000027

Programación con Objetos II

INSTITUTOS/S: **INSTITUTO DE TECNOLOGÍA E INGENIERÍA**

CARRERA/S: **Licenciatura en Informática**

RESPONSABLE DE LA ASIGNATURA Y EQUIPO DOCENTE:
Gerardo Gonzalez Tulian

Darío Moreira
Marcelo Abal

AÑO: 2025

CRÉDITOS: 7

CARGA HORARIA DE INTERACCIÓN PEDAGÓGICA: 96

CARGA HORARIA TOTAL: 175

CÓDIGO DE LA MATERIA EN SIU: 765

000027

UNAHUR
UNIVERSIDAD NACIONAL DE HURLINGHAM

1. Fundamentación

La programación orientada a objetos es esencial en la formación y desarrollo de todo estudiante universitario en carreras relacionadas con la tecnología y la informática. En un mundo impulsado por el avance tecnológico, es fundamental tener una comprensión profunda de los principios y conceptos que sustentan la programación orientada a objetos. Por lo tanto, es esencial proporcionar a los estudiantes los conceptos avanzados y habilidades prácticas necesarias para abordar problemas complejos en su ámbito profesional.

La materia de Programación con Objetos 2 se presenta como un pilar esencial en el recorrido académico, profundizando en los conocimientos adquiridos en Programación con Objetos 1. Esta materia no solo refuerza los fundamentos de la programación orientada a objetos, sino que también introduce conceptos avanzados que son vitales para el desarrollo de software moderno y eficiente.

Durante la cursada de Programación con Objetos 2 se proporcionarán herramientas y técnicas avanzadas que serán aplicables y esenciales para la comprensión de materias más especializadas como: Construcción de Interfaces de Usuario, Estrategias de persistencia, Elementos de Ingeniería de Software, Programación con Objetos 3, entre otras. A través de este curso, los estudiantes desarrollarán habilidades que les permitirán diseñar y construir sistemas de software robustos y escalables.

2. Propósitos y/u objetivos

- *Desarrollar competencias avanzadas: brindar a los estudiantes herramientas y conocimientos avanzados en programación orientada a objetos, permitiéndoles enfrentar desafíos complejos en el desarrollo de software moderno.*

- *Fortalecer la capacidad de diseño: enfatizar la importancia del diseño de software como un proceso de toma de decisiones críticas, desarrollando la habilidad para crear soluciones de software robustas y eficientes.*

- *Integrar conceptos teóricos y prácticos: lograr una integración efectiva de los conceptos teóricos con su aplicación práctica, utilizando un entorno integrado de desarrollo (IDE) y fomentando el aprendizaje basado en proyectos.*

- *Promover buenas prácticas de desarrollo: introducir y promover el uso de patrones de diseño, notación UML, testeo unitario y manejo adecuado de errores para mejorar la calidad y mantenibilidad del software.*

3. Contenidos mínimos

Aproximación al diseño de software, entendiendo el diseño como un proceso de toma de decisiones críticas. Noción de decisión de diseño y su impacto en el desarrollo de software. Conceptos de acoplamiento y cohesión, y los problemas que surgen de un grado de acoplamiento inadecuado. Vinculación entre las ideas básicas de diseño y el paradigma de objetos. Características deseadas en un diseño de objetos y cómo lograrlas. Introducción y aplicación de patrones de diseño. Nociones sobre el proceso de diseño, incluyendo el manejo de eventos y metaprogramación. Uso de un entorno integrado de desarrollo de software (IDE). Notación UML para diagramas de clases, objetos y secuencias. Implementación de testeo unitario y automático. Manejo de errores y su impacto en el diseño de software.

4. Carga horaria

Créditos	*Interacción pedagógica*		*Trabajo Autónomo*	*Total*
7	*Teoría: 32*	*Práctica: 64*	*79*	*175*

4.1 Trabajo autónomo del/la estudiante

Actividad	*Carga horaria*
Lectura de contenidos teóricos apuntes o bibliografía obligatoria.	*9*
Desarrollo de los ejercicios prácticos propuestos de codificación.	*15*
Preparación de evaluaciones.	*20*
Construcción de diagramas sobre enunciados. Y uso de herramientas.	*15*
Aprendizaje autodirigido.	*20*

5. Programa analítico

UNIDAD 1: *Introducción al diseño de software.*

000027

Concepto de Diseño de Software. Definición y relevancia. Diseño como proceso de toma de decisiones. Principios de Diseño. Acoplamiento y cohesión. Problemas derivados del acoplamiento inadecuado. Herramientas y Entornos de Desarrollo. Uso de entornos integrados de desarrollo (IDE).

UNIDAD 2: *Patrones de diseño.*

Introducción a los Patrones de Diseño. Definición y clasificación de patrones. Importancia y beneficios. Patrones Creacionales. Singleton, Factory y Abstract Factory. Patrones Estructurales. Adapter, Composite y Decorator. Patrones de Comportamiento. Observer, Strategy y Command.

UNIDAD 3: *Modelado con UML.*

Introducción a UML. Historia y propósito de UML. Beneficios del modelado UML. Diagramas de Clases. Componentes y notaciones. Ejemplos y casos prácticos. Diagramas de Objetos. Componentes y notaciones. Ejemplos y casos prácticos. Diagramas de Secuencia. Componentes y notaciones. Ejemplos y casos prácticos.

UNIDAD 4: *Testeo y Calidad del Software.*

Principios de Testeo de Software. Importancia del testeo. Tipos de pruebas: unitarias, integrales (integración) y de sistema (e2e). Testeo Unitario. Introducción a JUnit. Escribir y ejecutar pruebas unitarias. Testeo Automático. Herramientas y frameworks para pruebas automáticas. Diseño guiado por pruebas.

UNIDAD 5: *Manejo de Errores y Excepciones.*

Conceptos Básicos de Manejo de Errores. Tipos de errores: sintácticos, lógicos y de tiempo de ejecución. Excepciones en Programación Orientada a Objetos. Manejo de excepciones. Código protegido y tratamiento diferenciado de excepciones. Impacto del Manejo de Errores en el Diseño. Estrategias para minimizar el impacto negativo en el diseño. Buenas prácticas.

UNIDAD 6*: Eventos y Metaprogramación.*
Contenido de la unidad: Introducción al Manejo de Eventos. Definición y tipos de eventos. Implementación de eventos en lenguajes de programación orientado a objetos. Metaprogramación. Conceptos y aplicaciones. Reflexión en lenguajes de programación orientado a objetos. Aplicaciones Prácticas. Ejemplos y casos de estudio.

6. Ejes y enunciados Multidimensionales y transversales

Eje	*Nivel de logro de Aprendizaje*	*Acciones de enseñanza*
C1. Identificación, formulación y resolución de problemas de informática	*Medio*	***Aprendizaje basado en problemas (ABP)**: Presentar casos reales donde los estudiantes deban analizar, formular y proponer soluciones a problemas informáticos.* ***Talleres de depuración y optimización:** Analizar código defectuoso o ineficiente para mejorarlo.*
C2. Concepción, diseño y desarrollo de proyectos de informática	*Medio*	***Metodologías ágiles:** Aplicar SCRUM o Kanban en el desarrollo de proyectos.* ***Proyectos iterativos:** Dividir el diseño y desarrollo en sprints con entregas parciales.* ***Casos de estudio de software exitoso:** Analizar cómo se concibieron proyectos reales.*
C3. Gestión, planificación, ejecución y control de proyectos de informática	*No aporta*	*No aplica.*
C4. Utilización de técnicas y herramientas de aplicación en la informática	*Medio*	***Laboratorios de herramientas:** Uso de entornos de desarrollo como GitHub, Docker, JIRA, Jenkins.* ***Integración de herramientas en proyectos reales:** Aplicar técnicas y herramientas en situaciones concretas.* ***Capacitaciones en frameworks y lenguajes emergentes:** Talleres sobre las herramientas más demandadas en la industria.* ***Simulación de entornos de desarrollo profesional:** Trabajo en proyectos con entornos similares a los del mercado laboral.*
C5. Generación de desarrollos tecnológicos y/o innovaciones tecnológicas	*No aporta*	*No aplica.*
C6. Fundamentos para el desempeño en equipos de trabajo	*Bajo*	***Tareas colaborativas:** Uso de Git en grupo con code reviews entre pares.* ***Roles en proyectos:** Asignar roles (Project Manager, DevOps, QA, etc.) en proyectos grupales.*

		Dinámicas de team building: *Actividades para mejorar la comunicación y cohesión en equipos. Juegos de simulación: Simular escenarios donde se requiera coordinación y toma de decisiones en grupo.*
C7. Fundamentos para la comunicación efectiva	*Medio*	***Presentaciones técnicas:*** *Exposición de proyectos con feedback.* ***Escritura técnica****: Redacción de documentación de software, reportes y propuestas.* ***Debates sobre tendencias tecnológicas:*** *Fomentar la argumentación fundamentada.* ***Uso de herramientas de comunicación profesional****: Slack, Notion, Trello, Google Docs, etc.*
C8. Fundamentos para la acción ética y responsable	*No aporta*	*No aplica.*
C9. Fundamentos para evaluar y actuar en relación con el impacto social de su actividad en el contexto global y local	*No aporta*	*No aplica.*
C10.Fundamentos para el aprendizaje continuo	*Medio*	***Aprendizaje autodirigido:*** *Promoción de certificaciones online (Coursera, Udemy, edX).* ***Foros de discusión técnica:*** *Participación en Stack Overflow, Reddit, y grupos de GitHub.* ***Reflexión sobre la evolución tecnológica:*** *Discusión de tendencias, nuevas herramientas y cambios en la industria.* ***Aprendizaje basado en proyectos:*** *Incentivar la autoexploración y desarrollo de proyectos propios.*
C11.Fundamentos para la acción emprendedora	*No aporta*	*No aplica.*

7. Bibliografía y recursos

7.1 Bibliografía obligatoria

Juárez, A. (2019). "Java + Programación Orientada a Objetos". EUDEBA. Facultad de Ingeniería de la Universidad de Buenos Aires.

Cairó Battistutti Osvaldo (2022). "Aprende a programar en Java: de cero al infinito". 1ra Edición México D.F.: Alfaomega.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2018). " Patrones de diseño: Elementos reutilizables en la programación orientada a objetos" (edición en español). Addison-Wesley.

Dreamtech Tech- Serie (2020) " Core and advanced JAva".Serie: Black Book. Delhi.

Martin, R. C. (2015). "Clean code: Un manual de artesanía del software ágil" (edición en español). Editorial Pearson.

7.2 Bibliografía optativa

Larman, C. (2013). "UML y patrones: Introducción al análisis y diseño orientado a objetos y al proceso unificado" (3ª ed. en español). Prentice Hall.

Deitel, P., & Deitel, H. (2015). "Java: Cómo programar" (10ª ed. en español). Pearson Educación.

Richards, M., & Ford, N. (2020)." Arquitectura de software: Patrones para sistemas complejos". O'Reilly Media.

Shore, J., & Warden, S. (2021)." Principios de desarrollo ágil: Cómo entregar valor y calidad". O'Reilly Media.

Fowler, M. (2019). "Refactoring: Mejora del diseño de código existente" (2ª ed. en español). Addison-Wesley.

7.3 Recursos

Uso de IDE (Integrated Development Environment) como entorno de desarrollo integrado que proporciona herramientas para facilitar la programación.

- *Eclipse.*
- *IntelliJ.*
- *VS Code (Visual Studio Code).*

8. Metodología de enseñanza

8.1 Modalidades u opciones pedagógicas

Las clases se presentan en modalidad mixta, presencial y virtual utilizando las herramientas provistas por la universidad (Campus, reuniones virtuales usando Google Meet y Laboratorios).

Se presenta cada unidad temática introduciendo los conceptos fundamentales realizando analogías con ejemplos reales, que permite relacionar los contenidos de la materia con las herramientas habituales de trabajo.

Los contenidos de la materia se presentan de forma iterativa e incremental, esto le permite al alumno construir sus propios procedimientos para resolver una situación problemática, lo que implica que sus ideas puedan verse modificadas y de esta forma siga construyendo nuevos conocimientos.

Se motiva a los estudiantes en el uso del Campus, para la resolución de dudas tanto de conceptos teóricos como prácticos. Además, la cátedra habilita a los alumnos a realizar grabaciones y generen sus propios repositorios con soporte audiovisual de los contenidos, que pueden consultar luego de haber asistido a la clase.

8.2 Formación Práctica

Se desarrollarán diferentes prácticas individuales y/o grupales aplicando los contenidos dados en las diferentes unidades temáticas, para poder fijar los conocimientos de forma práctica. Se fomentará al alumno al trabajo en grupo.

9. Evaluación y régimen de aprobación

9.1 Modalidad de evaluación

El sistema normal de evaluación consistirá en 2 Exámenes Parciales con un recuperatorio por cada uno de ellos, según el cronograma previsto en el campus de la materia. Tanto en los parciales como en los recuperatorios, los contenidos a evaluar representan la totalidad de la materia descripta en el programa. Los mismos se realizarán en las fechas, que, a tal efecto, se establezcan en el cronograma correspondiente.

9.2 Aprobación de la cursada

Para aprobar la cursada y obtener la condición de regular, el régimen académico establece que debe obtenerse una nota no inferior a cuatro (4) puntos. Todas las instancias evaluativas deberán tener una instancia de recuperatorio. Podrán acceder a la administración de esta modalidad solo aquellos y aquellas estudiantes que hayan obtenido una nota inferior o igual a 6 (seis) puntos en el examen parcial.
Siempre que se realice una evaluación de carácter recuperatorio, la calificación que los/as estudiantes obtengan reemplazará la calificación obtenida en el examen que se ha recuperado y será la considerada definitiva a los efectos de la aprobación.

9.3 Acreditación de la materia

La materia puede aprobarse por promoción, evaluación integradora, examen final o libre.

Promoción directa: *tal como lo establece el art°17 del Régimen Académico, para acceder a esta modalidad, el/la estudiante deberá aprobar la cursada de la materia con una nota no inferior a siete (7) puntos, no obteniendo en ninguna de las instancias de evaluación parcial menos de seis (6) puntos, sean evaluaciones parciales o recuperatorios. El promedio estricto resultante deberá ser una nota igual o superior a siete (7) sin mediar ningún redondeo.*

Evaluación integradora: *tal como lo establece el art°18 del Régimen Académico, podrán acceder a esta evaluación aquellos estudiantes que hayan aprobado lo cursado con una nota de entre cuatro (4) y seis (6) puntos.*
La evaluación integradora tendrá lugar por única vez en el primer llamado a exámenes finales posterior al término de la cursada. Deberá tener lugar en el mismo día y horario de la cursada y será administrada, preferentemente, por el/la docente a cargo de la comisión. Se aprobará tal instancia con una nota igual o superior a cuatro (4) puntos, significando la aprobación de la materia.
La nota obtenida se promediará con la nota de la cursada.

Examen final*: Instancia destinada a quienes opten por no rendir la evaluación integradora o hayan regularizado la materia en cuatrimestres anteriores. Se evalúa la totalidad de los contenidos del programa de la materia y se aprueba con una calificación igual o superior a cuatro (4) puntos. Esta nota no se promedia con la cursada.*