DevelopIntelligence

Intermediate Java

The Intermediate Java course examines common language features and APIs required to develop complex stand-alone Java applications. The course builds on the Introduction to Java course and examines topics such as lambda expressions, generic programming, and new features in Java 8.

Productivity Objectives:

This Class Does Include:

- Key Design Patterns
- Using and Defining Generic Classes and Methods
- Introduction to Functional Programming with Higher Order Functions and Immutable Data
- Java 8 Streams and Simple Monads
- Dynamic Java
- Introduction to Concurrency

This Class Does Not Include:

- Basic OO concepts; Encapsulation, Inheritance, Generalization, Instance vs Static Features
- Java syntax from Java 7 other than Generics

Course Duration: This course will be delivered in 3 Days

Course Outline:

Design Patterns

- Singleton
- Builder
- Factory
- Command

Lambda Expressions

- Lambda Syntax
- Constraints on Using Lambas
- Functional Interfaces

- Method References

Functional Programming Concepts
- Higher Order Functions
- Working with Immutable Data
- Closures

Streams
- Terminal vs non-terminal operations
- forEach, filter, map, flatMap
- Reduction - reduce and collect
- Collectors
- Primitive Stream Types
- Parallel Streams
- Optional

Generics
- Creating Generic Types
- Generic Methods
- Type Variable Bounds
- Type Erasure
- Inheritance and Generics
- Co- and Contra-Variance

Reflection and Annotations
- Examining a Loaded Class
- Loading a Class Dynamically
- Finding Annotations on Syntactic Elements of Loaded Classes
- Interacting with Methods and Fields Dynamically
- Defining Annotations
- Controlling Applicability and Retention of Annotations
- Using Annotations with Key-Value Pairs

Introduction to Concurrency
- Creating Threads with Runnable
- Key concerns of cooperating threads:
    - Visibility and Safe Publication
    - Transactional Integrity
    - Timing and Race Conditions
- Blocking Queues and the Pipeline/Actor Model
- Thread-Safe Collections
- Using Locks and Synchronizers
- Thread Pools With ExecutorService, Callable, and Future