

## 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
<b>TOTAL</b>	<b>112</b>

# OBJECT-ORIENTED ANALYSIS & DESIGN USING THE UNIFIED MODELING LANGUAGE

**Objetivo:** 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.

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

## Contenido

### INTRODUCTION TO ANALYSIS AND DESIGN

- Why is Programming Hard?
- The Tasks of Software Development
- Modules
- Models
- Modeling
- Perspective
- Objects
- Change
- New Paradigms

### OBJECTS

- Encapsulation
- Abstraction
- Objects
- Classes
- Responsibilities
- Attributes
- Composite Classes
- Operations and Methods
- Visibility
- Inheritance
- Inheritance Example
- Protected and Package Visibility
- Scope
- Class Scope

### NEW MODELS IN UML 2.0

- New to UML 2.0
- Composite Structure Diagrams
- Timing Diagrams
- Interaction Overview Diagrams

### USE CASES

- Use Cases
- Use Case Diagram Components
- Use Case Diagram
- Actor Generalization
- Include and Extend
- Other Systems
- Narrative
- Template for Use Case Narrative
- Using Use Cases

### PROCESS

- Process
- Risk Management
- Test
- Reviews
- Refactoring
- History
- The Unified Process
- Agile Processes

## ADVANCED OBJECTS

- Constructors & Destructors
- Instance Creation
- Abstract Classes
- Polymorphism
- Polymorphism Example
- Multiple Inheritance
- Solving Multiple Inheritance Problems
- Interfaces
- Interfaces with Ball and Socket Notation
- Templates

## CLASSES AND THEIR RELATIONSHIPS

- Class Models
- Associations
- Multiplicity
- Qualified Associations
- Roles
- Association Classes
- Composition and Aggregation
- Using Class Models

## SEQUENCE DIAGRAMS

- Sequence Diagrams
- Interaction Frames
- Decisions
- Loops
- Creating and Destroying Objects
- Activation - 2.0
- Synchronous & Asynchronous
- The Objects Drive the Interactions
- Evaluating Sequence Diagrams
- Using Sequence Diagrams

## COMMUNICATION DIAGRAMS

- Communication Diagrams
- Communication and Class Diagrams
- Evaluating Communication Diagrams
- Using Communication Diagrams

## STATE MACHINE DIAGRAMS

- What is State?
- State Notation
- Transitions and Guards
- Registers and Actions
- More Actions
- Internal Transitions
- Superstates and Substates
- Concurrent States
- Using State Machines
- Implementation

## THE PROJECT

- Inception
- Elaboration
- Elaboration II
- Construction Iterations
- Construction Iterations - The Other Stuff

## DOMAIN ANALYSIS

- Top View - The Domain Perspective
- Data Dictionary
- Finding the Objects
- Responsibilities, Collaborators, and Attributes
- CRC Cards
- Class Models
- Use Case Models
- Other Models
- Judging the Domain Model

## REQUIREMENTS AND SPECIFICATION

- The Goals
- Understand the Problem
- Specify a Solution
- Prototyping
- The Complex User
- Other Models
- Judging the Requirements Model

## DESIGN OF OBJECTS

- Design
- Factoring
- Design of Software Objects
- Features
- Methods
- Cohesion of Objects
- Coupling between Objects
- Coupling and Visibility
- Inheritance

## SYSTEM DESIGN

- Design
- A Few Rules
- Object Creation
- Class Models
- Interaction Diagrams
- Printing the Catalog
- Printing the Catalog II
- Printing the Catalog III
- Object Links
- Associations

## ACTIVITY DIAGRAMS

- Activity Notation
- Decisions and Merges
- Synchronization
- Drilling Down
- Iteration
- Partitions
- Parameters and Pins
- Expansion Regions
- Using Activity Diagrams

## PACKAGE, COMPONENT, AND DEPLOYMENT

### DIAGRAMS

- Modeling Groups of Elements - Package Diagrams
- Visibility and Importing
- Structural Diagrams
- Components and Interfaces
- Deployment Diagram

## REFACTORING

- Refactoring
- Clues and Cues
- How to Refactor
- A Few Refactoring Patterns

## APPENDIX A - UML SYNTAX

## APPENDIX B - DESIGN BY CONTRACT

- Contracts
- Enforcing Contracts
- Inheritance and Contracts

## APPENDIX C - IMPLEMENTATIONS

- C++
- Java
- C#

# OBJECT-ORIENTED PROGRAMMING IN JAVA

**Objetivo:** 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 Java.

## OBJECTS AND CLASSES

- Defining a Class
- Creating an Object
- Instance Data and Class Data
- Methods
- Constructors
- Access Modifiers
- Encapsulation

## USING JAVA OBJECTS

- Printing to the Console
- Methods and Messages
- Parameter Passing
- Comparing and Identifying Objects
- Destroying Objects
- Using the Primitive-Type Wrapper Classes

## INHERITANCE IN JAVA

- Inheritance
- Inheritance in Java
- Casting
- Method Overriding
- Polymorphism
- super
- The Object Class

## ADVANCED INHERITANCE AND LANGUAGE CONSTRUCTS

- Enumerated Types
- Abstract Classes
- Interfaces
- Using Interfaces
- Comparable
- Collections
- Generics

# 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.

# DCInternet

## 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 IN JAVA SOFTWARE

**Objetivo:** Desarrollar un vocabulario y conocimiento de los patrones de diseño y sus mejores prácticas. Aprenderás a reconocer y aplicar patrones de diseño, esto es, cómo incorporarlos a nuestro propio análisis, diseño e implementación.

**Audiencia:** Programadores, analistas, líderes de proyecto que deseen conocer de arquitectura para aplicaciones Java.

**Prerrequisitos:** Programación Java y conocimientos de UML (Unified Modeling Language).

## Contenido

### RECOGNIZING AND APPLYING PATTERNS

- Design Patterns
- Defining a Pattern
- Unified Modeling Language
- Seeing Patterns
- Warning Signs and Pitfalls

### STRUCTURAL PATTERNS

- The Composite Pattern
- The Adapter Pattern
- The Decorator Pattern
- The Façade Pattern
- The Flyweight Pattern

### CREATIONAL PATTERNS

- Factory Patterns
- The Singleton Pattern
- APIs and Providers
- Cascading Factories

### J2EE PATTERNS

- Model/View/Controller, Redux
- The Intercepting Filter Pattern
- The Front and Application Controller Patterns
- The Business Delegate Pattern
- The Service Locator Pattern
- The Transfer Object Pattern
- The Composite Entity Pattern
- The Data Access Object Pattern

### BEHAVIORAL PATTERNS

- The Strategy Pattern
- The Template Method Pattern
- The Observer Pattern
- The Model/View/Controller Pattern
- The Command Pattern
- The Chain of Responsibility Pattern

**Duración aproximada:**

108-112 horas

**Lugar:**

Altadena 26 Col. Nápoles

**Incluye:**

*Incluye material del diplomado en inglés, mochila, servicio de galletas, café y refrescos, estacionamiento 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

# DCInternet