**Software Design and Architecture: Principles, Patterns and Practices**

**After completing this course, you will be able to:**

* Improve logical design thinking using design patterns.
* Recognize that software engineering is more that writing code.
* Understand the meaning of clean code.
* Implement programming solutions using multiple approaches and recognize tradeoffs.
* Use object-oriented concepts including interfaces and abstract classes for solving complex design issues.
* Understand what good code are and what bad code are.
* Understand what are good design and what are bad design
* Recognize the code smell and find the appropriate refactoring solution.
* Understand the design principles and implement them in writing object-oriented code.

**Course Outline**

1. Quick overview on object oriented programming and concept.
2. What is software design?
3. What is software architecture? What are roles of a software architect?
4. Is design dead?
5. Code smells and refactoring techniques.
6. Design Patterns:

|  |  |  |
| --- | --- | --- |
| 1. Creational Patterns | 1. Structural Patterns | 1. Behavioral Patterns |
| * Abstract Factory * Builder * Factory Method * Prototype * Singleton | * Adapter * Bridge * Composite * Decorator * Façade * Flyweight * Proxy | * Chain of Responsibility * Command * Iterator * Mediator * Memento * Observer * State * Strategy * Template Method * Null object pattern |

1. Design Principles
   * Single Responsibility Principle (SRP)
   * Open Closed Principle (OCP)
   * Liskov Substitution Principle (LSP)
   * Interface Segregation Principle (ISP)
   * Dependency Inversion Principle (DIP)
   * DRY–Don’t Repeat yourself
   * Once and only once
   * Tell Don’t Ask
   * The Law of Demeter
   * Inversion of Control
   * Dependency Injection