2. System Design

Duration: 16 hours (4 hours * 4 days)

Course level: Intermediate

Course Outline: Common & technology area specific architectural patterns and practices along with use-cases and benefits of adopting programming best Practices.

Pre-requisite: Knowledge & skills on Java Programming Language

What participants will learn:
Participants will be able to understand need for Programming Best Practices
Participants will be able to understand need for Design Patterns

CONTENTS

Day 1

Technical Debt, Design quality, Design principles

- Software Engineering
- Why care about design quality?
- Q-Minimum design guidelines
- Introduction to design principles and design smells
- Design smell examples
- Design metrics computation and smell detection tools

SOLID

- Single Responsibility Principle
- Procedural (open) version
- A Closed Implementation
- Open-Closed Principle
- Liskov's Substitution Principle
- Interface Segregation Principle
- ISP in Action
- Dependency Inversion Principle
- Applying the DIP
- Dependency Inversion Principle
- Some Other Commonly Used Principle

Day 2

Overview of Design Patterns

- What are design patterns?
- What kind of problems do they solve?
- What makes a pattern a “pattern”?
- Benefits and drawbacks of patterns.
- Design Patterns in Real Life
- Software Design Patterns – Gang of Four

Pattern Catalog

- Creational Patterns – Initialize and configure objects
- Structural Patterns – Decouple interface and implementation
- Behavioral Patterns – Dynamic interaction between classes and Objects

Applying Patterns

- Communicable Name
- Intent
- Force(s) addressed
- Structure and Collaboration
- Consequences
- Known uses
- Related pattern(s)

Day 3

Creational Patterns

- Singleton Pattern – Restricting multi-instance problem in given context
- Class Factory (non GoF) – Dynamically creating objects at runtime
- Factory method – Deferring creation to subclasses
- Abstract Factory – Creating family of related classes
- Prototype – Cloning objects for enhanced performance
- Object Pool (non GoF) – Reducing creation-time overheads
- Builder – Constructing complex objects incrementally

Day 4

Structural Patterns

- Adapter – Translating Server interface for the client
- Bridge – Abstraction binding one of many implementations
- Composite – Building recursive aggregations
- Decorator – Extending objects transparently
- Facade – Building a simple interface for complex subsystems
- Flyweight – Sharing fine-grained objects efficiently
- Proxy – Approximating target object

Behavioral Patterns

- Chain of responsibility – Routing requests thru a list of service providers
- Command – Encapsulation request in objects and dispatching
- Interpreter – Implementing small language grammar
- Iterator – Accessing aggregation/collection sequentially
- Mediator – Coordinate interaction between associates
- Memento – Capture and Restore object's state's snapshots
- Observer – Implementing update notification when subject changes
- State – Object behavior change based on current state
- Strategy – Substitute algorithms dynamically at runtime
- Template Method – Deferring algorithm steps selectively to subclasses
- Visitor – Externalizing operations on a object's structure

Course will have theory, demonstration, assignments. The course will begin with Pre-Test and end with Post-Test