



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO  
FACULTAD DE INGENIERÍA



PROGRAMA DE ESTUDIO

PROGRAMACIÓN ORIENTADA A OBJETOS

1323

3

10

Asignatura

Clave

Semestre

Créditos

INGENIERÍA ELÉCTRICA

INGENIERÍA  
EN COMPUTACIÓN

INGENIERÍA  
EN COMPUTACIÓN

División

Departamento

Licenciatura

Asignatura:

Obligatoria ☒

Optativa ☐

Horas/semana:

Teóricas

Prácticas

Total

Horas/semestre:

Teóricas

Prácticas

Total

**Modalidad:** Curso teórico-práctico

**Seriación obligatoria antecedente:** Estructura de Datos y Algoritmos I

**Seriación obligatoria consecuente:** Ninguna

**Objetivo(s) del curso:**

El alumno construirá programas con el paradigma orientado a objetos, así como el diseño de abstracciones para apoyar el diseño de software y bibliotecas reusables, empleando un enfoque de pruebas sistemático.

**Temario**

NÚM.	NOMBRE	HORAS
1.	El paradigma orientado a objetos	4.0
2.	UML	12.0
3.	Tipos, expresiones y control de flujo	10.0
4.	Herencia y polimorfismo	8.0
5.	Manejo de excepciones y errores	8.0
6.	Flujo de entrada y salida	8.0
7.	Programación de hilos	4.0
8.	Introducción a patrones	10.0
		64.0
	Actividades prácticas	32.0
	Total	96.0

## 1 El paradigma orientado a objetos

**Objetivo:** El alumno interpretará los conceptos de la programación orientada a objetos para aplicarlo a eventos del mundo real.

**Contenido:**

- 1.1 Elementos básicos del paradigma orientado a objetos.
  - 1.1.1 Tipos de datos primitivos y abstractos.
  - 1.1.2 Objetos.
- 1.2 Propiedades básicas del paradigma orientado a objetos.
  - 1.2.1 Abstracción.
  - 1.2.2 Cohesión.
  - 1.2.3 Encapsulamiento.
  - 1.2.4 Modularidad.
  - 1.2.5 Herencia.
  - 1.2.6 Polimorfismo.
  - 1.2.7 Acoplamiento.
  - 1.2.8 Jerarquía de clases.

## 2 UML

**Objetivo:** El alumno clasificará las diferentes vistas en el diseño orientado a objetos para aplicarlo en la solución de problemas.

**Contenido:**

- 2.1 Diseño estático.
- 2.2 Diseño dinámico.

## 3 Tipos, expresiones y control de flujo

**Objetivo:** El alumno aplicará las técnicas y herramientas de la programación orientada a objetos para la solución de problemas.

**Contenido:**

- 3.1 Generalidades.
  - 3.1.1 Identificadores.
  - 3.1.2 Palabras reservadas.
  - 3.1.3 Comentarios.
  - 3.1.4 Descripción de una clase.
  - 3.1.5 Descripción de un objeto.
- 3.2 Tipos de datos.
  - 3.2.1 Primitivos y su jerarquía.
  - 3.2.2 Referencias o instancias.
  - 3.2.3 Conversiones entre tipos primitivos (moldeado o casting).
  - 3.2.4 Operadores aritméticos.
  - 3.2.5 Operadores de asignación.
  - 3.2.6 Operadores relacionales.
  - 3.2.7 Operadores especiales (in/decremento (post o pre), concatenación, acceso a variables y métodos y de agrupación).
  - 3.2.8 Operadores a nivel de bits.

### 3.2.9 Operadores lógicos.

### 3.3 Arreglos.

### 3.4 Tipos y ámbito de las variables.

#### 3.4.1 Elementos estáticos.

#### 3.4.2 Elementos constantes.

### 3.5 Tipos de clases (públicas, sin modificador, abstractas, finales e internas).

### 3.6 Estructuras de selección.

#### 3.6.1 Estructura if-else.

#### 3.6.2 Estructura switch-case.

#### 3.6.3 Estructura ternaria.

### 3.7 Estructuras de selección

#### 3.7.1 Estructura while.

#### 3.7.2 Estructura do-while.

#### 3.7.3 Estructura for.

## 4 Herencia y polimorfismo

**Objetivo:** El alumno aplicará las diferentes propiedades de la programación orientada a objetos para la resolución de problemas.

**Contenido:**

#### 4.1 Herencia.

#### 4.2 Método constructor.

#### 4.3 Polimorfismo (moldeado o casting entre tipos referencia o instancias).

#### 4.4 Referencias a this y a la clase base.

#### 4.5 Modificadores de acceso (encapsulamiento).

#### 4.6 Tipos de clases: abstractas, comunes y finales.

#### 4.7 Interfaces.

#### 4.8 Paquetes y documentación.

## 5 Manejo de excepciones y errores

**Objetivo:** El alumno clasificará los diferentes tipos de errores y excepciones para generar programas y aplicaciones con calidad.

**Contenido:**

#### 5.1 Definición y diferencia entre error y excepción.

#### 5.2 Jerarquía de clases de errores.

#### 5.3 Estructura try-catch-finally.

#### 5.4 Manejo de errores y excepciones.

## 6 Flujo de entrada y salida

**Objetivo:** El alumno construirá programas con el principio de flujo de entrada y salida para procesar información a partir de un problema resuelto.

**Contenido:**

#### 6.1 Fundamentos de entrada y salida.

#### 6.2 Jerarquía de clases de los flujos de datos.

#### 6.3 Manipulación de archivos y carpetas.

6.4 Flujos de entrada de datos.

6.4.1 Lectura de archivo.

6.4.2 Lectura de teclado.

6.5 Flujos de salida de datos (escritura de archivo).

6.6 Procesamiento del flujo.

## 7 Programación de hilos

**Objetivo:** El alumno aplicará los conceptos avanzados de la programación orientada a objetos para la resolución de problemas complejos.

**Contenido:**

7.1 Definición de hilo.

7.2 Ciclo de vida del hilo.

7.3 Control básico del hilo.

7.4 Clases para el manejo de hilos.

7.5 Planificador y prioridad.

7.6 Métodos sincronizados.

## 8 Introducción a patrones

**Objetivo:** El alumno aplicará los patrones de diseño adecuados para aplicarlo la resolución de problema de ingeniería.

**Contenido:**

8.1 Definición de patrón de diseño.

8.2 Diseñando problemas.

8.3 Patrones de creación.

8.4 Patrones estructurales.

8.5 Patrones de comportamiento.

---

### Bibliografía básica

### Temas para los que se recomienda:

DEITEL, Paul, DEITEL, Harvey

*Java How to Program (early objects) plus MyProgrammingLab*  
with Pearson eText 9th edition

New Jersey

Prentice Hall, 2011

Todos

DEITEL, Paul, DEITEL, Harvey

*C++ How to Program*

8th edition

New Jersey

Prentice Hall, 2011

Todos

DEITEL, Paul, DEITEL, Harvey

*C# 2010 for Programmers*

4th edition

New Jersey

Prentice Hall, 2010

Todos

GAMMA, Erich, HELM, Richard, et al. <i>Design Patterns: Elements of Reusable Object-Oriented Software</i> Boston Addison-Wesley Professional, 1994	8
GOMAA, Hassan <i>Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures</i> Washington Cambridge University Press, 2011	2, 8
LARMAN, Craig <i>Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development</i> 3rd edition New Jersey Prentice Hall, 2004	2, 8
MILES, Russ, HAMILTON, Kim <i>Learning UML 2.0</i> Boston O Reilly Media, 2006	2
OAKS, Scott, WONG, Henry <i>Java Threads</i> 3rd edition Boston O Reilly Media, 2004	7
SARANG, Poornachandras <i>Java Programming (Oracle Press)</i> Boston McGraw-Hill Osborne Media, 2012	Todos
SZNAJDLEDER, Pablo <i>Algoritmos a fondo: con implementación en C y JAVA</i> Buenos Aires Alfaomega, 2012	Todos

**Bibliografía complementaria****Temas para los que se recomienda:**

ARLOW, Jim, NEUSTADT, Ila <i>UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design</i> 2nd edition Boston Addison-Wesley Professional, 2005	2
---	---

FLANAGAN, David

*Java In A Nutshell*

5th edition

New Jersey

O Reilly Media, 2005

Todos

FOWLER, Martin

*UML Distilled: A Brief Guide to the Standard Object*

*Modeling Language* 3th edition

Washington

Addison-Wesley Professional, 2003

2

**Sugerencias didácticas**

Exposición oral	<input checked="" type="checkbox"/>
Exposición audiovisual	<input checked="" type="checkbox"/>
Ejercicios dentro de clase	<input checked="" type="checkbox"/>
Ejercicios fuera del aula	<input checked="" type="checkbox"/>
Seminarios	<input type="checkbox"/>
Uso de software especializado	<input type="checkbox"/>
Uso de plataformas educativas	<input type="checkbox"/>

Lecturas obligatorias	<input checked="" type="checkbox"/>
Trabajos de investigación	<input checked="" type="checkbox"/>
Prácticas de taller o laboratorio	<input checked="" type="checkbox"/>
Prácticas de campo	<input type="checkbox"/>
Búsqueda especializada en internet	<input type="checkbox"/>
Uso de redes sociales con fines académicos	<input type="checkbox"/>

**Forma de evaluar**

Exámenes parciales	<input checked="" type="checkbox"/>
Exámenes finales	<input checked="" type="checkbox"/>
Trabajos y tareas fuera del aula	<input checked="" type="checkbox"/>

Participación en clase	<input checked="" type="checkbox"/>
Asistencia a prácticas	<input checked="" type="checkbox"/>

**Perfil profesiográfico de quienes pueden impartir la asignatura**

Licenciatura en Ingeniería en Computación, Ciencias de Computación, Matemáticas Aplicadas o una carrera similar. Deseable haber realizado estudios de posgrado, contar con conocimientos y experiencia en el área de ciencias de la computación, contar con experiencia docente o haber participado en cursos o seminario de iniciación en la práctica docente.