

Szkozenie: Capstone Courseware  
104 Intermediate Java Programming

FORMA SZKOLENIA	MATERIAŁY SZKOLENIOWE	CENA	CZAS TRWANIA
Stacjonarne	Tradycyjne	4600 PLN NETTO*	5 dni
Stacjonarne	Tablet CTAB	5000 PLN NETTO*	5 dni
Metoda dlearning	Tradycyjne	4600 PLN NETTO*	5 dni
Metoda dlearning	Tablet CTAB	4600 PLN NETTO*	5 dni

\* (+VAT zgodnie z obowiązującą stawką w dniu wystawienia faktury)

## LOKALIZACJE

Kraków - ul. Tatarska 5, II piętro, godz. 9:00 - 16:00

Warszawa - ul. Bielska 17, godz. 9:00 - 16:00

## Cel szkolenia:

## Version 8.0

This course teaches programming in the Java language -- i.e. the **Java Standard Edition** platform. It is intended for students with previous Java experience or training, who already know the fundamentals of the Java architecture and basic procedural programming. This course provides in-depth coverage of object-oriented concepts and how to apply them to Java software design and development. We then move from these basic skills into key parts of the **Java SE Core API**, including collections and logging, and introduces features of functional programming, new to the language as of Java 8, including functional interfaces, lambda expressions, and streams.

This revision of the course targets the Java 8 language and Core API.

Students come to Java from a wide range of backgrounds, and this course is designed to be as flexible as possible over the upper end of that range. Specifically:

- Experienced C and C++ programmers will find this course a very good fit and if anything will find that they complete it in a little less than the full five-day timeline.
- Those with experience in languages less like Java, such as Visual Basic, ASP and other Web-scripting languages, and other pseudo-object-oriented languages may need more time in the early going, and this course covers its introductory topics in good depth and offers many optional and "challenge" labs to support this.
- Less experienced programmers or those coming from non-structured languages -- such as COBOL, PL/1, or 4GL tools -- will probably not cover the whole course in a week, and may want to pursue an abbreviated version at a slower pace. This too is quite feasible, but this audience may also want to consider course 102 Introduction to Java Programming, for a more relaxed

pace through the early material.

## Learning Objectives

- Chiefly, learn to program effectively in the Java language.
- Understand Java as a purely object-oriented language, and implement software as systems of classes.
- Implement and use inheritance and polymorphism, including interfaces and abstract classes.
- Design appropriate exception handling into Java methods, and use the logging API appropriately.
- Use Java as a functional language, making appropriate choices of tools including inner classes, functional interfaces, method references, and lambda expressions.
- Use the Stream API for efficient processing of data sets.

## Plan szkolenia:

- Chapter 1. Review of Java Fundamentals
  - The Java Architecture
  - Forms for Java Software
  - Three Platforms
  - The Java Language
  - Numeric Types
  - Characters and Booleans
  - Enumerations
  - Object References
  - Strings and Arrays
  - Conditional Constructs
  - Looping Constructs
  - Varargs
- Object-Oriented Software
  - Complex Systems
  - Abstraction
  - Classes and Objects
  - Responsibilities and Collaborators
  - UML
  - Relationships
  - Visibility
- Classes and Objects

- Java Classes
- Constructors and Garbage Collection
- Naming Conventions and JavaBeans
- Relationships Between Classes
- Using this
- Visibility
- Packages and Imports
- Overloading Methods and Constructors
- JARs
- Inheritance and Polymorphism in Java
  - UML Specialization
  - Extending Classes
  - Using Derived Classes
  - Type Identification
  - Compile-Time and Run-Time Type
  - Polymorphism
  - Overriding Methods
  - The @Override Annotation
  - Superclass Reference
- Using Classes Effectively
  - Class Loading
  - Static Members
  - Statics and Non-Statics
  - Static Initializers
  - Static Imports
  - Prohibiting Inheritance
  - Costs of Object Creation
  - Strings and StringBuffer
  - Controlling Object Creation
  - Understanding Enumerated Types
  - Stateful and Behavioral Enumerations
- Interfaces and Abstract Classes
  - Separating Interface and Implementation
  - UML Interfaces and Realization
  - Defining Interfaces
  - Implementing and Extending Interfaces

- Abstract Classes
- Collections
  - Dynamic Collections vs. Arrays
  - UML Parameterized Type
  - Generics
  - Using Generics
  - The Collections API
  - The Collection and List Interfaces
  - The ArrayList and LinkedList Classes
  - Looping Over Collections: Iterable
  - Collecting Primitive Values: Auto-Boxing
  - Using Wildcards with Generic Types
  - Iterators and the Iterator Interface
  - Maps and the Map Interface
  - Sorted Collections
  - The SortedSet and SortedMap Interfaces
  - The Collections Class Utility
  - Algorithms
  - Conversion Utilities
- Exception Handling and Logging
  - Reporting and Trapping Errors
  - Exception Handling
  - Throwing Exceptions
  - Declaring Exceptions per Method
  - Catching Exceptions
  - The finally Block
  - Catch-and-Release
  - Chaining Exceptions
  - try-with-resources
  - Logging
  - The Java SE Logging API
  - Loggers
  - Logging Levels
  - Handlers
  - Configuration
  - Best Practices

- Nested Classes
  - Nested Classes
  - Static Classes
  - Inner Classes
  - Relationship with the Outer Object
  - Local Classes
  - Enclosing Scope
  - Anonymous Classes
- Functional Programming
  - Passing Behavior as a Parameter
  - Inner Classes
  - Functional Interfaces
  - Built-In Functional Interfaces
  - Lambda Expressions
  - Scope and Visibility
  - Deferred Execution
  - Method References
  - Creational Methods
  - Designing for Functional Programming
  - Default Methods
- Streams
  - The Stream Processing Model
  - Streams
  - Relationship to Collections
  - Advantages and Disadvantages
  - Iterating, Filtering, and Mapping
  - Primitive-Type Streams
  - Aggregate Functions and Statistics
  - Sorting
  - Generating, Limiting, and Reducing
  - Finding and Matching
  - Grouping
  - Flattening and Traversing
  - Sequential vs. Parallel Processing

## Wymagania:

Students must be able to write, compile, test, and debug simple [Java programs](#), using structured programming techniques, strong data types, and flow-control constructs such as conditionals and loops. **Course 102 [Introduction to Java Programming](#)** is ideal preparation for this course.

## Poziom trudności



## Certyfikaty:

The participants will obtain certificates signed by Capstone Courseware.

## Prowadzący:

Authorized Capstone Courseware Trainer.