Szkolenie: Oracle
Java SE 8 Programming

<table>
<thead>
<tr>
<th>FORMA SZKOLENIA</th>
<th>MATERIAŁY SZKOLENIOWE</th>
<th>CENA</th>
<th>CZAS TRWANIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stacjonarne</td>
<td>Cyfrowe</td>
<td>5000 PLN NETTO*</td>
<td>5 dni</td>
</tr>
<tr>
<td>Stacjonarne</td>
<td>Tablet CTAB</td>
<td>5600 PLN NETTO*</td>
<td>5 dni</td>
</tr>
<tr>
<td>Metoda dlearning</td>
<td>Cyfrowe</td>
<td>5000 PLN NETTO*</td>
<td>5 dni</td>
</tr>
<tr>
<td>Metoda dlearning</td>
<td>Tablet CTAB</td>
<td>5000 PLN NETTO*</td>
<td>5 dni</td>
</tr>
</tbody>
</table>

* (+VAT zgodnie z obowiązującą stawką w dniu wystawienia faktury)

LOKALIZACJE
Kraków - ul. Tatarska 5, II piętro, godz. 9:00 - 16:00
Warszawa - ul. Bielska 17, godz. 9:00 - 16:00

DOSTĘPNE TERMINY
2020-03-23 | 5 dni | Kraków  (Promocja)

Cel szkolenia:

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

Plan szkolenia:

- 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
○ Using 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

Wymagania:

Recommended Related Training Courses:

- Java EE 6: Develop Database Applications with JPA
- Java EE 6: Develop Business Components with JMS & EJBs
- Java EE 6: Develop Web Applications with JSF
- Java EE 6: Develop Web Components with Servlets & JSPs
- Java EE 6: Develop Web Services with JAX-WS & JAX-RS

Poziom trudności

[ ] [ ] [ ] [ ] [ ]
Certyfikaty:

Uczestnicy szkoleń otrzymają zaświadczenia o ukończeniu kursu sygnowane przez firmę Oracle.

Prowadzący:

Autoryzowany wykładowca Oracle.