learning Java programming. Also, Java programmers are actively involved in a variety of forums where they vet ideas or solve programming problems. Java has a vast community and gets a lot of support.

Although Java is a more complex language to learn, the trade-off is experienced developers are in demand. The average Java developer earns around [$94,000](https://www.glassdoor.co.in/Salaries/us-java-developer-salary-SRCH_IL.0,2_IN1_KO3,17.htm).

|  |  |
| --- | --- |
| Level: | Intermediate |
| Skills Needed: | Problem-solving, knowledge of the object-oriented structure |
| Platform: | Web, Mobile, Desktop |
| Popularity Among Programmers: | One of the world’s most popular; high demand |
| Benefits: | * Regarded as a good start for learning to think like a programmer * You’ll be able to access/manipulate the most important computer parts like the file system, graphics, and sound for any fairly sophisticated and modern program—that can run on any operating system. |
| Downsides: | Lots of new vocabulary to learn, a higher-level language |
| Degree of Use: | Widely used; highly applicable |
| Annual Salary Projection: | $94,000 |

**Topics of Core Java:**

Core Java contains two sub-topics:

* Java Basics
* OOPs concept

**Java basics**includes:

* Java Features
* [Java Installation](https://intellipaat.com/tutorial/java-tutorial/installation-java/)
* JVM, JRE, JDK
* Language Fundamentals
  + Keywords
  + Variable
  + Identifiers
  + Data Types
  + Literals
  + [Operators](https://intellipaat.com/tutorial/java-tutorial/java-operators/)
* [Java Array](https://intellipaat.com/tutorial/java-tutorial/java-array/)
* [Java String](https://intellipaat.com/tutorial/java-tutorial/java-strings/)
* Control Statements
  + Conditional Control Statements
  + Unconditional Control Statements
  + Jump Statements

**OOPs topics:**

* [Classes and Object](https://intellipaat.com/tutorial/java-tutorial/classes-objects/)
* Access Modifiers
* [Methods and Blocks:](https://intellipaat.com/tutorial/java-tutorial/java-methods/)
  + instance
  + local
  + static
* [Constructors](https://intellipaat.com/tutorial/java-tutorial/java-constructors/)
* [Polymorphism:](https://intellipaat.com/tutorial/java-tutorial/polymorphism/)
  + Method Overloading
  + Method Overriding
* [Inheritance:](https://intellipaat.com/tutorial/java-tutorial/inheritance-is-a/)
  + Single Inheritance
  + Multi-level Inheritance
  + Hierarchical Inheritance
  + Multiple Inheritance(Not allowed in JAVA-check why?)
  + Hybrid Inheritance
* Abstraction:
  + [abstract keyword](https://intellipaat.com/tutorial/java-tutorial/abstract-class-interface/)
  + [Interface keyword](https://intellipaat.com/tutorial/java-tutorial/abstract-class-interface/)
* [Encapsulation:](https://intellipaat.com/tutorial/java-tutorial/encapsulation/)
  + [Packages:](https://intellipaat.com/tutorial/java-tutorial/java-packages/)
    - Java lang package
    - [Java i/o package](https://intellipaat.com/tutorial/java-tutorial/java-file-io/)
    - Java util package
  + Import packages
* [Exception Handling:](https://intellipaat.com/tutorial/java-tutorial/java-exception-handling/)
  + try, catch and finally block
  + throw and throws keyword
* [Multi-threading](https://intellipaat.com/tutorial/java-tutorial/java-multithreading/)

**Advance Java**topics:

* [Collections](https://www.javatpoint.com/collections-in-java)
* [JDBC](https://www.javatpoint.com/java-jdbc)
* JSP
* Applets
* GUI Programming
* [Servlets](https://www.geeksforgeeks.org/introduction-java-servlets/)
* Internationalization
* Reflection
* Networking
* AWT & Events
* Java FX
* Java Swing
* [RMI Apllication](https://www.javatpoint.com/RMI)

**Core Java**

Basics of Java

 Java - What, Where and Why?  
 History and Features of Java  
 Internals of Java Program  
 Difference between JDK,JRE and JVM  
 Internal Details of JVM  
 Variable and Data Type  
 Unicode System  
 Naming Convention

OOPS Concepts

 Advantage of OOPs  
 Object and Class  
 Method Overloading  
 Constructor  
 static variable, method and block  
 this keyword  
 Inheritance (IS-A)  
 Aggregation and Composition(HAS-A)  
 Method Overriding  
 Covariant Return Type  
 super keyword  
 Instance Initializer block  
 final keyword  
 Runtime Polymorphism  
 static and Dynamic binding  
 Abstract class and Interface  
 Downcasting with instanceof operator  
 Package and Access Modifiers  
 Encapsulation  
 Object class  
 Object Cloning  
 Java Array  
 Call By Value and Call By Reference  
 strictfp keyword  
 Creating API Document

String Handling

 String : What and Why?  
 Immutable String  
 String Comparison  
 String Concatenation  
 Substring  
 Methods of String class  
 StringBuffer class  
 StringBuilder class  
 Creating Immutable class  
 toString method  
 StringTokenizer class

Exception Handling

 Exception Handling : What and Why?  
 try and catch block  
 Multiple catch block  
 Nested try  
 finally block  
 throw keyword  
 Exception Propagation  
 throws keyword  
 Exception Handling with Method Overriding  
 Custom Exception

Nested Classes

 Nested Class : What and Why?  
 Member Inner class  
 Annonymous Inner class  
 Local Inner class  
 static nested class  
 Nested Interface

Multithreading

 Multithreading : What and Why?  
 Life Cycle of a Thread  
 Creating Thread  
 Thread Schedular  
 Sleeping a thread  
 Joining a thread  
 Thread Priority  
 Daemon Thread  
 Thread Pooling  
 Thread Group  
 ShutdownHook  
 Performing multiple task by multiple thread  
 Garbage Collection  
 Runnable class

Synchronization

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

Input and output

 FileOutputStream & FileInputStream  
 ByteArrayOutputStream  
 SequenceInputStream  
 BufferedOutputStream & BufferedInputStream  
 FileWriter & FileReader  
 CharArrayWriter  
 Input from keyboard by InputStreamReader  
 Input from keyboard by Console  
 Input from keyboard by Scanner  
 PrintStream class  
 PrintWriter class  
 Compressing and Uncompressing File  
 Reading and Writing data simultaneously  
 DataInputStream and DataOutputStream  
 StreamTokenizer class

Serialization

 Serialization & Deserialization  
 Serialization with IS-A and Has-A  
 transient keyword

Networking

 Socket Programming  
 URL class  
 Displaying data of a web page  
 InetAddress class  
 DatagramSocket and DatagramPacket  
 Two way communication

AWT and EventHandling

 AWT Controls  
 Event Handling by 3 ways  
 Event classes and Listener Interfaces  
 Adapter classes  
 Creating Games and Applications

Swing

 Basics of Swing  
 JButton class  
 JRadioButton class  
 JTextArea class  
 JComboBox class  
 JTable class  
 JColorChooser class  
 JProgressBar class  
 JSlider class  
 Digital Watch  
 Graphics in swing  
 Displaying Image  
 Edit Menu for Notepad  
 Open Dialog Box  
 Creating Notepad  
 Creating Games and applications

LayoutManagers

 BorderLayout  
 GridLayout  
 FlowLayout  
 BoxLayout  
 CardLayout

Applet

 Life Cycle of Applet  
 Graphics in Applet  
 Displaying image in Applet  
 Animation in Applet  
 EventHandling in Applet  
 JApplet class  
 Painting in Applet  
 Digital Clock in Applet  
 Analog Clock in Applet  
 Parameter in Applet  
 Applet Communication  
 Creating Games

Reflection API

 Reflection API  
 newInstance() & Determining the class object  
 javap tool  
 creating javap tool  
 creating appletviewer  
 Accessing private method from outside the class

Collection

 Collection Framework  
 ArrayList class  
 LinkedList class  
 ListIterator interface  
 HashSet class  
 LinkedHashSet class  
 TreeSet class  
 PriorityQueue class  
 ArrayDeque class  
 Map interface  
 HashMap class  
 LinkedHashMap class  
 TreeMap class  
 Hashtable class  
 Comparable and Comparator  
 Properties class

JDBC

 JDBC Drivers  
 Steps to connect to the database  
 Connectivity with Oracle  
 Connectivity with MySQL  
 Connectivity with Access without DSN  
 DriverManager  
 Connection interface  
 Statement interface  
 ResultSet interface  
 PreparedStatement  
 ResultSetMetaData  
 DatabaseMetaData  
 Storing image  
 Retrieving image  
 Storing file  
 Retrieving file  
 Stored procedures and functions  
 Transaction Management  
 Batch Processing  
 JDBC New Features  
 Mini Project

Java New Features

 Assertion  
 For-each loop  
 Varargs  
 Static Import  
 Autoboxing and Unboxing  
 Enum Type  
 Annotation  
etc.

Internationalization

 Internationalization  
 ResourceBundle class  
 I18N with Date  
 I18N with Time  
 I18N with Number  
 I18N with Currency

**Advance Java**

**Servlet**

Basics of Servlet

 Servlet: What and Why?  
 Basics of Web  
 Servlet API  
 Servlet Interface  
 GenericServlet  
 HttpServlet  
 Servlet Life Cycle  
 Working with Apache Tomcat Server  
 Steps to create a servlet in Tomcat  
 How servlet works?  
 servlet in Myeclipse  
 servlet in Eclipse  
 servlet in Netbeans

ServletRequest

 ServletRequest methods  
 Registration example with DB

Servlet Collaboration

 RequestDispatcher  
 sendRedirect

ServletConfig

 ServletConfig methods  
 ServletConfig example

ServletContext

 ServletContext methods  
 ServletContext example

Attribute

 How to set, get and remove example?

Session Tracking

 Cookies  
 Hidden Form Field  
 URL Rewriting  
 HttpSession

Event and Listener

Filter

 Filter  
 Authentication Filter  
 FilterConfig  
 Useful examples

ServletInputStream and ServletOutputStream

Annotation Servlet

Project Development

**JSP**

Basics of JSP

 Life cycle of JSP  
 JSP API  
 JSP in Eclipse and other IDE's

Scripting elements

 scriptlet tag  
 expression tag  
 declaration tag

9 Implicit Objects

 out  
 request  
 response  
 config  
 application  
 session  
 pageContext  
 page  
 exception

Directive Elements

 page directive  
 include directive  
 taglib directive

Exception Handling

Action Elements

 jsp:forward  
 jsp:include  
 Bean class  
 jsp:useBean  
 jsp:setProperty & jsp:getProperty  
 Displaying applet in JSP

Expression Language

MVC in JSP

JSTL

Custom tags

 Custom Tag : What and Why?  
 Custom Tag API?  
 Custom Tag Example  
 Attributes  
 Iteration  
 Custom URI

Project Development in JSP

.

**JavaMail API**

JavaMail API

 Sending Email  
 Sending email through Gmail server  
 Receiving Email  
 Sending HTML content

**Design Pattern**

Design Pattern

 Singleton  
 DAO  
 DTO  
 MVC  
 Front Controller  
 Factory Method  
 etc.

**JUnit**

JUnit

 JUnit: What and Why?  
 Types of Testing  
 Annotations used in JUnit  
 Assert class  
 Test Cases

**Maven**

Maven

 Maven: What and Why?  
 Ant Vs Maven  
 How to install Maven?  
 Maven Repository  
 Understanding pom.xml  
 Maven Example  
 Maven Web App Example  
 Maven using Eclipse

**Struts 2**

Basics of Struts2

 Struts : What and Why?  
 Model1 vs Model2  
 Struts2 Features  
 Steps to create Struts2 application  
 Understanding Action class  
 Understanding struts.xml file  
 Struts2 in Eclipse IDE  
 Struts2 in Myeclipse IDE

Core Components

 Interceptors  
 ValueStack  
 ActionContext  
 ActionInvocation  
 OGNL

Struts 2 Architecture

Struts2 Action

 Action Interface  
 ActionSupport class

Struts2 Configuration

 multi configuration  
 multi namespace

Interceptors

 Custom Interceptor : Life Cycle of Interceptor  
 params interceptor  
 execAndWait interceptor  
 prepare interceptor  
 modelDriven interceptor  
 exception interceptor  
 fileUpload interceptor

Struts 2 Validation

 Custom Validation : workflow interceptor  
 Input Validation : validation interceptor  
 Ajax Validation : jsonValidation interceptor

Aware Interfaces

 ServletActionContext  
 SessionAware  
 Login and Logout Application  
 ServletContextAware

Struts2 with I18N

Zero Configuration

 By convention  
 By annotation

Struts2 with Tiles2

Hibernate with Struts2

Spring with Struts2

Project Development in Struts2

**Working with IDE**

IDE

 Eclipse IDE  
 Netbeans IDE  
 Myeclipse IDE

**Working with Servers**

Servers

 Apache Tomcat  
 Glassfish Server  
 JBoss Server  
 Weblogic Server

**EJB 3**

Introduction to JavaEE

 The Need for JavaEE.  
 Overview on the JavaEE Architecture  
1 tier  
2 tier  
3 tier  
N tier  
 JavaEE Key Standards

Introduction to EJB3

 The EJB Model  
 Key Services of the Application Server

Developing Session Beans

 Stateless Session Beans  
 Stateful Session Beans  
 Packaging  
 Writing Clients

Using Dependency Injection

 No More JNDI  
 Injection of EJBContext

JMS

 JMS Overview  
 JMS Messaging Domains  
 Example of JMS using Queue  
 Example of JMS using Topic

Message Driven Beans

Persistence Introduction to JPA

 Object Relational Mapping  
 Mapping configurations  
 Embedded Objects  
 EJBQL

**Web Technology**

HTML 5

 Introduction to HTML  
 HTML Tags  
 Creating Forms  
 Creating tables  
 Managing home page

CSS

 Introduction to CSS  
 Three ways to use CSS  
 CSS Properties  
 Designing website  
 Working with Templates

Javascript

 Introduction to Javascript  
 Three ways to use Javascript  
 Working with events  
 Client-side Validation

JQuery

 Introduction to JQuery  
 Validation using JQuery  
 JQuery Forms  
 JQuery Examples

AJAX

 Introduction to AJAX  
 Servlet and JSP with AJAX  
 Interacting with database

A lot of it depends on your interest. You can:

* **Java 8** - A foray into functional paradigm. It is a different style of coding that can enhance your coding skills. Java 8 in action (Manning publication) is a good book to start on Java 8.
* pick up **Data structure and Algorithms in Java**. The book will cover all the basics data structures. As you are going to code all data structure and algorithms in Java, it will improve your Java skills as well. Knowledge of Data Structures will help you in long run.
* **Design patterns** - Another popular topic that can improve you coding skills. Read about various popular design patterns. You can pick up Head First Design Pattern.
* **Java FX** - If you are interested in desktop applications that can run on android and ios, you can also start with Java FX. [Getting Started with JavaFX](http://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm) - Oracle's tutorial on Java FX.
* **Web Frameworks** - If you like creating web sites you can start with Jsp/Servlets. Jsp/Servlets will introduce you to the concepts of MVC architecture. Once you understand MVC architecture you can start with any of the below popular web framework.
  + *Spring MVC* - Spring provides you end to end solution. The framework is well documented and widely used. It has a huge community. Along with MVC architecture it comes along with other features such as DI/IOC, AOP, DAO support etc. There are a lot of tutorials available that can help you getting started.
  + *Play* - It is another framework which is lightweight, fast and reliable. Play framework supports scala as well. So tomorrow is you decide to foray into functional programming, you only need to learn Scala syntax.
  + Other popular ones are Struts and JSF.
* If perspective is towards job/personal skill improvement,
  + I would recommend doing an AWS certification.
  + You can implement advanced projects. I have listed some [here](https://www.quora.com/What-Java-projects-can-I-make-on-my-own-so-that-I-can-have-better-learning-of-the-language-assuming-my-Java-level-is-a-beginner).
  + You can learn advanced frameworks like Netty, Lucene etc..
  + You can implement distributed systems and multithreaded high performance architectures.
  + You can contribute to open source projects.
  + You can look at scala, as it would appreciate pithy code and concise code descriptions.
* Develop a web-based application using Servlets & JSPs.
* Process XML using Java APIs.
* Develop web applications using Struts 2.x framework.
* Skill to use a pre-built framework for rapid application development.
* Strong foundation for applying design patterns like MVC.