## Effective Java 3 Days Hands-On Course

**Take the next step in your career and become a serious Java developer! You will implement a number of patterns and best practices in order to produce high quality code in terms of performance, maintainability and readability.**

**The course adheres to the guide lines defined by Joshua Bloch in ”Effective Java”, and systematically covers a number of areas within software development using Java, explaining how to avoid common pitfalls. The course expands on the many new features introduced in Java 8 such as the new Date & Time API, functional programming using lambda expressions, enhanced interface definitions and the new Stream API.**

**Code examples/exercises will be presented in UML and correct robust Java code, complemented by a great number of predefined unit tests.**

**Target Group**

Java programmers who want to enhance his/her competence in Java programing aiming for producing robust code of great quality.

Programmers having skills from a previous Java version, who need an update and want to get a deeper knowledge of the Java programing language and its core APIs.

Experienced C++ / C# programmers who want an introduction to Java at full speed!

**Hands-On**

Every chapter is backed by complete examples that are ready to run. In addition there are a number of exercises that will be addressed and discussed during the course. All Java code, examples and solutions, will be part of the documentation available for the participants.

## Documentation

Copies of course slides

“Effective Java 2nd ed” Joshua Bloch

Complete code for all examples/exercises

## Contents:

**1. Unit Testing using JUnit4/TestNG**

* Overview of unit testing

**2. Overriding Object Methods**

* equals
* hashCode
* toString
* Implementation of interface Comparable<T>

**3. Software Design - Classes**

* Level of access
  + Encapsulation
  + Public methods and private attributes
  + Immutable classes – value classes
* Class design
  + Inheritance vs. composition
* Creating and destroying objects
  + Static factory methods vs. constructors
  + Design Pattern Singleton
* Nested classes
  + Static classes, Inner classes, Anonymous classes, Local classes

**4. Software Design – Interfaces**

* Class design
  + Abstract base classes vs. interfaces
* Java 8 additions
  + Interfaces supporting default and static methods
  + default methods vs. Inheritance

**5. General Programming Concepts**

* Know the core Java APIs
* Avoid floating point types for exact calculations
* Performance for wrapper classes vs. primitive types
* Performance issues for String
* Validating method arguments
* Defensive copying of return values
* Avoid returning null
* Using Formatter for textual output

**6. Date & Time API**

* Time-based Time
  + Instant
  + Duration
* Date-based Time
  + LocalTime, LocalDate, LocalDateTime
  + Period
* Zone-based Time
  + ZoneID
  + ZonedDateTime

**7. Generics and Typesafe Collections**

* Generics
  + Parameterized types
  + Arrays vs. Generics
* Type safety in collection classes
  + Erasure guarantees backwards compatibility

**8. The Collections Framework in Detail**

* Basic interfaces
  + List, Set, SortedSet, Queue, Map, SortedMap
* Basic implementation classes
  + ArrayList, HashSet, TreeSet, HashMap, TreeMap
* Iterator vs. for:each-loop
* Specialized interfaces
  + Deque, NavigableSet, NavigableMap
* Implementation classes for specialized interfaces
  + ArrayDeque, PriorityQueue
* Performance aspects for implementation classes
* Synchronized classes
* Java 8 enhancements

**9. Enums**

* Enum defining named constants
* Enum as a Java type defining attributes and methods
* Collection classes using Enum based keys
  + EnumSet,EnumMap

**10. Annotations**

* + Annotation as a Java type
  + Rules for usage
  + Retention policies
  + Annotations usage in Java frameworks

**11. Exception handling**

* Overview exceptions
  + Runtime exceptions
  + Checked exceptions
  + Chained exceptions
* Exception or special return value
* Exception usage best practice

**12. Serialization**

* Serialization mechanism in Java
* Customization of serialization
* Externalizable for special purposes
* Version handling for Serializable classes

**13. Lambda Expressions**

* Functional programming using lambda expressions
  + Syntax for Lambda Expressions
  + Deferred execution
  + Capture of variables
* Functional Interfaces
  + **Built in support in Java APIs**
* **Predefined Interfaces supporting Lambda Expressions**
  + **Function, Operator, Supplier, Consumer, Predicate**
* **Exception handling in lambda expressions**
* Method references

**14. The Streams API**

* Stream creation from existing sources
  + Arrays, Collections, Files
* Iterations using stream operations
* Streams using lambda expressions
* Filtering, transforming and reducing streams
  + Optional<T> handling existing/missing return value
* Bulk operations for collections
  + Piping operations for enhanced performance
* Enabling parallel processing of streams