Java8P2 - *Java SE 8 Programmer II*

DURATION: 5 Days; Instructor-led

WHAT YOU WILL LEARN
From developing applications to creating web applications to powering over 3 billion devices across the world, the Java programming language is ever present in today's world. When you become a student of Java – and eventually, an Oracle Certified expert – you create limitless opportunities for exciting jobs in the technology industry.

Learn how to create secure, portable, high-performance applications using the world's #1 programming language. This course offers expert-led courses for beginner to advanced Java developers, covering core concepts, such as language constructs and data types, to intermediate and advanced concepts, such as modular programming, secure coding, and convenience methods.

This is a second-level course for programmers learning the Java language. It rounds-out the topics that were taught in the previous course, Java SE: Programming I, and provides additional key skills for Java programmers, such as secure coding, modular programming, and database access. This course uses today's main Java version for its practices

AUDIENCE
This course is aimed at those already have basic java programming kill and knowledge, especially Data Scientist, Developer, and implementer.

PREREQUISITES
Participants should equipped with the basic java and knowledge as in Java SE8 Programmer I.

OBJECTIVES
- Create Java applications that leverage the object-oriented features of the Java language, such as encapsulation, inheritance, and polymorphism
- Create applications that use the Java Collections framework
- Search and filter collections using Lambda Expressions
- Implement error-handling techniques using exception handling
- Implement input/output (I/O) functionality to read from and write to data and text files
- Manipulate files, directories and file systems using the JDK NIO.2 specification
- Perform multiple operations on database tables, including creating, reading, updating and deleting using both JDBC and JPA technology
- Use Lambda Expression concurrency features
- Apply modular programming practices and services to applications
- Create high-performing multi-threaded applications

Development Tools
OS: MS Windows platform with JDK 8
IDE: NetBeans/Eclipse

METHODOLOGY
This program will be conducted with interactive lectures, PowerPoint presentation, discussions and practical exercise

COURSE OUTLINES

Module 1: *Java Class Design*
- Implement encapsulation
- Implement inheritance including visibility modifiers and composition
- Implement polymorphism
- Override hasCode, equals, and toString methods from Object class
- Create and use singleton classes and immutable classes
- Develop code that uses the static keyword on initialize blocks, variables, methods, and classes

Module 2: *Advanced Class Design*
- Develop code that uses abstract classes and methods
- Develop code that uses the final
- Create inner classes including static inner classes, local classes, nested classes, and anonymous innter classes
- Use enumerated types including methods, and constructors in an enum type
- Develop code that declares, implements and/or extends interfaces and use the @Override annotation.
- Create and use lambda expressions

Module 3: *Generics and Collections*
- Generics and Collections
- Create and use ArrayList, TreeSet, TreeMap, and ArrayDeque objects
- Use java.util.Comparator and java.lang.Comparable interfaces
- Collections, streams, and filters
- Iterate using forEach methods of Streams and List
- Describe the Stream interface and the Stream pipeline
- Filter a collection by using lambda expressions
- Use method references with streams

Module 4: *Lambda Built-In Functional Interfaces*
- Use the built-in interfaces included in the java.util.function package such as Predicate, Consumer, Function, and Supplier
- Develop code that uses primitive versions of functional interfaces
- Develop code that uses binary versions of functional interfaces
- Develop code that uses the UnaryOperator interface

Module 5: *Java Stream API*

- Develop code to extract data from an object using peek() and map() methods including primitive versions of the map() method
- Search for data by using search methods of the Stream classes including findFirst, findAny, anyMatch, allMatch, noneMatch
- Develop code that uses the Optional class
- Develop code that uses Stream data methods and calculation methods
- Sort a collection using Stream API
- Save results to a collection using the collect method and group/partition data using the Collectors class
- Use flatMap() methods in the Stream API

Module 6: *Exceptions and Assertions*

- Use try-catch and throws statements
- Use catch, multi-catch, and finally clauses
- Use autoclose resources with a try-with-resources statement
- Create custom exceptions and autocloseable resources
- Test invariants by using assertions

Module 7: *Working with Inheritance*

- Describe inheritance and its benefits
- Develop code that makes use of polymorphism; develop code that overrides methods; differentiate between the type of a reference and the type of an object
- Determine when casting is necessary
- Use super and this to access objects and constructors
- Use abstract classes and interfaces

Module 8: *Use Java SE 8 Date/Time API*

- Create and manage date- and time-based events, including a combination of date and time in a single object, by using LocalDate, LocalTime, LocalDateTime, Instant, Period, and Duration
- Work with dates and times across time zones and manage changes resulting from daylight savings, including Format date and times values
- Define, create, and manage date - and time - based events using Instant, Period, Duration, and TemporalUnit

Module 9: *Java I/O Fundamentals*

- Read and write data from the console
- Use BufferedReader, BufferedWriter, File, FileReader, FileWriter, FileInputStream, FileOutputStream, ObjectOutputStream, ObjectInputStream, and PrintWriter in the java.io package

Module 10: *Java File I/O (NIO.2)*

- Use the Path interface to operate on file and directory paths
- Use the Files class to check, read, delete, copy, move, and manage metadata a file or directory
- Use Stream API with NIO.2

Module 11: *Object-Oriented Design Principles*

- Write code that declares, implements and/or extends interfaces
- Choose between interface inheritance and class inheritance
- Develop code that implements "is-a" and/or "has-a" relationships
- Apply object composition principles
- Design a class using the Singleton design pattern
- Write code to implement the DAO pattern
- Design and create objects using a factory, and use factories from the API

Module 12: *String Processing*

- Search, parse and build strings
- Search, parse, and replace strings by using regular expressions
- Use string formatting

Module 13: *Concurrency*

- Create worker threads using Runnable, Callable and use an ExecutorService to concurrently execute tasks
- Identify potential threading problems among deadlock, starvation, livelock, and race conditions
- Use synchronized keyword and java.util.concurrent.atomic package to control the order of thread execution
- Use java.util.concurrent collections and classes including CyclicBarrier and CopyOnWriteArrayList.
- Use parallel Fork/Join Framework
- Use parallel Streams including reduction, decomposition, merging processes, pipelines and performance

Module 14: *Building Database Applicatons with JDBC*

- Describe the interfaces that make up the core of the JDBC API including the Driver, Connection, Statement, and ResultSet interfaces and their relationship to provider implementations
- Identify the components required to connect to a database using the DriverManager class including the JDBC URL
- Submit queries and read results from the database including creating statements, returning result sets, iterating through the results, and properly closing result sets, statements, and connections

Module 15: *Localization*

- Read and set the locale by using the Locale object
- Create and read a Properties file
- Build a resource bundle for each locale and load a resource bundle in an application