Java Programming

Description

Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on desktops and servers. Java offers the rich user interface, performance, versatility, portability, and security that today's applications require.

This Java SE 8 Programming training covers the core language features and Application Programming Interfaces (API) you will use to design object-oriented applications with Java Standard Edition 8 (Java SE 8) Platform.

Delegates will learn to

- Create Java technology applications with the latest JDK Technology
- Develop your object-oriented skills
- Identify good practices in the use of the language to create robust Java application
- Use Lambda expressions in Java applications
- Store and manipulate data using collections
- Manipulate files, directories and file systems
- Connect to databases using standard SQL queries through JDBC
- Create high-performance multi-threaded applications

Outline

Java Platform Overview

- Defining how the Java language achieves platform independence
- Defining how the Java language continues to evolve
- Differentiating between the Java ME, Java SE, and Java EE Platforms
- Evaluating Java libraries, middle-ware, and database options

Java Syntax and Class Review

- Using operators
- Creating primitive variables
- Creating simple Java classes
- Using if-else and switch statements
- Creating and manipulate strings
- Iterating with loops: while,do-while,for,enhanced for
- Creating arrays
- Using Java fields, constructors, and methods

Encapsulation and Subclassing

- Creating and use Java subclasses
- Using encapsulation in Java class design
- Overloading methods
- Making classes immutable
- Modeling business problems using Java classes

Overriding Methods, Polymorphism, and Static Classes

- Using the instanceof operator to compare object types
- Using varargs to specify variable arguments
- Using access levels: private, protected, default, and public.
- Overriding methods
- Implementing the singleton design pattern
- Modeling business problems by using the static keyword
- Using upward and downward casts
- Using virtual method invocation

Abstract and Nested Classes

- Constructing abstract Java classes and subclasses
- Designing general-purpose base classes by using abstract classes

- Applying final keyword in Java
- Distinguish between top-level and nested classes

Interfaces and Lambda Expressions

- Defaulting methods
- Defining a Java interface
- Anonymous inner classes
- Defining a Lambda Expression
- Extending an interface
- Choosing between interface inheritance and class inheritance

Collections and Generics

- Creating a collection by using generics
- Implementing an ArrayList
- Implementing a HashMap
- Implementing a Deque
- Using the type inference diamond to create an object
- Creating a custom generic class
- Implementing a TreeSet
- Ordering collections

Collections Streams, and Filters

- Describing the Stream interface
- Calling an existing method using a method reference
- Chaining multiple methods together
- Filtering a collection using lambda expressions
- Defining pipelines in terms of lambdas and collections
- Iterating through a collection using lambda syntax
- Describing the Builder pattern

Lambda Built-in Functional Interfaces

- Using primitive versions of base interfaces
- Using binary versions of base interfaces
- Listing the built-in interfaces included in java.util.function
- Core interfaces – Predicate, Consumer, Function, Supplier

Lambda Operations

- Extracting data from an object using map
- Describing the Optional class
- Sorting a stream
- Describing lazy processing
- Describing the types of stream operations
- Grouping and partition data using the Collectors class
- Saving results to a collection using the collect method

Exceptions and Assertions

- Recognizing common exception classes and categories
- Autoclose resources with a try-with-resources statement
- Creating custom exceptions
- Using the catch, multi-catch, and finally clauses
- Defining the purpose of Java exceptions
- Using the try and throw statements
- Testing invariants by using assertions

Java Date/Time API

- Working with dates and times across time zones
- Combining date and time into a single object
- Creating and manage date-based events
- Applying formatting to local and zoned dates and times
- Creating and manage time-based events
- Managing changes resulting from daylight savings
- Defining and create timestamps, periods and durations

I/O Fundamentals

- Writing and read objects using serialization
- Using streams to read and write files
- Read and write data from the console
- Describing the basics of input and output in Java

File I/O (NIO.2)

- Using the Files class to check, delete, copy, or move a file or directory
- Using the Path interface to operate on file and directory paths

- Using Stream API with NIO2

Concurrency

- Creating worker threads using Runnable and Callable
- Describing operating system task scheduling
- Using an ExecutorService to concurrently execute tasks
- Using the java.util.concurrent collections
- Identifying potential threading problems
- Using monitor locks to control the order of thread execution
- Using synchronized and concurrent atomic to manage atomicity

The Fork-Join Framework

- The need for Fork-Join
- RecursiveTask
- Work stealing
- Parallelism

Parallel Streams

- Listing the key performance considerations for parallel streams
- Reviewing the key characteristics of streams
- Defining reduction
- List the key assumptions needed to use a parallel pipeline
- Calculating a value using reduce
- Describing the process for decomposing and then merging work
- Describing how to make a stream pipeline execute in parallel
- Describing why reduction requires an associative function

Database Applications with JDBC

- Connecting to a database by using a JDBC driver
- Specifying JDBC driver information externally
- Performing CRUD operations using the JDBC API
- Defining the layout of the JDBC API
- Submitting queries and get results from the database

Localization

- Building a resource bundle for each locale

- Describing the advantages of localizing an application
- Calling a resource bundle from an application
- Changing the locale for a resource bundle
- Defining what a locale represents
- Read and set the locale by using the Locale object

Prerequisites

Completing Java Essentials course.