

Szkolenie: Capstone Courseware 106 Advanced Java Programming



Cel szkolenia:

Version 8.0

This course provides advanced training in developing software using the Java Platform, Standard Edition, or Java SE. It is intended for students with solid experience in structured and object-oriented Java programming, including use of the Collections API and exception handling. Generic types should be understood, at least at a basic level; the course does begin with a refresher and then a more advanced treatment of generic types.

After a quick introduction to the Java Time API, students get familiar with the I/O streams model, file handling, and object serialization, and learn to use streams to communicate over network sockets. A two-chapter unit covers multi-threaded programming and concurrency techniques. We look at dynamic typing in Java, in the Reflection API and with dynamic proxies, and understand the underpinnings of source-code annotations.

Finally, several chapters at the end of the course introduce unit-testing and test-driven-development practices. Here for the first time we introduce external libraries -- [JUnit](#), and the [Mockito](#) dynamic-mocking library -- and the study is not entirely about technology but leans more into design and good practice.

Learning Objectives

- Make effective use of Java generic types.
- Understand the structure of streams in Java, and learn how to use streams to manage file I/O.
- Learn how to use Java Serialization to internalize and externalize potentially complex graphs of objects.
- Communicate between processes using network sockets.
- Write multi-threaded Java applications that safely manage concurrent access to application state.
- Use the Reflection API and dynamic proxies for highly generic tasks, discovery, or code-generation.
- Use standard annotations and develop custom annotations to express meta-data in Java source files.
- Build unit tests for Java classes using JUnit.
- Write effective tests, and design classes for testability.
- Understand test-driven development (TDD) and use dynamic mocking to support isolated

testing.

Plan szkolenia:

- Generics
 - Using Generics
 - Type Erasure
 - Type Boundaries
 - Wildcards
 - Generic Methods
 - Strengths and Weaknesses of Generics
 - Legacy Code and Generics
- The Time API
 - A History of Time ... in Java
 - Limitations of Date and Calendar
 - The Time API
 - Temporal Types
 - Accessors and Adjusters
 - Formatting
 - Decomposition Into Fields
 - Date Arithmetic
 - Managing Precision
 - Duration and Period
 - Time Zones and Offsets
 - Converting Between Time Zones
- The Java Streams Model
 - Delegation-Based Stream Model
 - InputStream and OutputStream
 - Media-Based Streams
 - Filtering Streams
 - Readers and Writers
 - Byte-Array Streams
 - String Readers and Writers
 - Closing Streams, Readers and Writers
- Working with Files
 - The File Class
 - Modeling Files and Directories

- File Streams
- Working with File Systems
- The Path Interface
- The Paths and Files Utilities
- Processing with `java.util.stream.Streams`
- Delegating Streams
 - Buffering
 - Data Streams
 - Push-Back Parsing
 - Byte-Array Streams and String Readers and Writers
- Java Serialization
 - The Challenge of Object Serialization
 - Serialization API
 - Serializable Interface
 - `ObjectInputStream` and `ObjectOutputStream`
 - The Serialization Engine
 - Transient Fields
 - `readObject` and `writeObject`
 - Externalizable Interface
- Sockets
 - The OSI Reference Model
 - Network Protocols
 - The Socket Class
 - The `ServerSocket` Class
 - Connecting Through URL Objects
 - HTTP and Other TCP Servers
 - Datagram Clients and Servers
 - Non-Blocking Sockets
- Threads
 - Java Thread Model
 - Creating and Running Threads
 - Manipulating Thread State
 - Thread Synchronization
 - Synchronized Blocks and Methods
 - `wait` and `notify`
 - `join` and `sleep`

- Multi-Threading in Servers
- Concurrency
 - The Concurrency API
 - Semaphore and Other Synchronizers
 - Concurrent Collections
 - Atomic Operations
 - Executor and ExecutorService
 - Thread Pools
 - Parallel Processing
- Reflection
 - Uses for Meta-Data
 - The Reflection API
 - The ClassClass
 - The java.lang.reflect Package
 - Reading Type Information
 - Navigating Inheritance Trees
 - Dynamic Instantiation
 - Dynamic Invocation
 - Reflecting on Generics
- Dynamic Proxies
 - The Proxy Pattern
 - Dynamic Proxies in Java
 - Use Cases
 - The InvocationHandler Interface
 - Proxy Classes
- Annotations
 - Aspect-Oriented Programming and Java
 - The Annotations Model
 - Annotation Types and Annotations
 - Built-In Annotations
 - Annotations vs. Descriptors (XML)
- Automated Unit Testing with JUnit
 - Automated Testing
 - JUnit and Related Tools
 - The @Test Annotation
 - The Assert Class Utility

- Test Runners
- Lifecycle Methods
- Expecting Exceptions
- Test Suites
- Writing Tests
 - Test Granularity
 - Reusing Test Logic
 - Recording and Comparing Output
 - Test Isolation
 - Controlling the Test Environment
 - Managing Dependencies
 - Non-Invasive Testing
 - Designing for Testability
 - Factories
 - Testing and Threads
- Test-Driven Development
 - Writing the Test First
 - The TDD Cycle
 - Advantages of TDD
 - Resistance to TDD
 - A Case Study
- Mocking
 - Mock Objects in Testing
 - Mock Objects in Test-Driven Development
 - Static vs. Dynamic Mocks
 - Stubbing
 - Verifying
 - Matching and Capturing
 - Using a Spy
 - Partial Mocking

Wymagania:

Solid **Java programming** experience is essential -- especially object-oriented use of the language. Language features and techniques that are integral to some lab exercises include interfaces and abstract classes, threading, generics and collections, and recursive methods. Course 103, "[Java Programming](#)," is excellent preparation.

Poziom trudności



Certyfikaty:

The participants will obtain certificates signed by Capstone Courseware.

Prowadzący:

Authorized Capstone Courseware Trainer.