Diplomado de UML

Análisis y Diseño de Sistemas Orientados a Objetos (UML 2.0, RUP, EA y Patrones de Diseño)

Introducción: Orientado a quienes tienen que hacer la labor de arquitectura de Sistemas Orientados a Objetos, incluyendo tareas como levantamiento de requerimientos, modelación, definición e implementación de patrones y definición de clases, nuestro *Diplomado Diseño de Sistemas Orientados a Objetos* tiene como finalidad enseñar el estándar UML para la modelación de los requerimientos reales de un sistema, enseñar el uso de la metodología RUP así como de herramientas y patrones de diseño que permitan aterrizar los conceptos para la generación de modelos, clases y código en desarrollo de sistemas orientados a objetos. El diplomado incluye el curso *Object Oriented Analysis and Design with UML* y otros temas como Patrones de Diseño, Enterprise Architect como herramienta de modelación, Programación Orientada a Objetos y creación de código.

Descripción: Este diplomado tiene como objetivo que el alumno aprenda las técnicas de Análisis y Diseño Orientado a Objetos así como la metodología RUP para el desarrollo de aplicaciones utilizando UML y Patrones de Diseño para Java (principalmente) aunque aplica a otros lenguajes como .NET.

Contempla los cursos: Análisis Orientado a Objetos con UML 2.0 y RUP, POO en Java, Patrones de diseño con Java y Enterprise Architect. A lo largo del diplomado el alumno aprenderá y utilizará Enterprise Architect, una herramienta líder en el mercado para generar sus diagramas UML y desarrollará una aplicación hasta la generación de código.

Audiencia: Arquitectos, analistas y desarrolladores de software que desean aprender UML, RUP y Patrones de Diseño para generar la arquitectura de sus proyectos de ingeniería de software con Java.

Prerrequisitos: Conocimientos de algún lenguaje de programación son necesarios para entender la generación de código en Java.

Módulos del Diplomado:

Módulos	Horas
UML	40
OOP en Java	16
Uso de Enterprise Architecture	12
Diseño de Patrones en Java	24
Desarrollo de la aplicación	20
TOTAL	112

Análisis y Diseño Orientados a Objetos usando UML

Descripción: El alumno aprenderá las técnicas para analizar los requerimientos del mundo real y a diseñar soluciones que queden listas para codificar utilizando el lenguaje UML. Aprenderá a plasmar con casos de uso y diagramas de interacción el levantamiento de requerimientos del sistema a desarrollar; aprenderá a identificar y diseñar objetos, clases, y las relaciones entre ellos, incluyendo ligas, asociaciones y herencia. Asimismo, identificará la interrelación entre los diagramas UML y, utilizando la Metodología RUP, aprenderá a evolucionarlos para llegar a diagramas codificables en cualquier lenguaje orientado a objetos.

Audiencia: Analistas, diseñadores y programadores responsables de aplicar las técnicas Orientadas a Objetos en los proyectos de ingeniería de software.

Prerrequisitos: Es recomendable que el alumno haya programado en algún lenguaje orientado a objetos.

Introducción al análisis y diseño orientado a objetos

- El proceso de desarrollo de software
- Herramientas para el desarrollo de software
- Modelar
- Perspectivas
- ¿Por qué UML y UP?
- Las herramientas CASE
- Práctica

Diagramas de casos de uso

- Definición
- Uso
- Elementos
- Generalización
- Inclusiones y extensiones
- Narrativas
- Ejemplo de narrativa
- Práctica

Diagramas de estado

- El estado
- Elementos
- Transiciones y condiciones
- Efectos
- Transiciones internas
- Superestados y subestados
- Estados concurrentes
- Uso de los diagramas de estado
- Práctica

Diagramas de actividad

- Definición y uso
- Elementos
- Otros elementos del diagrama
- A detalle
- Particiones
- Nodos adicionales
- Parámetros
- Pines
- Manejador de excepciones
- Regiones de expansión
- Práctica

Objetos

- Definición
- Ejemplo del uso de un objeto
- Representación de un objeto en UML
- Responsabilidades
- Elementos de un objeto
- Objetos compuestos
- Abstracción
- Encapsulamiento
- Polimorfismo
- Herencia
- Práctica

Clases

- Definición del diagrama de clases
- Uso del diagrama de clases
- Diagramas de clase
- Clases compuestas
- Objetos
- Constructores
- Destructores
- Creación de una instancia
- Alcance de clase
- Plantillas
- Práctica

Relaciones

- Asociaciones
- Multiplicidad
- Asociaciones calificadas
- Clases de Asociación
- Roles
- Composición
- Agregación
- Dependencias
- Herencia
- Herencia múltiple
- Visibilidad
- Polimorfismo
- Clases abstractas
- Interfaces
- Práctica

Diagramas de secuencia

- Definición y uso
- Elementos
- Marcos de iteración
- Creación y destrucción de objetos
- Activación
- Procesos síncronos y asíncronos
- Evaluación
- Práctica

Diagramas de organización

- Definición y uso
- Diagrama de paquete
- Diagrama de componentes
- Diagrama de distribución
- Práctica

Los procesos

- Definición y uso
- Procesos formales
- El proceso unificado
- El análisis de riesgo
- Las pruebas
- Refactorización
- Los procesos en cascada e iterativos
- Procesos ágiles
- Práctica

El proceso unificado

- Presentación
- Inicio
- Elaboración
- Construcción
- Transición
- Práctica

El Análisis de dominio

- El dominio
- Diccionario de datos
- Formación de elementos
- Las tarjetas CRC
- Los modelos
- Práctica

Desarrollo de la solución

- Los requerimientos
- Los prototipos
- Otras consideraciones
- Práctica

Diagramas de comunicación

- Definición y uso
- Elementos
- Otros aspectos
- Práctica

Object Oriented Programming in JAVA

Descripción: Aprender a hacer código orientado a Objetos en Java. Las clases y las relaciones que aprendiste en el módulo de UML ahora las codificarás en Java.

Audiencia: Analistas, diseñadores y programadores responsables de aplicar técnicas de POO a sus proyectos de ingeniería de software usando Java.

Prerrequisitos: Familiaridad con UML y Java.

POO

- ABSTRACCIÓN
- ENCAPSULACIÓN
- HERENCIA
- POLIMORFISMO
- CLASES E INSTANCIAS

MANEJO DE OBJETOS

- Creación de Objetos
- El operador new
- Constructores
- La clase Objeto
- Destrucción de un objeto
- Paquetes
- El objeto String
- Arreglos
- for each en Arreglos

CREACIÓN DE TUS PROPIAS CLASES

- Creación de una clase
- Propiedades
- Métodos
- Constructores
- Destructores
- Overloading

Herencia

- Herencia
- Overriding
- Super clase
- Interfaces

Polimorfismo

- Polimorfismo
- Enlace tardío (Late Binding)

Using Enterprise Architect to Model your UML Diagrams

Objetivo: Aprender a usar Enterprise Architect como una herramienta para modelar tus diagramas de UML.

Audiencia: Analistas, diseñadores y programadores responsables de aplicar técnicas de POO a sus proyectos de ingeniería de software usando Java.

Prerrequisitos: Familiaridad con UML y Java.

Contenido:

Using Enterprise Architect

- The Application Workspace
- The Start Page
- Model Patterns
- Arranging Windows and Menus
- The Main Menu
- View Options
- Searching a Project
- Workspace Toolbars
- The Project View Browser
- Dockable Windows
- The Quick Linker
- The UML Toolbox
- Package Tasks
- Diagram Tasks
- Element Tasks
- Element Inplace Editing Options
- Defaults and User Settings
- Keyboard Shortcuts

Structural Diagrams

- Class Diagrams
- Object Diagrams
- Component Diagrams
- Composite Structure Diagrams
- Deployment Diagrams
- Package Diagrams

Behavioral Diagrams

- Interaction Diagrams
- Sequence Diagrams
- Communication Diagrams
- Interaction Overview Diagrams
- Timing Diagrams
- Activity Diagrams
- Use Case Diagrams
- State Machine Diagrams

Importante: Este módulo es práctico y no se entrega documentación de EA.

Design Patterns

Descripción: Este módulo cubre los patrones en tres de las áreas base: Creación, Estructural y Comportamiento. Es práctico conformado con proyectos de diseño y laboratorios de programación.

Audiencia: Desarrolladores de aplicación, programadores, diseñadores de sistemas y administradores de proyectos que necesitan mejorar el desarrollo de los sistemas usando patrones de diseño.

Prerrequisitos: Conocimiento profesional en Programación Orienta a Objetos, tecnologías orientada a objetos y diagramas UML. Conocimiento básico de Java.

Contenido

INTRODUCTION

- What's our World?
- OK – So Just What is a Design Pattern?
- Design Patterns are not Esoteric
- Why Use Patterns?
- The Adapter Pattern
- Reviewing Interfaces & Abstract Classes
- Interface Types
- Interface Definitions
- Abstract Methods
- Abstract Classes
- Using Abstract Classes
- Important Principal of OO Design

THE ITERATOR PATTERN

- Patterns: Traversing a Collection
- A Simple ArrayList
- Using Our ArrayList
- Using Our Simple Collection
- Another Design for Collection Traversal
- Using Our New Collection
- Differences in Traversing Our Collection
- Why is This Important?
- Why is This a Design Pattern?
- We Will Expand on Our Design

DECORATOR PATTERN

- Motivation – Forces and Solution
- Structure
- Participants an Collaborations
- Structure
- Writer and FilterWriter Classes
- UpperCaseFilterWriter Class
- Consequences
- Implementation
- Known Uses and Related patterns

COMPOSITE PATTERN

- Motivation – Forces
- Motivation – Solution
- Structure
- Participants
- Collaborations
- Consequences
- Implementation
- Known Uses and Related Patterns

TEMPLATE METHOD PATTERN

- Motivation – Forces and Solution
- Structure
- Participants and Collaborations
- Consequences
- Implementation
- Known Uses and Related Patterns

DESIGN PATTERNS – BACKGROUND

- Design Patterns Arise From Architecture
- Christopher Alexander
- The TimelessWay
- A Core Principle of His Books
- Patterns in A Pattern Language
- Sitting Circle (185)
- Different Chairs (251)
- Patterns Evolution in Software
- OOPSLA 88
- Patterns Evolution in Software
- Patterns Today

UML OVERVIEW

- Unified Modeling Language (UML)
- Using UML
- UML Diagrams
- Class Diagram
- Class Diagram Notation
- Association Relationships in Detail
- Class Diagram Notation
- Abstract Class Notation
- Interface Notation
- Another Class Diagram

GANG OF FOUR DESIGN PATTERNS DESCRIPTION

- What Do We Know Now About Patterns
- GOF Pattern Description
- Iterator: Overview
- Iterator: Motivation
- Iterator: Applicability
- Iterator: Structure – Java
- Iterator: Structure – General
- Iterator: Participants
- Iterator: Collaborations and Consequences
- Iterator: Implementation
- Implementation: Who Controls the Iteration
- Implementation: Who Defines the Traversal
- Implementation: Robustness
- Iterator: Known Uses and Related Patterns
- So – What is a Design Pattern?

THE GOF PATTERNS CATALOG

- Organizing the Catalog
- Creational, Structural, and Behavioral Purpose
- Class and Object Scope
- Design Pattern Space
- The GOF Catalog of Design Patterns

COMMAND PATTERN

- Motivation – Forces and Solution
- Structure
- Participants and Collaborations
- Consequences
- Implementation
- Undo and Redo
- Known Uses

CHAIN OF RESPONSIBILITY PATTERN

- Motivation – Forces
- Motivation – Solution
- Structure
- Participants and Collaborations
- Consequences/Applicability
- Implementation
- Known Uses and Related Patterns

FAÇADE PATTERN

- Motivation – Forces and Solution
- Structure
- Participants and Collaborations
- Consequences/Applicability
- Implementation
- Known Uses

PATTERNS FOR ENTERPRISE SYSTEMS

- Meeting the Challenge – Technologies
- Meeting the Challenge – Best Practices
- Some Patterns for Enterprise Systems
- Business Delegate
- Business Delegate: Solution
- Business Delegate: Structure
- Business Delegate: Consequences
- Value Object
- Value Object: Solution
- Value Object: Structure
- Value Object: Consequences
- Data Access Object (DAO)
- DAO: Solution
- DAO: Structure
- DAO: Consequences
- Lazy Load
- Lazy Load: Solution
- Lazy Load: Consequences

FACTORY METHOD PATTERN

- Motivation – Forces and Solution
- Motivation
- Factor Method: Iterator Usage
- Factory Method: General Structure
- Participants
- Collaborations and Applicability
- Applicability
- Consequences
- Implementation
- Known Uses and Related Patterns

STRATEGY PATTERN

- Motivation – Forces and Solution
- Structure
- Alternative to Strategy
- How Do We Choose Among Alternative?
- Participants
- Collaborations and Applicability
- Consequences
- Implementation
- Known Uses and Related Patterns
- Difference From Factory Method

WRAP-UP

- What Have We Done?
- So – What Do You Think About Patterns?
- Where Do We Go From Here?
- Do We Fit Into Alexander's Vision?
- Design Patterns Isn't All You Need
- Have Fun

Duración aproximada:
112 horas

Lugar:
Altadena 26 Col. Nápoles

Incluye:
Incluye material del diplomado en inglés, servicio de galletas, café, té y refrescos, estacionamiento (si el entrenamiento es en nuestras instalaciones del DF) y diploma de participación en el curso.

Formas de pago:
Este pago puede realizarse de las siguientes maneras:

1. Depósito en Banamex cuenta 4923239 Suc. 575 a nombre de Desarrollo y Capacitación en Internet, S. A. de C. V. (CLABE en caso de transferencia electrónica vía Internet 002180057549232394)
2. Cheque a nombre de Desarrollo y Capacitación en Internet, S. A. de C. V.
3. Tarjetas de Crédito Visa o Master Card