JAVA -SE-8- Programming

Course Duration 40 Hours

Why Java ?

- Oracle's Java Programmer certification is critical to maximizing your potential and moving your Java career forward.
- Over 30, 000 Java Developers per year become certified on Java.
- More than 800,000 of these Java developers are using the knowledge, skills and recognition learned through Java Certifications to improve their job prospects, earn more money* and become more valuable employees.
- With 3 billion devices running Java worldwide, the world's top companies rely on experienced Java Programmers to build and maintain these critical applications.
- Java is the number one development platform.
- 97% of desktops run Java.
- Java is the number one language for enterprise development.
- Java Programmers Are In Demand

Because of the prevalence of Java, there is a continued demand for well-trained, highly-skilled Java programmers to create and maintain critical applications.

Student benefits

1) Oracle License ekit *
2) 25% discount on certification exam (1z0-809) *
3) Course Completion Certificate from Oracle University *

What you will learn

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.

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

Benefits to You

You can use this course to further develop your skills with the Java language and prepare for the Oracle Certified Professional, Java SE 8 Programmer Exam!

Audience

- Developer
- Java Developers
- Java EE Developers

Required Prerequisites

- Java SE 8 Fundamentals

Course Objectives

- Creating high-performing multi-threaded applications
- Creating Java technology applications that leverage the object-oriented features of the Java language, such as encapsulation, inheritance, and polymorphism
- Implementing input/output (I/O) functionality to read from and write to data and text files and understand advanced I/O streams
- Executing a Java technology application from the command line
- Manipulating files, directories and file systems using the JDK NIO.2 specification
- Creating applications that use the Java Collections framework
- Performing multiple operations on database tables, including creating, reading, updating and deleting using both JDBC and JPA technology
- Searching and filter collections using Lambda Expressions
- Implementing error-handling techniques using exception handling
- Using Lambda Expression concurrency features

Course Topics

Java Platform Overview

Defining how the Java language achieves platform independence

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

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

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

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

Abstract and Nested Classes
Designing general-purpose base classes by using abstract classes
Constructing abstract Java classes and subclasses
Applying final keyword in Java
Distinguish between top-level and nested classes

Interfaces and Lambda Expressions

Defining a Java interface

Choosing between interface inheritance and class inheritance

Extending an interface

Defaulting methods

Anonymous inner classes

Defining a Lambda Expression

Collections and Generics

Creating a custom generic class

Using the type inference diamond to create an object

Creating a collection by using generics

Implementing an ArrayList

Implementing a TreeSet

Implementing a HashMap

Implementing a Deque

Ordering collections

Collections Streams, and Filters

Describing the Builder pattern

Iterating through a collection using lambda syntax

Describing the Stream interface

Filtering a collection using lambda expressions

Calling an existing method using a method reference

Chaining multiple methods together

Defining pipelines in terms of lambdas and collections

Lambda Built-in Functional Interfaces

Listing the built-in interfaces included in java.util.function

Core interfaces - Predicate, Consumer, Function, Supplier

Using primitive versions of base interfaces

Using binary versions of base interfaces

Lambda Operations

Extracting data from an object using map

Describing the types of stream operations

Describing the Optional class

Describing lazy processing

Sorting a stream
Saving results to a collection using the collect method
Grouping and partition data using the Collectors class

Exceptions and Assertions
Defining the purpose of Java exceptions
Copyright © 2013, Oracle. All rights reserved. Page 3Using the try and throw statements
Using the catch, multi-catch, and finally clauses
Autoclose resources with a try-with-resources statement
Recognizing common exception classes and categories
Creating custom exceptions
Testing invariants by using assertions

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

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

File I/O (NIO.2)
Using the Path interface to operate on file and directory paths
Using the Files class to check, delete, copy, or move a file or directory
Using Stream API with NIO2

Concurrency
Describing operating system task scheduling
Creating worker threads using Runnable and Callable
Using an ExecutorService to concurrently execute tasks

Identifying potential threading problems
Using synchronized and concurrent atomic to manage atomicity
Using monitor locks to control the order of thread execution
Using the java.util.concurrent collections

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

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

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

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

Exam details :

Exam Code : 1z0-809

Certification : **Oracle Certified Professional, Java SE 8 Programmer**

Exam duration : 150 minutes

Questions : 85

Passing Score : 65%

Format : Multiple Choice