

# Fernkurs-Dokumentation



**Zentrum für Informatik ZFI AG**

**Fernkurs: Upgrade Oracle Certified**

**Professional, Java SE 7 Programmer, 1Z0-805**

**(FOCU)**

<http://www.zfi.ch/FOCU>

Weitere Infos finden Sie unter [www.zfi.ch](http://www.zfi.ch) oder via Adresse:

**Zentrum für Informatik ZFI AG  
Zentralsekretariat  
Rütistrasse 28  
CH-8952 Zürich-Schlieren  
Telefon: 044 732 40 00  
Telefax: 041 530 31 68**

**Zürich, Basel, Bern, Zürich, Schweiz**

<b>Titel</b>	<b>Fernkurs: Upgrade Oracle Certified Professional, Java SE 7 Programmer, 1Z0-8</b>
<b>Untertitel</b>	<b>Der Vorbereitungslehrgang auf die offizielle Oracle Certified Professional, Java SE 7 Programmer 1Z0-805 Upgrade Prüfung als Fernkurs.</b>
<b>Einleitung</b>	<p>Die Oracle Certified Professional (OCP), Java SE 7 Programmer Zertifizierung richtet sich an Software-Entwickler, welche ihre Kenntnisse der Java Programmiersprache attestiert haben möchten.</p> <p>Bereits zertifizierte Java Entwickler können mit dieser Prüfung (1Z0-805) auf die Java SE 7 Version upgraden.</p> <p>Mit der Variante Fernkurs erarbeiten Sie sich die theoretischen Grundlagen der neuen Java SE 7 Features selbständig.</p>
<b>Ihr Nutzen</b>	
<b>Voraussetzungen</b>	
<b>Teilnehmerkreis</b>	<b>Dieser ZFI-Lehrgang richtet sich an bereits zertifizierte Java Programmierer (z.B. Java 5 oder 6 oder einer früheren Version).</b>
<b>Unterlagen</b>	<ul style="list-style-type: none"><li>- Begleitbuch</li><li>- Tutorials</li><li>- Intranet Site</li></ul>
<b>Folgekurse</b>	
<b>Inhalt</b>	<p><b>Upgrade Oracle Certified Professional, Java SE 7 Programmer</b></p> <ul style="list-style-type: none"><li>- Language Enhancements</li><li>- Develop code that uses String objects in switch statements</li><li>- Develop code that uses binary literals and numeric literals with underscores</li><li>- Develop code that uses try-with-resources statements (including using classes that implement the AutoCloseable interface)</li><li>- Develop code that handles multiple Exception types in a single catch block</li><li>- Develop code that uses the diamond with generic declarations</li><li>- Design Patterns</li><li>- Design a class using a Singleton design pattern</li><li>- Apply object composition principles (including has-a relationships)</li></ul>

- Write code to implement the Data Access Object (DAO) pattern
- Design and create objects using a factory pattern
  
- Database Applications 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)
- Construct and use RowSet objects using the RowSetProvider class and the RowSetFactory interface
- Use JDBC transactions (including disabling auto-commit mode, committing and rolling back transactions, and setting and rolling back to savepoints)
- 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)
- Create and use PreparedStatement and CallableStatement objects
  
- Concurrency
  
- Identify code that may not execute correctly in a multi-threaded environment.
- Use collections from the java.util.concurrent package with a focus on the advantages over and differences from the traditional java.util collections.
- Use Lock, ReadWriteLock, and ReentrantLock classes in the java.util.concurrent.locks package to support lock-free thread-safe programming on single variables.
- Use Executor, ExecutorService, Executors, Callable, and Future to

**execute tasks using thread pools.**

- Use the parallel Fork/Join Framework
  
- Localization
  
- Describe the advantages of localizing an application
- Define a locale using language and country codes
- Read and set the locale with a Locale object
- Build a resource bundle for a locale
- Call a resource bundle from an application
  
- Format dates, numbers, and currency values for localization with the NumberFormat and DateFormat classes (including number format patterns)
  
  
- Java File I/O (NIO.2)
  
- Operate on file and directory paths with the Path class
- Check, delete, copy, or move a file or directory with the Files class
- Read and change file and directory attributes, focusing on the BasicFileAttributes, DosFileAttributes, and PosixFileAttributes interfaces
- Recursively access a directory tree using the DirectoryStream and FileVisitor interfaces
- Find a file with the PathMatcher interface
- Watch a directory for changes with the WatchService interface

**Beitrag**

**Der Teilnehmerbeitrag versteht sich rein netto. Das ZFI ist (gemäss MwSt-Gesetz) nicht Mehrwertsteuerpflichtig und erhebt somit keine MwSt. Bei länger als einen Monat dauernden Lehrgängen ist die Zahlung des Teilnehmerbeitrages in mehreren Raten möglich (pro rata temporis).**