4125 - LENGUAJE DE PROGRAMACIÓN I

I - Datos de identificación de la asignatura

Carrera:	Licenciatura en Análisis de Sistemas		
Código:	4125	**Plan:**	2024
Denominación:	Lenguaje de programación I		
Área:	Tecnologías aplicadas		
Año:	Segundo		
Horas con acompañamiento docente (HTD), semanal			4
Horas de Trabajo Independiente del estudiante (HTI), semanal			6
Horas semanales (HS)			10
Cantidad de sesiones			32
Total Horas de Trabajo con el docente (THTD)			128
THD teóricas	32	**THD prácticas**	96
Total de Horas de Trabajo Independiente del estudiante (THTI)			192
Total Horas Académicas (THA)			320
Crédito académico (CA)			12,8
Pre-requisito:	Fundamentos de algorítmica y programación		

II - Fundamentación

La asignatura tiene como objetivo principal proporcionar a los estudiantes los conocimientos fundamentales sobre los paradigmas de programación y, en particular, la orientación a objetos.

En la actualidad, el desarrollo de software va más allá de simples construcciones de programación y se enfoca en la utilización de componentes modulares, siguiendo las mejores prácticas para construir aplicaciones extensibles y manejables. Los paradigmas de programación, como la orientación a objetos, ofrecen una perspectiva sobre cómo organizar datos y rutinas en bibliotecas de código reutilizable, centrándose en la estructuración de datos y rutinas en clases (para especificación) y objetos (para instanciación).

La orientación a objetos es fundamental en el desarrollo de software y sistemas, ya que proporciona un conjunto de disposiciones de comportamiento que especifican cómo interactúan las entidades y jerarquías de clases y objetos. Estos conceptos forman la base de los patrones arquitectónicos ampliamente aceptados en el desarrollo de software. Además, la orientación a objetos también es utilizada para el modelado de dominios de problemas, trascendiendo el ámbito de la programación.

La asignatura aborda tanto los fundamentos teóricos como la aplicación práctica de la orientación a objetos. Los estudiantes aprenderán los conceptos esenciales de la programación orientada a objetos y su implementación en un lenguaje de programación específico. Se explorarán los principios de encapsulamiento, herencia, polimorfismo y otros aspectos clave de la orientación a objetos.

El enfoque de la asignatura se centra en las aplicaciones prácticas que abarcan desde el diseño hasta la implementación de sistemas de software. Aunque el análisis y diseño de sistemas generalmente se aborda en cursos separados, se proporcionarán elementos

relevantes en esta asignatura para comprender cómo la orientación a objetos se aplica en el contexto del desarrollo de software.

La asignatura busca enseñar a los estudiantes los fundamentos teóricos y prácticos de la programación orientada a objetos. Al comprender y aplicar estos conceptos, los estudiantes podrán desarrollar software de calidad, utilizando las mejores prácticas de diseño y construcción de componentes modulares. La naturaleza de la asignatura es teórica-práctica, combinando la adquisición de conocimientos teóricos con la aplicación práctica en la resolución de problemas y el desarrollo de programas.

III - Competencias a desarrollar

Competencias genéricas

1. Actuar con autonomía.
2. Demostrar capacidad de abstracción, análisis y síntesis.
3. Identificar, analizar, abstraer, formular y resolver problemas relacionados con sus áreas de competencia.
4. Conocer y saber aplicar técnicas y herramientas actualizadas en sus áreas de competencia.
5. Diseñar, programar, ejecutar, analizar e interpretar resultados de pruebas realizadas en sus áreas de competencia.

Competencias específicas

1. Comprender y aplicar con claridad los conceptos fundamentales de la programación orientada a objetos (clase, objeto, atributo, método), para crear estructuras lógicas que representen entidades del mundo real en un lenguaje de programación.
2. Utilizar de forma adecuada las modalidades de instanciación de objetos (constructores, clonación, instanciación directa), para crear instancias según el contexto del problema y el diseño del sistema.
3. Implementar con precisión la comunicación entre objetos mediante el envío y recepción de mensajes, para lograr interacción efectiva y modular entre componentes del programa.
4. Comprender y aplicar el principio de encapsulación, utilizando modificadores de acceso y buenas prácticas de ocultamiento de información, para proteger los datos y asegurar la integridad del sistema.
5. Aplicar correctamente el concepto de herencia, para construir jerarquías de clases y reutilizar código, gestionando las dependencias entre clases y promoviendo un diseño modular y flexible mediante técnicas como la inyección de dependencias.
6. Identificar y aplicar con claridad mecanismos de abstracción (clases abstractas, interfaces, métodos abstractos), para representar de forma eficiente los elementos del dominio del problema y reducir la complejidad del sistema.
7. Utilizar eficazmente el polimorfismo, mediante enlace dinámico o tardío, para crear código genérico y flexible capaz de operar con diferentes tipos de objetos.
8. Aplicar con criterio técnico patrones de diseño orientados a objetos (como fábrica, observador, estrategia), para resolver problemas comunes de estructura, flexibilidad y mantenimiento en el diseño de software.
9. Modelar con precisión entidades del dominio del problema mediante clases y objetos, utilizando técnicas de análisis y diseño orientado a objetos, para representar adecuadamente su estructura y comportamiento.

10. Organizar y estructurar de manera modular programas utilizando conceptos de la programación orientada a objetos, diseñando clases y métodos que reflejen las interacciones, responsabilidades y comportamientos del sistema.

IV - Cuerpo de conocimientos

Unidad 1: Aplicar elementos fundamentales de objetos y clases.
Contenidos:

- Modelado de las propiedades de una entidad usando tipos de datos y variables dentro de una definición de clase
- Modelado de los comportamientos de una entidad usando métodos dentro de una definición de clase
- Distinción entre instancia y miembros de clase

Unidad 2: Modalidades de creación de instancias.
Contenidos:

- Operaciones de instanciación de objetos
- Base de clase para instanciación de objetos
- Instanciación de objetos basados en prototipos
- Creación de instancias de objetos mediante constructores y operadores relacionados
- Implementación de funciones de lenguaje relacionadas con la gestión de la memoria (no gestionada frente a gestionada)

Unidad 3: Comunicación interna e intercomunicación de entidades.
Contenidos:

- Declaración y uso de métodos.
- Llamadas por referencia y valor.
- Visibilidad de miembros y modificación de acceso específico del lenguaje.
- Empaquetado de clases para reutilización (espacios de nombres, paquetes, bibliotecas).
- Acceso a clases empaquetadas.

Unidad 4: Encapsulamiento.
Contenidos:

- Funciones de lenguaje para estratificar y organizar la visibilidad y accesibilidad de los miembros dentro de una clase.
- Autonomía e integridad de los objetos utilizando técnicas de encapsulación.
- Refactorización de código aplicando encapsulación.
- Desacoplamiento mediante encapsulación.

Unidad 5: Gestión de herencias y dependencias.
Contenidos:

- Extensibilidad y reutilización de objetos a través de la herencia.
- Herencia única y múltiple.
- Extensibilidad y reutilización de objetos a través de la composición.
- Dependencias y constructores.
- Inyección de dependencia e inversión de control.
- Interfaces y mixins como alternativa a la herencia múltiple.
- Extensibilidad y reutilización de objetos mediante delegación.

Unidad 6: Abstracción.
Contenidos:
- Características del lenguaje que identifican una clase como abstracta.
- Refactorización o diseño de jerarquías de herencia abstractas.
- Abstracción frente a interfaces (miembros virtuales).
- Estructuración de datos de tipos de datos relacionados a través de referencias abstractas.

Unidad 7: Polimorfismo.
Contenidos:
- Relación entre referencias de clase base/principal e instancias de clase derivada/ secundaria.
- Sobrescribir los comportamientos y atributos de la clase base/principal en las clases derivadas/secundarias.
- Comportamientos que responden de manera única a las llamadas a métodos comúnmente heredados.
- Separación de responsabilidades.

Unidad 8: Patrones de diseño.
Contenidos:
- Patrones creacionales, estructurales y de comportamiento
- Relación con las notaciones de modelado
- Principios SOLID
- Patrón de arquitectura de software.
- Implementación de patrones de arquitectura de software.

Unidad 9: Objetos y clases para el modelado de entidades.
Contenidos:
- Relación entre la implementación de la programación orientada a objetos y UML
- Limitaciones en la implementación del lenguaje
- Mapeo objeto-relacional y persistencia de datos utilizando lenguaje estructurado de consultas
- Reutilización y Bibliotecas

Unidad 10: Conceptos de Orientación a Objetos en la organización y estructuración de programas para la gestión de comportamientos y conceptos.
Contenidos:
- Diseño de objetos y clases a partir de descripciones de dominio
- Identificación de oportunidades de abstracción y especialización.
- Diseño para la gestión de dependencias mediante herencia y composición.

V - Estrategias didácticas a ser implementadas en el proceso de enseñanza aprendizaje. (abarcando actividades de formación e investigación)

La materia se desarrollará por medio de clases expositivas que expliquen los conceptos de la programación orientada a objetos y cómo aplicarlos en un lenguaje de programación.

En clases en laboratorio se aplicarán los conceptos estudiados en la solución de problemas planteados previo diseño de la solución en diagramas de clases. Además, se asignarán trabajos prácticos a los estudiantes. En los trabajos prácticos se organizan actividades por equipos de trabajo, con 2 a 4 alumnos en las actividades prácticas. En principio los alumnos son “pares” sin roles determinados en el equipo, aunque dado un problema a resolver, ellos pueden definir sus roles.

Los equipos deben demostrar capacidad de aprender (a partir de problemas planteados en la práctica y ejemplos desarrollados en la teoría), teniendo la posibilidad de consultar a sus docentes. Cada comisión/equipo debe documentar la solución de los ejercicios que se plantean y son examinados en forma individual en las evaluaciones prácticas (por escrito) y pueden tener que defender sus soluciones en un coloquio de teoría.

La cátedra mantiene planillas que permiten calificar diferentes aptitudes de los miembros del equipo (conocimientos / modo de expresarse / predisposición al trabajo colaborativo). Estas planillas son reunidas por el docente para ser tenidas en cuenta en las evaluaciones parciales y finales de los alumnos.

En el seguimiento y evaluación de los alumnos se trata de formarlos en una metodología de ir del "caso problema del mundo real" a su solución efectiva con herramientas informáticas, ejemplificadas en programas informáticos y/o lenguajes alternativos. Para ello se pone énfasis en el modo de abstraer el problema y diseñar una solución verificable.

El alumno es evaluado en todos los aspectos relacionados con las competencias constando el resultado de esta evaluación en la corrección de las pruebas (parciales y finales) del alumno. Se pone énfasis en detallar los aspectos técnicos que debe perfeccionar relacionados con los conceptos de Algoritmos, Datos y Programas.

VI - Estrategias de evaluación.

La evaluación será formativa y procesual, se realizará a través de pruebas (exámenes) que podrán ser escritas, orales o de ejecución que a su vez podrá ser mediante trabajos individuales o grupales. La materia consta de dos pruebas parciales, con un recuperatorio y tres oportunidades para la prueba final.

En estos parciales, así como en el examen final, se evaluarán las competencias alcanzadas a través de actividades de contenido teórico y práctico que permitan dar cuenta del avance conceptual en los temas que se han desarrollado, se incorporan preguntas específicas tipo sobre "dónde cree Ud. que es aplicable este conocimiento/método" y se refleja en la corrección de las pruebas del alumno.

En algunos temas se trabaja también con ejercitaciones de aplicación en clase, que requieren de un ejercicio de integración de conceptos y que complementan la evaluación a través de los parciales.

Para la obtención de calificaciones parciales y finales se tendrá en cuenta el Reglamento Académico de la universidad.

VII - Actividades de extensión y de responsabilidad social universitaria.

Rige de acuerdo al reglamento de la Universidad y el reglamento interno de la facultad.

VIII - Fuentes bibliográficas

Básica

- López Román, L. (2006). Metodología de la programación orientada a objetos (i9789701511732).
- Regino, E. O. (2015). Lógica de programación orientada a objetos. Ecoe Ediciones.
- Wilson, R. R., & Mario Silva, M. (2016). Introducción a Java: guía de actividades prácticas. Universidad del Bosque.

- Pérez, J. C. M. (2015). Programación orientada a objetos. Grupo Editorial RA-MA.
- De Parga, C. J. (2015). UML. Aplicaciones en Java y C++. Grupo Editorial RA-MA.
- Debrauwer, L., & Van der Heyde, F. (2016). UML 2.5: iniciación, ejemplos y ejercicios corregidos. Ediciones ENI.

Complementaria

- Eckstein, R. (2017). Java SE Application Design With MVC. Recuperado de https://www.oracle.com/technical-resources/articles/javase/mvc.html
- Freeman, E., Freeman, E. (2004) Head First Design Patterns. O'Reilly Media.
- McLaughlin, B. D, Pollice, G., West, D. (2007). Head First Object-Oriented Analysis and Design