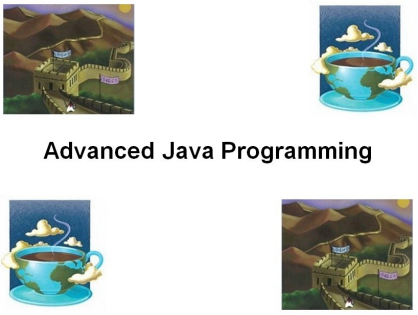
Advanced Java Programmeren

Duur: 4 dagen **Prijs: € 2650**

Open Rooster



Doelgroep Cursus Advanced Java Programmeren

De cursus Advanced Java Programmeren is bestemd voor ervaren Java developers die diepgaandere kennis van **Java** willen opdoen.

Voorkennis Cursus Advanced Java Programmeren

Om aan deze cursus te kunnen deelnemen is kennis van Java en ervaring met programmeren in Java vereist.

Uitvoering Training Advanced Java Programmeren

De theorie wordt behandeld aan de hand van presentaties en wordt afgewisseld met oefeningen. Demo's dienen ter verheldering van de theorie. De cursustijden zijn van 9.30 tot 16.30.

Officieel Certificaat Cursus Advanced Java Programmeren

De deelnemers krijgen na het goed doorlopen van de cursus een officieel certificaat Advanced Java Programmeren.

Inhoud Cursus Advanced Java Programmeren

In de cursus Advanced Java Programmeren komen een reeks geavanceerde aspecten van Java aan de orde. De cursus behandelt de onderwerpen die gevraagd worden op het Oracle Certified Java Professional of OCP examen en vormt een goede voorbereiding om dit examen te behalen.

Advanced Classes

Allereerst wordt aandacht besteed aan aspecten van Advanced Class Design zoals het implementeren van inheritance en composition, het gebruik van polymorfisme, interfaces, inner en anonymous classes en het singleton pattern.

Concurrency

Vervolgens wordt ingegaan op multithreaded applicaties en de synchronisatie tussen threads bij het benaderen van shared data. Bij de bespreking van het concurrency package komen daarbij geavanceerde synchronisatie mechanismes zoals cyclic barriers en countdown latches aan de orde.

Lambda's

Ook de in recente Java versies geïntroduceerde functionele taal constructies komen aan bod bij de behandeling van lambda's en functional interfaces.

Generics

Vervolgens worden generics besproken waarmee classes en methods kunnen worden geparametriseerd, strong typing wordt opgelegd en de kans op runtime errors wordt beperkt. Generics worden meestal gebruikt in het Collection Framework en de belangrijkste container classes daaruit worden besproken.

Stream API

Vervolgens is er aandacht voor de **Stream API** waarmee transformaties op data collections kunnen worden uitgevoerd door een combinatie van elkaar opvolgende simpelere methoden waaronder map en reduce.

Exceptions

Ook de diverse mogelijkheden bij de afhandeling van errors en exceptions staat op het programma en er wordt aandacht besteed aan file I/O en new I/O bij het benaderen van files an directories.

JDBC

Ook wordt database access met Java Database Connectivity (JDBC) behandeld waarbij queries, prepared statements en transactions aan de orde worden gesteld.

Reflection

Tenslotte staat optioneel, als de tijd het toelaat, reflection op het programma, waarmee gecompileerde Java classes softwarematig kunnen worden geanalyseerd, en komen optioneel diverse aspecten het verbeteren van de Java performance aan bod.

Modules Cursus Advanced Java Programmeren

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