

# Sela.

J7J-xtrm

## Java Design Patterns & Java Extreme

college@sela.co.il

03-6176666





## Java Design Patterns & Java Extreme

J7J-xtrm - Version: 1

 8 days Course

### Description:

Java Design Patterns segment:<br><br>

In this segment, programmers will be introduced to ideas and techniques commonly referred to as Java language patterns. Patterns are reusable solutions to recurring problems arising during software development. This course will introduce the programmer to common patterns and their implementation in the Java language. Patterns will be associated with their uses within the Java API, followed by design and implementation exercises to correlate several design patterns.

To make the best from the course, the first day will be devoted to some java language core issues such as reflection and memory management. The course starts with a “Pitfalls” section in which we will discuss some common misunderstandings in the Java language, clarify them and learn how to avoid them.<br> <br>

Java Internals segment:<br><br>

This segment will take you deep into the internal of java. The segment will take you inside the JVM and will deal with issues such as memory and thread management. The course will help you understand better the internal of your program and how to optimize it.

### Intended audience:

This course is intended for Java programmers, team leaders, software project managers, Java designers and Object Oriented analysts that need to design system-level, software-level or module-level components to be implemented in the Java programming language.



## Prerequisites:

## Objectives:

- Learn what design patterns are useful for
- Java programmers will associate the ideas with a real-life usage of patterns
- Java programmers will be able to apply design patterns in their design work and implement them in the Java programming language
- Get to know the advanced topics pertinent to Java programming
- Learn to avoid common design and implementation mistakes and pitfalls
- JVM internal mechanisms and optimization
- efficient error management and data storage
- Thread management

## Topics:

### Inside the java virtual machine

- Java introduction
- JVM overview
- Java flow
  - Compiling Java
  - Linking Java
  - Running Java
- Class Loader
  - Class Loader overview
  - Implementing your class loader
  - Class loader & namespace
  - Class loader lab
- Understanding bytecode



- Dependency Injection architecture
- Instrumentation
- Java Assist
- Instrumentation Lab
- Java profiling
- JNI – Java Native Interface
  - What is JNI?
  - The “native” Keyword.
  - The javah process.
  - Invoking native methods.
  - Passing arguments to native methods.
  - Objects, Arrays and Strings.
  - Local and Global references.
  - Callbacks methods’ id and fields’ id.
  - Handling Java errors (exception handling).
  - Threads.
  - Performance issues

## Java concurrency in depth

- Threads & JVM
  - Inside java.lang.Thread
  - Green thread vs native threads
  - Thread priority
  - Appendix volatile
- Thread scheduling
  - Scheduling in the JVM
  - Performance meaning
- Threads operations
  - Stop / Suspend thread
  - isAlive & Active count



- Join a thread
- Recommended usage of yield & interrupt
- Daemon thread
- Thread lab
- Util concurrent package
  - Executors
  - Callback function
  - Future design pattern
  - Executors Service
  - ThreadPool
  - Thread pool lab
  - Barrier
  - Semaphore
  - Read & Write Lock
  - Util concurrent lab

## JMX

- Overview of the JMX Technology
- Introducing MBeans
- MXBeans
- Notifications
- Remote Management

## Optimization

- NIO
  - I/O overview
  - Buffers
  - Channels



- Selectors
- Selection Keys
- NIO lab
- Serialization
  - Serialization overview
  - The transient keyword
  - Customizing serialization
  - Serialization lab
- Optimization technique
  - Data structure optimization
  - Distributed optimization
  - Code optimization
- JVM tuning
  - Garbage collection tuning
  - Stack tuning
  - Memory tuning
  - Assertion
- Optimization lab

## Java Pitfalls

- Common Java pitfalls and how to avoid them
- Equals(...) vs. ==
- Cloning objects

## The reflection mechanism

- The java.lang.Class
- Dynamic instantiation
- Method invocation.

- Design Issues and limitations.
- annotation

## Memory management

- Memory overview
- Memory profiling
- Object creation
- Reference management
  - Soft reference
  - Weak reference
  - Phantom reference
- Garbage collection
- Cache management
- Memory management lab

## AOP appendix

- AOP overview
- AspectJ
  - AspectJ overview
  - AJDT
- AspectJ language & API
  - Points
  - Advice
  - Aspects
  - API
- AspectJ lab
- Appendix 2. References.



## ▫ Introduction to Design Patterns

### Design Principles

- The Open-Closed Principle
- Dependency Inversion Principle
- Interface Segregation Principle
- Single Responsibility Principle
- Liskov Substitution Principle & Design by Contract

### UML Overview

- Introduction to models
- Class diagrams
- Association
- Aggregation
- Generalization

### Fundamental design patterns

- Delegation
- Strategy
- Interface
- Immutable
- Marker interface
- Proxy
- The Dynamic Proxy class

### Creational patterns





- Singleton
- Factory method
- Abstract factory
- Object pool
- The Prototype Pattern

## Structural Patterns

- Composite
- Adapter
- Iterator
- Façade
- Decorator
- Bridge
- Double Dispatch idiom

## Behavioral Patterns

- Command
- Template method
- State
- The Flyweight Pattern
- Observer
- Visitor