

Lehrgang-Dokumentation



Zentrum für Informatik ZFI AG

Oracle Certified Professional Java Programmer SE 7 (OCPJP) (OCJP-0112) -IT Ausbildung nach Mass

<http://www.zfi.ch/OCJP-0112>

Weitere Infos finden Sie unter www.zfi.ch oder via Adresse:

Zentrum für Informatik ZFI AG
Zentralsekretariat
Technoparkstrasse 1
CH-8005 Zürich
Telefon: 044 732 40 00
Telefax: 044 732 40 09

Zürich, Basel, Bern, Zürich, Schweiz

Titel	Oracle Certified Professional Java Programmer SE 7 (OCPJP)
Untertitel	Der Vorbereitungslehrgang auf die offizielle Oracle Java Programmer-Prüfungen Java SE 7
Einleitung	<p>Die Oracle Certified Java Programmer Standard Edition 7 Zertifizierung richtet sich an Software-Entwickler, welche ihre Kenntnisse der Java Programmiersprache attestiert haben möchten. Das Bestehen dieser Prüfungen [Java Programmer I (1Z0-803) und Java Programmer II (1Z0-804)] bestätigt, dass der Programmierer den Syntax und die Struktur der Java-Programmiersprache versteht und mit Java Applikationen entwickeln kann, die auf Server- oder Desktop-Systemen mit Java SE 7 laufen.</p> <p>Während den Präsenztagen werden die theoretischen Grundlagen anhand von vielen Beispielen aufgezeigt. Die wichtigsten Fakten werden live in einem Storyboard aufgeschrieben und sind ein wichtiger Teil der Prüfungsvorbereitung. Zwischen den Präsenztagen sind Programmierübungen zu den behandelten Themen durch die Teilnehmer zu lösen.</p> <p>Als Entwicklungsumgebung setzen wird die Eclipse Java IDE ein.</p> <p>Die aktuellen Java Programmer Zertifizierungen von Oracle:</p> <ul style="list-style-type: none"> Java Standard Edition 6 Programmer Certified Professional Exam (1Z0-851) Java SE 7 Programmer I (1Z0_803) Java SE 7 Programmer II (1Z0_804) Upgrade to Java SE 7 Programmer (1Z0_805) <p>Falls Sie nur den Java Programmer I Teil (5-6 Tage) absolvieren möchten, so werden Ihnen nur diese Kurstage verrechnet oder je nach Fall zurückbezahlt.</p>
Ihr Nutzen	Dieser ZFI-Lehrgang richtet sich an Personen, welche die Sprache Java systematisch lernen möchten, um anschliessend Java-Applikationen in ihrem beruflichen Umfeld zu entwickeln und zu warten. Lernziel ist selbstverständlich das Bestehen der von Oracle durchgeführten Prüfungen zum Oracle Certified Professional Java Programmer I (1Z0_803) und II (1Z0_804).
Voraussetzungen	
Teilnehmerkreis	Dieser ZFI-Lehrgang richtet sich an Programmierer, welche bereits erste Schritte in Java gemacht haben oder bereits in einer anderen Programmiersprache entwickelt haben und die Grundlagen der objektorientierten Software-Entwicklung bereits kennen.
Unterlagen	<ul style="list-style-type: none"> - Begleitbuch - Tutorials - Intranet Site
Folgekurse	
Inhalt	Java Programmer I

- **Java Basics**
 - Define the scope of variables
 - Define the structure of a Java class
 - Create executable Java applications with a main method
 - Import other Java packages to make them accessible in your code

- **Working With Java Data Types**
 - Declare and initialize variables
 - Differentiate between object reference variables and primitive variables
 - Read or write to object fields
 - Explain an object's lifecycle
 - Manipulate data using the `StringBuilder` class and its methods
 - Create and manipulate strings

- **Using Operators and Decision Constructs**
 - Use Java operators
 - Use parentheses to override operator precedence
 - Test equality between strings and other objects using `==` and `equals ()`
 - Create `if` and `if/else` constructs
 - Use a `switch` statement

- **Creating and Using Arrays**
 - **Declare, instantiate, initialize and use a one-dimensional array**
 - **Declare, instantiate, initialize and use multi-dimensional array**
 - **Declare and use an ArrayList**

- **Using Loop Constructs**
 - **Create and use while loops**
 - **Create and use for loops including the enhanced for loop**
 - **Create and use do/while loops**
 - **Compare loop constructs**
 - **Use break and continue**

- **Working with Methods and Encapsulation**
 - **Create methods with arguments and return values**
 - **Apply the static keyword to methods and fields**
 - **Create an overloaded method**
 - **Differentiate between default and user-defined constructors**
 - **Create and overload constructors**
 - **Apply access modifiers**
 - **Apply encapsulation principles to a class**
 - **Determine the effect upon object references and primitive values when they are passed into methods that change the values**

- **Working with Inheritance**
- **Implement inheritance**
- **Develop code that demonstrates the use of polymorphism**
- **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**

- **Handling Exceptions**

- **Differentiate among checked exceptions, RuntimeExceptions and Errors**
- **Create a try-catch block and determine how exceptions alter normal program flow**
- **Describe what exceptions are used for in Java**
- **Invoke a method that throws an exception**
- **Recognize common exception classes and categories**

Java Programmer II

- **Java Class Design**
- **Use access modifiers: private, protected, and public**

- **Override methods**
- **Overload constructors and other methods appropriately**
- **Use the instanceof operator and casting**
- **Use virtual method invocation**
- **Override methods from the Object class to improve the functionality of your class**
- **Use package and import statements**

- **Advanced Class Design**
- **Identify when and how to apply abstract classes**
- **Construct abstract Java classes and subclasses**
- **Use the static and final keywords**
- **Create top-level and nested classes**
- **Use enumerated types**

- **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**

- **Generics and Collections**
- **Create a generic class**
- **Use the diamond syntax to create a collection**
- **Analyze the interoperability of collections that use raw type and generic types**
- **Use wrapper classes and autoboxing**
- **Create and use a List, a Set and a Deque**
- **Create and use a Map**
- **Use `java.util.Comparator` and `java.lang.Comparable`**
- **Sort and search arrays and lists**

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

- **Exceptions and Assertions**
- **Use `throw` and `throws` statements**
- **Use the `try` statement with multi-catch, and finally clauses**
- **Autoclose resources with a `try-with-resources` statement**

- Create custom exceptions
- Test invariants by using assertions

- Java I/O Fundamentals

- Read and write data from the console
- Use streams to read and write files

- Java File I/O (NIO.2)

- Use the Path class to operate on file and directory paths
- Use the Files class to check, delete, copy, or move a file or directory
- Read and change file and directory attributes
- Recursively access a directory tree
- Find a file by using the PathMatcher class
- Watch a directory for changes by using WatchService

- Building Database Applications with JDBC

- Define the layout of the JDBC API
- Connect to a database by using a JDBC driver
- Update and query a database
- Customize the transaction behavior of JDBC and commit transactions
- Use the JDBC 4.1 RowSetProvider, RowSetFactory, and RowSet interfaces

- **Threads**
- **Create and use the Thread class and the Runnable interface**
- **Manage and control thread lifecycle**
- **Synchronize thread access to shared data**
- **Identify potential threading problems**

- **Concurrency**
- **Use java.util.concurrent collections**
- **Apply atomic variables and locks**
- **Use Executors and ThreadPools**
- **Use the parallel Fork/Join Framework**

- **Localization**
- **Read and set the locale by using the Locale object**
- **Build a resource bundle for each local**
- **Load a resource bundle in an application**
- **Format text for localization by using NumberFormat and DateFormat**

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).