ORIENTACIÓN A OBJETOS 1

Año 2026

Carrera/ Plan:

Licenciatura en Informática Plan 2021/Plan 2015
Licenciatura en Sistemas Plan 2021/Plan 2015
Analista Programador Universitario Plan 2021/Plan 2015
Analista en Tecnologías de la Información y la Comunicación Plan 2021/Plan 2017
Año: 2°
Régimen de Cursada: *Semestral*
Carácter (Obligatoria/Optativa): Obligatoria
Correlativas: Taller de Programación
Profesor/es: Alejandro Fernandez, Diego Torres, Gabriela Pérez
Hs. Semanales de teoría: 3 hrs
Hs. Semanales de práctica: 3 hrs

FUNDAMENTACIÓN

La Programación Orientada a Objetos (POO) es fundamental en la formación del profesional informático. Aporta técnicas de análisis, diseño, programación y prueba que combinan los principios de abstracción, modularización, encapsulamiento, polimorfismo, y herencia para dar lugar a una descomposición en módulos (objetos) con alta cohesión y bajo acoplamiento. Esto favorece la modificabilidad, mantenimiento y reuso de los componentes del software.

OBJETIVOS GENERALES

Presentar formalmente el paradigma de orientación a objetos, sus características, ventajas y aplicaciones dentro del desarrollo de sistemas de software. Desarrollar prácticas concretas con lenguajes orientados a Objetos. Establecer metodologías de análisis y diseño orientados a objetos.

RESULTADOS DE APRENDIZAJE

1.2. Describir las características de los últimos avances en hardware y software y sus correspondientes aplicaciones prácticas (Básico).
1.3. Describir los avances informáticos actuales e históricos y demostrar cierta visión sobre tendencias y avances futuros (Básico).
1.4. Aplicar e integrar conocimientos de otras disciplinas informáticas como apoyo al estudio de la propia área de especialidad (o áreas de especialidad) (Básico).
1.5. Demostrar sensibilización ante la necesidad de contar con amplios conocimientos a la hora de crear aplicaciones informáticas en otras áreas temáticas (Básico).
2.1. Utilizar una serie de técnicas con las que identificar las necesidades de problemas reales, analizar su complejidad y evaluar la viabilidad de las posibles soluciones mediante técnicas informáticas (Adecuado).
2.2. Describir un determinado problema y su solución a varios niveles de abstracción (Adecuado).
2.3. Seleccionar y utilizar los correspondientes métodos analíticos, de simulación y de modelización (Básico).
2.4. Escoger los patrones de solución, algoritmos y estructuras de datos apropiados (Adecuado).
3.1. Definir y diseñar hardware/software informático/de red que cumpla con los requisitos establecidos (Básico).
3.2. Describir las fases implicadas en distintos modelos de ciclo de vida con respecto a la definición, construcción, análisis y puesta en marcha de nuevos sistemas y el mantenimiento de sistemas existentes (Básico).

3.3. Elegir y utilizar modelos de proceso adecuados, entornos de programación y técnicas de gestión de datos con respecto a proyectos que impliquen aplicaciones tradicionales así como aplicaciones emergentes (Básico).
3.4. Describir y explicar el diseño de sistemas e interfaces para interacción persona-ordenador y ordenador-ordenador (Básico).
5.2. Describir y explicar las técnicas de gestión correspondientes al diseño, implementación, análisis, uso y mantenimiento de sistemas informáticos, incluyendo gestión de proyectos, de configuración y de cambios, así como las técnicas de automatización correspondientes (Adecuado).

COMPETENCIAS

- CGS6- Capacidad para interpretar la evolución de la Informática con una visión de las tendencias tecnológicas futuras.
- CGT1- Identificar, formular y resolver problemas de Informática.
- CGT2- Concebir, diseñar y desarrollar proyectos de Informática.
- CGT5- Utilizar de manera efectiva las técnicas y herramientas de aplicación de la Informática.
- CGT8 Capacidad de interpretación y resolución de problemas multidisciplinarios, desde los conocimientos de la disciplina informática.
- LI - CE4 – Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real, especificación formal de los mismos, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software/sistemas de información que se ejecuten sobre equipos de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfaces humano computador y computador-computador.
- LS - CE1 – Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real. Especificación formal, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software que se ejecuten sobre sistemas de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfaces humano computador y computador-computador.

CONTENIDOS MINIMOS (de acuerdo al Plan de Estudios)

- Objetos.
- Clases e instancias.
- Encapsulamiento.
- Jerarquías de clase.
- Herencia. Polimorfismo.
- Lenguajes y aplicaciones.

PROGRAMA ANALÍTICO

Unidad 1: La crisis del software. Problemas de las técnicas tradicionales (procedurales). Resolución de problemas complejos. El problema de la extensibilidad, el reuso y el mantenimiento. Cohesión y acoplamiento como criterios rectores.

Unidad 2: Principios del paradigma de Orientación a Objetos: clase, instancia, mensaje, método, atributos, relaciones entre objetos, ocultamiento, binding dinámico, inicialización, igualdad e identidad.

Unidad 3: El programa OO. El modelo de dominio en el programa como servicio y en la aplicación interactiva.

Unidad 4: Herencia, redefinición y extensión de comportamiento heredado. Principio de "es-un". Heurísticas y buenas prácticas.

Unidad 5: El polimorfismo como herramienta de diseño. Relación entre tipado de variables y polimorfismo. Las clases como tipos. Las Interfaces como tipos. Heurísticas y buenas prácticas.

Unidad 6: Delegación de responsabilidades vía envío de mensajes. Delegación a self/this para simplificar el mantenimiento y potenciar el reuso. Heurísticas y buenas prácticas.

Unidad 7: Convenciones para aumentar la legibilidad, y simplificar el mantenimiento y reuso. Elección de nombres, y organización del código.

Unidad 8: UML (Lenguaje Unificado de Modelado). Diagramas de clases. Diagramas de secuencia. Diagrama de Casos de uso. Correspondencia entre elementos UML y elementos de lenguajes de programación.

Unidad 9: Introducción al análisis y diseño de software con RUP (Proceso Unificado). La fase de elaboración: derivación del diseño y la implementación a partir de los casos de uso.

Unidad 10: Heurísticas y patrones para identificar clases y asignar adecuadamente responsabilidades: Expertos, Creadores, Controladores, Bajo Acoplamiento y Alta Cohesión. Los principios SOLID.

Unidad 11: Particularidades y variantes de lenguajes de programación OO. Correspondencia entre elementos del paradigma y construcciones del lenguaje. Lenguajes basados en clases y basados en instancias. Tipado estático y dinámico.

Unidad 12: Los Tests Unitarios como mecanismo de aseguramiento de calidad y como guía al proceso de desarrollo. Automatización de test de unidad con herramientas de la familia xUnit .

BIBLIOGRAFIA

Bibliografía Obligatoria

Craig Larman. UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. 2da. Edición. Prentice Hall. (2003). ISBN 978-84-205-3438-1

Weisfeld, Matt. The Object-Oriented Thought Process. 5th edition. Hoboken, NJ: Pearson Education, Inc, 2019. ISBN 978-0-13-518196-6

Budd, Timothy. An Introduction to Object-Oriented Programming (3rd Edition). Addison Wesley; 3 edition (2001), ISBN-10: 0201760312

Evans, Eric. Domain-Driven Design: Tackling Complexity in the Heart of Software. 1st edition. Boston: Addison-Wesley Professional, 2003.

Fowler, Martin. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley Professional, 2003. ISBN: 978-0-321-19368-1

Bibliografía complementaria

Vernon, Vaughn. *Domain-Driven Design Distilled*. Boston: Addison-Wesley, 2016.

Fowler. UML Gota a Gota. Addison Wesley Longman, 2000. ISBN: 968-444-364-1

Wirfs-Brock, Rebecca, Brian Wilkerson, and Lauren Wiener. Designing Object-Oriented Software. 1st edition. Englewood Cliffs, N.J: Pearson, 1990.

Designing Object-Oriented Software. Rebecca Wirfs-Brock, Brian Wilkerson (Contributor), Lauren Wiener. Prentice Hall PTR; (January 1991), ISBN 0136298257

Martin, Robert C., ed. Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River, NJ: Prentice Hall, 2009. ISBN 978-0-13-235088-4

METODOLOGÍA DE ENSEÑANZA

Expectativas de logro

Presentar formalmente el paradigma de objetos, sus características, ventajas y aplicaciones dentro del desarrollo de sistemas de software. Desarrollar prácticas concretas con lenguajes orientados a Objetos. Establecer metodologías de análisis y diseño orientados a objetos.

Existen tres objetivos particulares:

Diseñar y Escribir Programas Orientados a Objetos

Este objetivo implica:

- Adquirir los conocimientos teóricos básicos de la Programación Orientada a Objetos.
- Experimentar con los conceptos teóricos en situaciones prácticas donde manifiesten las ventajas del paradigma.
- Familiarizarse con un lenguaje y ambiente de desarrollo orientado a objetos, incluyendo herramientas de inspección, depuración, y crítica.
- Identificar clases y asignar adecuadamente responsabilidades, en el marco del proceso unificado, a partir de una especificación de requerimientos en formato de casos de uso breves

Manejo Adecuado de una Notación

Es necesario contar con alguna notación que facilite la comunicación, documentación y desarrollo de un diseño orientado a objetos. Se introducirá también el uso de un lenguaje de modelado gráfico orientado a objetos (UML), que le permitirá construir diagramas especificando distintos aspectos de un sistema.

Esta notación también debe acompañar el proceso de desarrollo de sistemas orientado a objetos como es el proceso de desarrollo unificado basado en UML (RUP).

Dentro de este objetivo también se encuentra la familiarización con un ambiente que soporte el diseño de diagramas UML e incluso el proceso de desarrollo.

Procedimientos didácticos

En función de los objetivos planteados se propone organizar el dictado de la materia en clases teórico-prácticas organizadas de la siguiente manera.

Clases Teóricas:

Las clases teóricas están destinadas a presentar los conceptos teóricos del programa de la materia. Los conceptos teóricos deben ser presentados a través de su motivación, su definición, relación e interacción con los demás conceptos. Estas actividades deben ser fuertemente soportadas por el uso de ejemplos concretos.

Clase practicas

Las clases prácticas deben ser dedicadas a aplicar los conceptos teóricos impartidos en las clases teóricas. Las mismas deben ser guiadas por un trabajo práctico. Cada trabajo práctico enfoca una temática y un conjunto de objetivos teóricos-prácticos a lograr con las ejercitaciones planteadas.

El desarrollo de la clase práctica debe contar con una explicación del trabajo práctico por el auxiliar docente a cargo, donde se le indiquen al alumno los objetivos de la práctica y los conceptos teóricos que se pretenden aplicar, más un conjunto de guías para la resolución de los problemas planteados. Luego, los alumnos desarrollan la práctica llevando a cabo cada ejercitación y contando permanentemente con la posibilidad de trabajar en conjunto con un auxiliar docente que guíe su trabajo y evacue sus dudas.

En la cátedra se plantean actividades planificadas para los alumnos y se les propone resolverlas con las herramientas que cuentan al llegar a la materia (por ejemplo, lenguajes y metodologías no-orientadas a objetos a objetos). Se los “desafía” a presentar la posible evolución de la solución para ese tipo de problema y en podría mejorarse la solución/soluciones actuales - dicha actividad genera el contexto adecuado para luego introducir las propuestas de la POO a dichos desafíos (y así buscar un aprendizaje significativo)

Si bien cada edición del curso se apoya en una selección de tecnologías en particular (por ejemplo, el lenguaje de programación Java), se mencionan otras alternativas (algunas cubiertas en otras asignaturas, otras que los alumnos pueden conocer de fuentes on-line) y se contrasta con ellas. Esto lleva al alumno a buscar bibliografía relacionada con los cambios tecnológicos relativos a la POO y formarse un criterio sobre las tendencias (por ejemplo, los lenguajes dinámicos, la programación con prototipos, etc.). La cátedra acompaña al alumno en el proceso de interpretación evolutiva de la disciplina, para contrastar sus conclusiones y validar su habilidad para esta competencia.

Los ejercicios de los trabajos prácticos serán diseñados de manera que los mismos impliquen el diseño de un programa, su implementación en un lenguaje OO y su testeo a través del uso de técnicas de testeo de unidad. En la cátedra se pone énfasis en el proceso de identificación de problemas del mundo real, especificación de los mismos como problemas resolubles desde la informática y en el desarrollo de soluciones verificables para los mismos. Se tratan proyectos multidisciplinarios (los requerimientos y dominios de aplicación tomados como ejemplo provienen de muchas disciplinas). Además de ofrecer ejemplos motivadores que sirvan como práctica de las temáticas de la asignatura, se trata de acompañar al alumno (aprovechando la experiencia de trabajo multidisciplinar del equipo docente) en la interpretación del rol del Informático como articulador de soluciones en áreas de conocimiento muy diferentes que requieren participación de expertos extra-disciplinares. Para la especificación y el diseño del programa se usarán los recursos del lenguaje de modelado UML. UML es un lenguaje gráfico con sintaxis formalmente definida, a través de la técnica del metamodelado. Esto permite contar con una especificación precisa del sistema bajo desarrollo. Para el testeo, cada práctica será acompañada por un test de unidad.

Cada trabajo práctico concluye con el visado de ejercicios de manera que el alumno reciba de parte de los docentes comentarios que permitan retroalimentar su evolución en el proceso de aprendizaje. Un trabajo integrador (y otras entregas grupales) ponen en juego y fortalecen la capacidad del alumno para gestionar, planificar, ejecutar y controlar proyectos de Informática (a pequeña escala). El grado con el que alcanza estas competencias se refleja directamente en el resultado del proyecto integrador y demás entregas.

En cuanto a las interfaces humano computador, en la materia se hace una primera aproximación a la construcción de aplicaciones interactivas orientadas a objetos (que se reforzará en Orientación a Objetos 2), haciendo énfasis en la importancia de la separación modelo-vista, característica fundamental para poder incorporar las competencias de diseño de interacción sin que se vean atadas a la evolución de la lógica del dominio.

EVALUACIÓN

La evaluación de las competencias generales y específicas a cuya adquisición esta cátedra contribuye, tiene lugar en dos instancias principales: parcial (principalmente enfocada en aspectos prácticos) y final (principalmente enfocada en aspectos teóricos e integración de contenidos).

La **evaluación parcial** tiene lugar por medio de exámenes prácticos y entrega de trabajos (grupales e individuales). Estas evaluaciones cuentan con, al menos, una instancia de recuperación.

La **evaluación final** consiste en un examen, que podrá ser escrito, oral, o combinar ambas modalidades.

La asignatura ofrece un mecanismo de **promoción opcional** que, en caso de ser completado y aprobado, cuenta como evaluación final y determina la nota final. Para acceder a la promoción los alumnos deberán completar cuestionarios intermedios, asistir al 70% de las consultas de práctica, aprobar las evaluaciones parciales en primera fecha o en primer recuperatorio, y aprobar un examen integrador de promoción. La calificación final de promoción se obtiene de una combinación entre el desempeño en las evaluaciones parciales y en el examen de promoción.

CRONOGRAMA DE CLASES Y EVALUACIONES

Clase	Fecha	Contenidos/Actividades
1	Semana del 17/08/2026	Organización de la materia. Motivación de la programación OO. Modularización (cohesión y acoplamiento). Abstracción. Clase, objeto/instancia, atributo, método, mensaje. Variables como punteros a objetos. Encapsulamiento y ocultamiendo de información. Binding dinámico y method lookup (enviar mensajes vs invocar métodos). Creación de objetos, inicialización, constructores. Identidad vs igualdad. La pseudovariable this (y su uso para descomponer métodos). Intuición de las colecciones como objetos.
2	Semana del 24/08/2026	Objetos que conocen a otros. Interfaces y su rol en el chequeo de tipos. Polimorfismo. Composición de objetos. Relaciónes uno a muchos con List y ArrayList. Delegación como estrategia de diseño. El sistema orientado a objetos: UI/Servicio + modelo/logica de dominio (nuestros objetos nunca hacen I/O) + tests automatizados. Arquitectura de referencia para Orientación a Objetos 1.
3	Semana del 31/08/2026	Herencia de conocimiento y comportamiento. Binding dinámico en el contexto de herencia. Generalizar y especializar. Clases abstractas y clases concretas. Pseudo variables super y this para aprovechar, redefinir, y extender comportamiento heredado. Chequeo de tipos y herencia.
4	Semana del 7/09/2026	Colecciones heterogéneas de objetos; polimorfismo entre colecciones y polimorfismo entre los elementos en una colección. Iteradores de colecciones (y objetos que abstraen y/o soportan el comportamiento de iteración). Clausuras léxicas en objetos. UML (Lenguaje Unificado de Modelado). Diagramas de clases. Diagramas de secuencia. Diagrama de Casos de uso. Correspondencia entre elementos UML y elementos de lenguajes de programación. Editores gráficos vs. Editores UML
5	Semana del 14/09/2026	Identificar clases y responsabilidades a partir de una especificación. Modelo conceptual (clases candidatas). Pre y post contratos (que hay antes y que queda después) para los casos de uso. Derivar métodos a partir de los pre y post contratos.
6	Semana del 21/09/2026	Identificar clases y responsabilidades a partir de una especificación. Modelo conceptual (clases candidatas). Pre y post contratos (que hay antes y que queda después) para los casos de uso. Derivar métodos a partir de los pre y post contratos. (Continuación)
7	Semana del 28/09/2026	Los Tests Unitarios como mecanismo de aseguramiento de calidad y como guía al proceso de desarrollo. Automatización de test de unidad con herramientas de la familia xUnit. Pruebas de particiones equivalentes y valores de borde.
8	Semana del 5/10/2026	Herencia vs Composición. Compara las ventajas y limitaciones de herencia y composición, mostrando ejemplos prácticos, buenas prácticas y criterios de diseño que fomenten la extensibilidad y mantenibilidad.
9	Semana del 12/10/2026	Taller de diseño (aplicamos todo lo visto a una nueva especificación). Identificamos clases, determinamos responsabilidades, y discutimos diseños alternativos. Obtenemos nuestro primer modelo de clases candidatas y programamos los casos de uso más simples.
10	Semana del 19/10/2026	Particularidades de otros lenguajes de programación OO. Lenguajes basados en clases y basados en instancias. El rol del tipado Tipado estático y dinámico.

11	Semana del 26/10/2026	Arquitectura de la aplicación / servicio. Separación en capas. La importancia de un buen modelo de dominio. Introducción a la persistencia de objetos
12	Semana del 02/11/2026	Evaluación parcial.
13	Semana del 9/11/2026	Consultas.
14	Semana del 16/11/2026	Consultas.
15	Semana del 23/11/2026	Evaluación parcial (1º recuperatorio)
16	Semana del 30/11/2026	Consultas
17	Semana del 7/12/2026	Evaluación parcial (2º recuperatorio). Examen integrador de Promoción.

Evaluaciones previstas	Fecha
Evaluación parcial	Semana del 2/11
Recuperatorio de la evaluación parcial	Semana del 23/11
Recuperatorio de la evaluación parcial	Semana del 7/12
Examen integrador de promoción	Semana del 7/12

Contacto de la cátedra (mail, sitio WEB, plataforma virtual de gestión de cursos):

alejandro.fernandez@lifia.info.unlp.edu.ar
diego.torres@lifia.info.unlp.edu.ar
gabriela.perez@lifia.info.unlp.edu.ar

Firma del/los profesor/es

Alejandro Fernandez

ORIENTACIÓN A OBJETOS 1
Redictado

Año 2026

Carrera/ Plan: *(Dejar lo que corresponda)*

Licenciatura en Informática Plan 2015/Plan 2021
Licenciatura en Sistemas Plan 2015/ Plan 2021
Analista Programador Universitario Plan 2015/Plan 2021
Analista en Tecnologías de la Información y la Comunicación Plan 2017/Plan 2021
Año: 2026
Régimen de Cursada: *Semestral*
Carácter (Obligatoria/Optativa): Obligatoria
Correlativas: Taller de Programación
Profesor/es: Andrés Rodriguez
Hs. Semanales teoría: 1 hrs
Hs. Semanales práctica: 5 hrs

FUNDAMENTACIÓN

La Programación Orientada a Objetos (POO) es fundamental en la formación del profesional informático. Aporta técnicas de análisis, diseño, programación y prueba que combinan los principios de abstracción, modularización, encapsulamiento, polimorfismo, y herencia para dar lugar a una descomposición en módulos (objetos) con alta cohesión y bajo acoplamiento. Esto favorece la modificabilidad, mantenimiento y reuso de los componentes del software.

OBJETIVOS GENERALES

Presentar formalmente el paradigma de objetos, sus características, ventajas y aplicaciones dentro del desarrollo de sistemas de software. Desarrollar prácticas concretas con lenguajes orientados a Objetos. Establecer metodologías de análisis y diseño orientados a objetos.

RESULTADOS DE APRENDIZAJE

1.2. Describir las características de los últimos avances en hardware y software y sus correspondientes aplicaciones prácticas (Básico).

1.3. Describir los avances informáticos actuales e históricos y demostrar cierta visión sobre tendencias y avances futuros (Básico).

1.4. Aplicar e integrar conocimientos de otras disciplinas informáticas como apoyo al estudio de la propia área de especialidad (o áreas de especialidad) (Básico).

1.5. Demostrar sensibilización ante la necesidad de contar con amplios conocimientos a la hora de crear aplicaciones informáticas en otras áreas temáticas (Básico).

2.1. Utilizar una serie de técnicas con las que identificar las necesidades de problemas reales, analizar su complejidad y evaluar la viabilidad de las posibles soluciones mediante técnicas informáticas (Adecuado).

2.2. Describir un determinado problema y su solución a varios niveles de abstracción (Adecuado).

2.3. Seleccionar y utilizar los correspondientes métodos analíticos, de simulación y de modelización (Básico).

2.4. Escoger los patrones de solución, algoritmos y estructuras de datos apropiados (Adecuado)

3.1. Definir y diseñar hardware/software informático/de red que cumpla con los requisitos establecidos (Básico).

3.2. Describir las fases implicadas en distintos modelos de ciclo de vida con respecto a la definición, construcción, análisis y puesta en marcha de nuevos sistemas y el mantenimiento de sistemas existentes (Básico).

3.3. Elegir y utilizar modelos de proceso adecuados, entornos de programación y técnicas de gestión de datos con respecto a proyectos que impliquen aplicaciones tradicionales, así como aplicaciones emergentes (Básico).

3.4. Describir y explicar el diseño de sistemas e interfaces para interacción persona-ordenador y ordenador-ordenador (Básico).

5.2. Describir y explicar las técnicas de gestión correspondientes al diseño, implementación, análisis, uso y mantenimiento de sistemas informáticos, incluyendo gestión de proyectos, de configuración y de cambios, así como las técnicas de automatización correspondientes (Adecuado).

COMPETENCIAS

CGS6- Capacidad para interpretar la evolución de la Informática con una visión de las tendencias tecnológicas futuras.
CGT1- Identificar, formular y resolver problemas de Informática.
CGT2- Concebir, diseñar y desarrollar proyectos de Informática.
CGT5- Utilizar de manera efectiva las técnicas y herramientas de aplicación de la Informática.
CGT8 Capacidad de interpretación y resolución de problemas multidisciplinarios, desde los conocimientos de la disciplina informática.
LI - CE4 – Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real, especificación formal de los mismos, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software/sistemas de información que se ejecuten sobre equipos de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfaces humano computador y computador-computador.
LS - CE1 – Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real. Especificación formal, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software que se ejecuten sobre sistemas de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfaces humano computador y computador-computador.

CONTENIDOS MINIMOS (de acuerdo al Plan de Estudios)

- Objetos.
- Clases e instancias.
- Encapsulamiento.
- Jerarquías de clase.
- Herencia. Polimorfismo.
- Lenguajes y aplicaciones.

PROGRAMA ANALÍTICO

Unidad 1: La crisis del software. Problemas de las técnicas tradicionales (procedurales). Resolución de problemas complejos. El problema de la extensibilidad, el reuso y el mantenimiento. Cohesión y acoplamiento como criterios rectores.

Unidad 2: Principios del paradigma de Orientación a Objetos: clase, instancia, mensaje, método, atributos, relaciones entre objetos, ocultamiento, binding dinámico, inicialización, igualdad e identidad.

Unidad 3: El programa OO. El modelo de dominio en el programa como servicio y en la aplicación interactiva.

Unidad 4: Herencia, redefinición y extensión de comportamiento heredado. Principio de "es-un". Heurísticas y buenas prácticas.

Unidad 5: El polimorfismo como herramienta de diseño. Relación entre tipado de variables y polimorfismo. Las clases como tipos. Las Interfaces como tipos. Heurísticas y buenas prácticas.

Unidad 6: Delegación de responsabilidades vía envío de mensajes. Delegación a self/this para simplificar el mantenimiento y potenciar el reuso. Heurísticas y buenas prácticas.

Unidad 7: Convenciones para aumentar la legibilidad, y simplificar el mantenimiento y reuso. Elección de nombres, y organización del código.

Unidad 8: UML (Lenguaje Unificado de Modelado). Diagramas de clases. Diagramas de secuencia. Diagrama de Casos de uso. Correspondencia entre elementos UML y elementos de lenguajes de programación.

Unidad 9: Introducción al análisis y diseño de software con RUP (Proceso Unificado). La fase de elaboración: derivación del diseño y la implementación a partir de los casos de uso.

Unidad 10: Heurísticas y patrones para identificar clases y asignar adecuadamente responsabilidades: Expertos, Creadores, Controladores, Bajo Acoplamiento y Alta Cohesión. Los principios SOLID.

Unidad 11: Particularidades y variantes de lenguajes de programación OO. Correspondencia entre elementos del paradigma y construcciones del lenguaje. Lenguajes basados en clases y basados en instancias. Tipado estático y dinámico.

Unidad 12: Los Tests Unitarios como mecanismo de aseguramiento de calidad y como guía al proceso de desarrollo. Automatización de test de unidad con herramientas de la familia xUnit .

BIBLIOGRAFIA

Bibliografía Obligatoria

Weisfeld, Matt. The Object-Oriented Thought Process. 5th edition. Hoboken, NJ: Pearson Education, Inc, 2019. ISBN 978-0-13-518196-6

Martin, Robert C., ed. Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River, NJ: Prentice Hall, 2009. ISBN 978-0-13-235088-4

Martin, Robert C. Agile Software Development, Principles, Patterns, and Practices. 1. ed., Pearson new international ed. Harlow: Pearson, 2014. 978-1-292-02594-0

Craig Larman. UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. 2da. Edición. Prentice Hall. (2003). ISBN 978-84-205-3438-1

Fowler, Martin. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley Professional, 2003. ISBN: 978-0-321-19368-1

Jacobson, I., Booch, G Rumbaugh, J., The Unified Software Development Process. Addison Wesley. 1999. ISBN 0-201-57169-2

Bibliografía complementaria

Fowler. UML Gota a Gota. Addison Wesley Longman, 2000. ISBN: 968-444-364-1

Wirfs-Brock, Rebecca, Brian Wilkerson, and Lauren Wiener. Designing Object-Oriented Software. 1st edition. Englewood Cliffs, N.J: Pearson, 1990.

Designing Object-Oriented Software. Rebecca Wirfs-Brock, Brian Wilkerson (Contributor), Lauren Wiener. Prentice Hall PTR; (January 1991), ISBN 0136298257

Budd, Timothy. An Introduction to Object-Oriented Programming(3rd Edition). Addison Wesley; 3 edition (2001), ISBN-10: 0201760312

CONDICIONES DE CURSADA

Podrán acceder a este redictado aquellas personas que hayan cursado Orientación a Objetos 1 en el segundo semestre 2025 y hayan rendido el parcial el cualquiera de sus fechas obteniendo una nota de desaprobado (sin haber entregado en blanco y dando evidencia un manejo mínimo de los conceptos dados en la materia).

METODOLOGÍA DE ENSEÑANZA

El curso está dirigido a alumnos que previamente cursaron Orientación a Objetos I, por lo cual, durante el redictado, se hará más énfasis en las actividades prácticas, repasando y reforzando los conceptos teóricos.

Habrá una clase semanal (en dos turnos: mañana y tarde), en donde se presentarán ejercicios con el fin de repasar los conceptos teóricos necesarios para resolverlos. Luego del repaso, los alumnos deberán hacer consultas sobre los ejercicios de las actividades prácticas para poder resolverlos.

Todo el material de presentación y resolución quedará disponible en la plataforma virtual de enseñanza de la materia.

Se dispondrá de una plataforma para que los alumnos puedan estar en contacto y obtener respuesta de los docentes en todo momento.

EVALUACIÓN

El curso se aprueba con un parcial que posee dos recuperatorios. Aprobado el curso, los alumnos pueden realizar un trabajo con el fin de promocionar el final. El trabajo posee una fecha de entrega, y los alumnos podrán hacer consultas durante el desarrollo. Finalmente, en la entrega deberán realizar un coloquio para defender el trabajo.

CRONOGRAMA DE CLASES Y EVALUACIONES

Clase	Fecha	Contenidos/Actividades
1	Semana del 9/03/2026	Orientación a Objetos. Clase, objeto/instancia, atributo, método, mensaje. Objetos que conocen a objetos. Variables como punteros. Creación de objetos, Inicialización. El sistema orientado a objetos: UI/Servicio + modelo (nuestros objetos nunca hacen I/O). Modularización. ¿Por qué lo hacemos en objetos? self/this como estrategia para evitar duplicación, aprovechar comportamiento heredado, acortar métodos.
2	Semana del 16/03/2026	Herencia: Heredar, redefinir, y extender comportamiento heredado. El keyword super. Binding dinámico en el contexto de herencia. Las colecciones como objetos.

		Polimorfismo. Delegación como estrategia de diseño. La importancia del encapsulamiento. Introducción a diagramas de clase de UML.
3	Semana del 23/03/2026	Aplicamos todo lo que vimos hasta ahora en nuestro primer caso de estudio. Interfaces, su rol en el chequeo de tipos y su relación con polimorfismo.
4	Semana del 30/03/2026	Completamos el caso de estudio. Los Tests Unitarios como mecanismo de aseguramiento de calidad y como guía al proceso de desarrollo. Automatización de test de unidad con herramientas de la familia xUnit. Pruebas de particiones equivalentes y valores de borde.
6	Semana del 06/04/2026	Colecciones heterogéneas de objetos; polimorfismo entre colecciones y polimorfismo entre los elementos en una colección. Iteradores de colecciones (y objetos que abstraen y/o soportan el comportamiento de iteración). Clausuras léxicas en objetos.
7	Semana del 13/04/2026	Identificar clases y responsabilidades a partir de una especificación. Modelo conceptual (clases candidatas). Pre y post contratos (que hay antes y que queda después) para los casos de uso. Derivar métodos a partir de los pre y post contratos. Editores UML: Editores gráficos vs. Editores UML.
8	Semana del 20/04/2026	Continuamos identificando clases y responsabilidades. Incorporamos herencia, sinónimos, y objetos no tan obvios. Discutimos Herencia vs. Composición.
9	Semana del 27/04/2026	Taller de diseño (aplicamos todo lo visto a una nueva especificación). Identificamos clases, determinamos responsabilidades, y discutimos diseños alternativos. Primera iteración de elaboración. Obtenemos nuestro primer modelo de clases candidatas y programamos los casos de uso más simples.
10	Semana del 4/05/2026	Taller de diseño. Segunda iteración de elaboración. Extendemos nuestro modelo de clases candidatas para cubrir casos de uso más complejos.
11	Semana del 11/05/2026	Particularidades de otros lenguajes de programación OO. Lenguajes basados en clases y basados en instancias. El rol del tipado Tipado estático y dinámico.
12	Semana del 18/05/2026	Arquitectura de la aplicación / servicio. Separación en capas. La importancia del modelo de dominio.
13	Semana del 25/05/2026	Consultas.
14	Semana del 01/06/2026	Consultas.
15	Semana del 08/06/2026	Parcial
16	Semana del 15/06/2026	Consultas.
17	Semana del 22/06/2026	Recuperatorio 1

18	Semana del 29/06/2026	Consultas.
19	Semana del 06/07/2026	Consultas.
20	Semana del 13/07/2026	Recuperatorio 2

Evaluaciones previstas	**Fecha**
Parcial anticipado – Fecha 0	Semana del 23/03/2026
Parcial normal – Fecha 1	Semana del 8/06/2026
Recuperatorio 3	Semana del 22/06/2026
Recuperatorio 4	Semana del 13/07/2026

Contacto de la cátedra (mail, sitio WEB, plataforma virtual de gestión de cursos):

Mail: arodrig@lifia.info.unlp.edu.ar
Plataforma virtual de gestión del curso: https://catedras.linti.unlp.edu.ar/
Sitio Web
Otros

Firma del profesor

Andrés Rodríguez