

## Advanced Java Programming

### Audience Course Advanced Java Programming

The course Advanced Java Programming is intended for experienced Java developers who want to gain more in depth knowledge of Java.

### Prerequisites Course Advanced Java Programming

Knowledge of the Java language and syntax and basic experience in **Java** programming is required to participate in this course.

### Realization Training Course Advanced Java Programming

The theory is treated on the basis of presentations and is interspersed with exercises. Demos are used to clarify the theory. The course material is in English. The course times are from 9.30 up and to 16.30.

### Certification Course Advanced Java Programming

Participants receive an official certificate Advanced Java Programming after successful completion of the course.

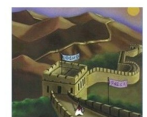
Duration: 4 days

Price: € 2650

[Open Schedule](#)



### Advanced Java Programming



## Content Course Advanced Java Programming

In the course Advanced Java Programming a series of advanced aspects of Java are discussed. The course covers the topics that are asked for on the Oracle Certified Java Professional or OCP exam and is a good preparation to pass this exam.

### Advanced Classes

In the first place attention is paid to aspects of Advanced Class Design such as the implementation of inheritance and composition, the use of polymorphism, interfaces, inner and anonymous classes and the singleton pattern.

### Concurrency

Next multithreaded applications are discussed and the synchronization between threads when accessing shared data. During the treatment of the concurrency package advanced synchronization mechanisms such as cyclic barriers and countdown latches are discussed.

### Lambda's

Also the functional language constructs introduced in recent Java versions are discussed with lambdas and functional interfaces.

### Generics

Next generics are on the course program with which classes and methods can be parameterized, strong typing is imposed and the chance of runtime errors is limited. Generics are used a lot in the Collection Framework and the most important container classes in this Framework are discussed.

### Stream API

Next attention is paid to the [Stream API](#) that enables transformations on data collections to be performed by a combination of successive simpler methods like map and reduce.

### Exceptions

The various possibilities for dealing with errors and exceptions are also on the program and attention is paid to file I/O and new I/O when accessing files and directories.

### JDBC

Database access with Java Database Connectivity (JDBC) is treated and queries, prepared statements and transactions are part of that.

### Reflection

Finally, if time permits, reflection is optionally on the course program, with which compiled Java classes can be analyzed by means of software, and optionally various aspects of enhancing the Java performance are discussed.

## Modules Course Advanced Java Programming

Module 1 : Advanced Class Design	Module 2 : Multiple Threads	Module 3 : Concurrency
Encapsulation and Inheritance Implementing Composition Polymorphism Singleton Patterns Abstract Classes Final Classes Inner Classes Static Inner Classes Anonymous Inner Classes Autonomous Classes Enumerated Types Implementing hashCode and equals	Java Thread Model Extending Thread Class Implementing Runnable Daemon Threads Thread Alive States Sleeping and Yielding Control Using join and interrupt Synchronized Statement Locking and Statics Deadlock Condition Synchronization Using wait and notify	Concurrency Package Task Scheduling Framework Executor Interface ExecutorService Callables and Futures Synchronizers Semaphores and Exchanger CountdownLatch and CyclicBarrier Concurrent Collections Lock Interface Reentrant Locks Atomic Variables
Module 4 : Lambda's	Module 5 : Generics	Module 6 : Streams
Passing Functionality Lambda Expressions Lambda Variable Access Lambda Scoping Rules Functional Interfaces Predicate Interface Consumer Interface Supplier Interface Function Interface UnaryOperator Interface BinaryOperator Interface Method References User Defined Functional Interfaces	Type Erasure and Raw Types Generics and Subtyping Bounded Type Parameters Wildcards Generics in Collections ArrayList and LinkedList TreeSet and Hash Set HashMap and TreeMap Comparable and Comparator Collections Streams and Filters Iteration using forEach Filtering using Lambda's Stream Pipeline	What are Streams? Lazy Evaluation and Parallelization forEach, Map and Filter findFirst and findAny toArray and collect Optional Class Limiting Stream Size allMatch and anyMatch Number Specialized Streams Parallel and Infinite Streams collect Method Grouping with Collectors class Using flatMap Method
Module 7 : Exception Handling	Module 8 : Java IO and NIO	Module 9 : Database Access
Errors and Exceptions Checked and Unchecked Exceptions Exception Hierarchy Multiple Catch Clauses finally Clause try with Resources Auto Closeable Resources Common Exceptions Throwing Exceptions User Defined Exceptions Chained Exceptions and Stack Traces Assertions	Standard I/O Streams Reading and Writing Files Buffered Streams Data Conversion Streams Serialization Object Streams NIO and Asynchronous I/O Processing IO Channels Stream API with NIO.2 Using Path Class Directory Traversing PathMatcher class	JDBC Architecture JDBC Drivers and URL's Database Connections Executing Statements Retrieving Results Handling Errors Prepared Statements Database Metadata Transactions Commit and Rollback Rowset Interfaces Using RowsetProvider
Module 10 : Localization	Optional Module 11 : Reflection	Optional Module 12 : Performance
LocalDate Class LocalTime and LocalDateTime Instant and Period Duration and TemporalUnit Defining Properties Reading Property Files Creating Resource Bundles Formatting Date and Times Locale Class Localizing Dates Localizing Numbers Localizing Currencies	What is Reflection? Reflection Classes Class Loading The Class Class Creating Objects Reflection Methods in Class Field Class Constructor Class Method Class AccessibleObject Class Dynamic Proxies Invocation Handler	Influences on Performance JIT Compilation and Hotspot JVM Garbage Collection String Types Buffered and New I/O Synchronization and Concurrency Primitives versus Wrappers Collections Exception Handling Serialization Native methods Lazy Loading and Object Reuse