Carrera INGENIERÍA EN INFORMÁTICA		
Asignatura 3646 - Paradigmas de Programación		
Trayecto de Programación		
Año académico 2023		
Responsable / Jefe de cátedra Dra. Verónica Inés Aubin		
Carga horaria semanal 4hs	**Carga horaria total** 64hs	**Créditos** ----
Modalidad: Presencial		
Correlativas anteriores: ALGORITMOS Y ESTRUCTURAS DE DATOS - ANALISIS MATEMATICO II	**Correlativas posteriores:** PROGRAMACION AVANZADA - AUTOMATAS Y GRAMATICAS - PROGRAMACION CONCURRENTE - INTELIGENCIA ARTIFICIAL	
Conocimientos necesarios -------		

Descripción de la asignatura

En esta asignatura se estudian los paradigmas de programación más representativos de las diferentes formas de modelar un programa informático. El objetivo principal de esta asignatura es proporcionar una visión general de los distintos paradigmas (imperativo, funcional, lógico) de forma que el estudiante sea capaz de elegir el modelo adecuado para resolver cada problema. Se profundiza en la programación orientada a objetos. Prepara al estudiante en el Desarrollo Dirigido por las Pruebas. Se introduce el concepto de patrones de diseño.

Metodología de enseñanza

Se utilizan metodologías activas de enseñanza, especialmente aprendizaje basado en problemas (ABP). Se presenta cada núcleo temático, introduciendo las ideas fundamentales con la asistencia de ejemplos reales, lo que motiva a los alumnos y permite relacionar los contenidos de la materia con las herramientas de desarrollo que están acostumbrados a utilizar. Luego, se formalizan las definiciones correspondientes para complementar y organizar los contenidos en un marco teórico establecido.

Los contenidos de la asignatura se presentan de forma iterativa e incremental, de modo que los estudiantes puedan integrar los nuevos conceptos a los anteriores de forma de lograr aprendizajes significativos.

Se motiva a los estudiantes en el uso de foros internos, además de las plataformas MIeL y Teams, para la resolución de dudas tanto de conceptos teóricos como prácticos, permitiendo desarrollar las capacidades de comunicación y afianzar el uso del lenguaje técnico.

La materia tiene una fuerte carga práctica. En la misma se resuelven problemas, desarrollando pequeños sistemas, aplicando lo visto en las clases, trabajando de forma colaborativa, simulando un entorno de trabajo real.

A lo largo de los encuentros se irá desarrollando un TP especial con puntos de control intermedios.

La cátedra cuenta con soporte digital de los contenidos, que los alumnos pueden consultar luego de haber asistido a la clase.

Objetivos de aprendizaje

A través de esta asignatura, el alumno habrá adquirido los conocimientos necesarios y suficientes para estar en condiciones de:

Objetivos Generales:

- Analizar, plantear y resolver situaciones problemáticas.
- Organizar y planificar su trabajo.
- Hacer transferencia de los conocimientos teóricos a la práctica.
- Adquirir la capacidad de trabajar en equipo.
- Identificar un problema a partir de una situación problemática presentada.
- Diseñar un algoritmo eficiente para la resolución de una situación problemática analizada.
- Desarrollar un algoritmo utilizando un lenguaje de programación.
- Plantear casos de prueba de forma tal de ver los casos generales y particulares de cada situación problemática planteada.
- Expresar los contenidos teóricos de la materia y su vinculación con situaciones de la vida real.
- Detectar la fuerte vinculación de esta asignatura con materias de años anteriores y posteriores del plan de carrera.
- Integrar grupos de trabajo, potenciando su propio aprendizaje a través de la interacción y cooperación con sus pares.
- Utilizar con fluidez el lenguaje técnico relacionado con la materia.

Objetivos Específicos:

1. Conocer las características de los distintos paradigmas de programación.
2. Identificar el paradigma más apropiado para la resolución de un problema en particular
3. Proporcionar una comprensión sólida de los conceptos fundamentales del modelo de objetos.
4. Identificar en un enunciado textual de un problema las abstracciones de datos susceptibles de dar lugar a clases de datos que permitan resolver el problema.
5. Interpretar una representación de sistema mediante UML.
6. Diseñar e implementar una clase en un lenguaje de programación aplicando buenas prácticas de Ingeniería de Software
7. Comprender e implementar correctamente los conceptos de herencia, polimorfismo, interfaces, manejo de errores y excepciones
8. Aplicar Patrones de diseño en la resolución de problemas.
9. Aplicar la técnica de Desarrollo Dirigido por Tests (Test-Driven Development o TDD) para el desarrollo del software.
10. Reconocer e incluir a la Verificación y Validación de Software como parte fundamental en la actividad del programador.

11. Adquirir conocimientos básicos de la programación funcional y lógica.
12. Estimular la integración del estudiante en grupos de trabajo, de forma de potenciar su propio aprendizaje a través de la interacción y cooperación con sus pares.

Contenidos mínimos

Paradigmas de programación. Programación Orientada a Objetos. Conceptos. Objetos y clases. Encapsulamiento y Abstracción. Herencia y Polimorfismo. Interfaces. Patrones. Manejo de Errores. Testeo. Programación Lógica. Lógica de predicados, argumentos, interpretación, reglas, preguntas, variables, ámbito de una variable, corte. Programación Funcional. Tipos y clases. Funciones y operadores.

Competencias a desarrollar

Genéricas

- Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
- Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
- Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
- Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.
- Desempeño en equipos de trabajo.
- Comunicación efectiva.
- Actuación profesional ética y responsable.
- Aprendizaje continuo.
- Desarrollo de una actitud profesional emprendedora.

Específicas

- Especificación, proyecto y desarrollo de software.
- Establecimiento de métricas y normas de calidad de software.
- Especificación, proyecto y desarrollo de sistemas de información.
- Especificación, proyecto y desarrollo de sistemas de comunicación de datos.

Programa analítico	
Unidad 1	**Paradigmas de Programación** Concepto y Tipos de Paradigmas. Introducción y objetivos. Concepto de Paradigma de Programación. Tipos de Paradigmas de Programación. Lenguajes de programación. Familias de lenguajes y evolución histórica.
Unidad 2	**Paradigma Imperativo** Estilos de programación en la programación Imperativa: Estructurada por procedimientos, modular, con objetos y orientada a objetos. Apoyo de un lenguaje para cada estilo de programación. Testeo. Pruebas y Desarrollo Dirigido por las Pruebas.
Unidad 3	**Programación Orientada a Objetos** Introducción a la Orientación a Objetos. Motivación.

	El modelo de objetos. Objeto. Programa. Abstracción. Encapsulamiento Clase. Relaciones entre clases. Jerarquía. Herencia. Delegación. Composición y Agregación. Polimorfismo. Manejo de errores. Errores y excepciones. Ciclo de desarrollo en objetos. Objetivos. Ventajas. Problemas
Unidad 4	**Herramientas de la Programación Orientada a Objetos** Alcance y acceso a atributos. Clases abstractas. Interfaces. Métodos virtuales. Diagramas de Clases utilizando UML. Atributos de clase y de objeto. Constructores. Colecciones. Patrones de diseño.
Unidad 5	**Programación Lógica** Lógica de predicados argumentos, interpretación, reglas, preguntas, variables, ámbito de una variable, corte. Estructura de un programa en Prolog. Listas y secuencias finitas. Equivalencia entre el Prolog y los operadores del TDA conjunto. Equivalencia entre Prolog y consultas SQL
Unidad 6	**Programación Funcional** Introducción a la programación funcional. Haskell. Tipos y clases. Funciones y operadores. Manejo de Listas. Pattern matching. Currying.

Planificación de actividades					
Semana	**Clase**	**Actividad**	**Tipo**	**Duración estimada**	**Unidad/es**
Semana 1	1	Presentación de los docentes del curso, breve explicación de las pautas generales de la materia. Paradigmas de programación. Instalación de herramientas. Entorno de desarrollo Eclipse. Introducción al lenguaje de programación Java. Ejercicios simples de programación.	Teoría - Práctica en laboratorio	4 hs	Unidad 1 Unidad 2
Semana 2	2	Introducción al Paradigma Imperativo. Pruebas de Software. La preparación del lote de prueba. Documentación del Lote de Pruebas. El programa probador. Ventajas de preparar la prueba antes de comenzar la programación. Pruebas del software: de caja negra, de caja blanca, inspecciones. Ventajas de la inspección del código fuente.	Teoría - Práctica en laboratorio	4 hs	Unidad 2
Semana 3	3	Paradigma Imperativo Lectura y escritura de archivos. Ejercicios con procesamiento de archivos	Teoría - Práctica en laboratorio	4 hs	Unidad 3
Semana 4	4	Conceptos de Programación Orientada a Objetos.	Teoría - Práctica en laboratorio	4 hs	Unidad 3 / 4

		Diferencia entre aproximación estructurada básica y programación orientada a objetos. Encapsulamiento. Abstracción. Clase, atributos, métodos y mensajes. Constructores. Pruebas como construcción de software. Ejemplos de Clase Complejo, entre otras.			
Semana 5	5	Modificadores de alcance. Ámbito de una variable. Static. Final. Sobrecarga y sobreescritura de métodos. Equals Vs ==. Hashcode. Representación como cadenas. Comparabilidad. Ejercicios de clases simples.	Teoría - Práctica en laboratorio	4 hs	Unidad 3 / 4
Semana 6	6	Programación Orientada a Objetos. Composición. Agregación. Colecciones.	Teoría - Práctica en laboratorio	4 hs	Unidad 3 / 4
Semana 7	7	Conceptos de Programación Orientada a Objetos. Herencia. Polimorfismo. Clases Abstractas e Interfaces.	Teoría - Práctica en laboratorio	4 hs	Unidad 3 / 4
Semana 8	8	Programación Orientada a Objetos. Errores y Excepciones.	Teoría - Práctica en laboratorio	4 hs	Unidad 3 / 4

		Práctica de Programación Orientada a Objetos.			
Semana 9	9	Programación Orientada a Objetos. Patrones de Diseño.	Teoría - Práctica en laboratorio	4 hs	Unidad 4
Semana 10	10	Programación Lógica. Reglas. Consultas. Variables. Ámbito.	Teoría - Práctica en laboratorio	4 hs	Unidad 5
Semana 11	11	Programación Lógica. Producto Cartesiano. Máximos y mínimos. Operador de corte.	Teoría - Práctica en laboratorio	4 hs	Unidad 5
Semana 12	12	Programación Lógica. Práctica. Programación Funcional. Funciones. Tipos. Operadores. Entrega trabajo práctico	Teoría - Práctica en laboratorio	4 hs	Unidad 5
Semana 13	13	Parcial	Parcial en laboratorio	4 hs	Unidades 1, 2, 3, 4 y 5.
Semana 14	14	Programación Funcional. Pattern Matching. Currying.	Teoría - Práctica en laboratorio	4 hs	Unidad 6
Semana 15	15	Recuperatorio Parcial Reentrega trabajo práctico	Recuperatorio en laboratorio	4 hs	Unidades 1, 2, 3, 4 y 5.
Semana 16	16	Defensa trabajo práctico. Entrega de notas finales	Defensa TP en laboratorio	4 hs	Unidades 1, 2, 3, 4 y 5.

Evaluación
Descripción del proceso evaluativo desarrollado por la cátedra El proceso de evaluación consta de: ● Un examen parcial presencial ● Un examen recuperatorio presencial.

- Un trabajo práctico grupal.

Las evaluaciones (examen parcial y recuperatorio) serán teórico-prácticas.
Constan de preguntas para desarrollar y/o del tipo opción múltiple justificando la respuesta y un porcentaje no menor al 50% de ejercicios prácticos donde el alumno deberá resolver problemas desarrollando código con las pruebas correspondientes. En examen parcial se evalúan los contenidos de la Unidad 1 a la Unidad 6. Los parciales corregidos serán entregados en mano a los estudiantes, teniendo la posibilidad de realizar preguntas sobre las correcciones efectuadas. Previamente a la entrega de los parciales corregidos, se explicará brevemente la solución a los problemas planteados haciendo hincapié en los puntos donde se observaron los errores más comunes. Una vez completado este procedimiento las calificaciones son confirmadas y volcadas al sistema Guaraní.

El trabajo práctico será sobre los contenidos vistos en las Unidades 1, 2, 3 y 4.
El trabajo se realizará en forma grupal, con equipos de 4 a 6 personas. Deberá trabajarse utilizando git y GitHub. La resolución debe incluir: Diagrama de clases UML y pruebas unitarias. Deben generar un breve video mostrando en hasta 5 minutos, cómo funciona el programa entregado, La aprobación del trabajo práctico conlleva la aprobación del código entregado y una defensa oral individual del mismo.

Primera evaluación	Semana 13	Teórico-práctica	4 hs
Segunda evaluación	A partir de la semana 13	Trabajo práctico	4 hs
Recuperatorio	Semana 15	Teórico-práctica	4 hs

Bibliografía obligatoria

Título	**Autor**	**Editorial**	**Edición**	**Año**
Core Java Volumen I – Fundamentos	Horstmann Cornell	Pearson Prentice Hill	7ma edición	2006

Java 1.1. The complete reference	Naughton, Patrick; Schildt, Herbert	McGraw –Hill	2a.ed	1998
The practice of Prolog	Sterling, Leon S..	MIT Press		1990
Introducción a la programación funcional con Haskell	Richard Bird	Prentice-Hall		2000

Bibliografía complementaria recomendada				
Titulo	**Autor**	**Editorial**	**Edición**	**Año**
Seven Languages in Seven Weeks	Bruce A. Tate	The Pragmatic Programmers	1ra edición	2012
Learn You a Haskell for Great Good!: A Beginner's Guide	Miran Lipovaca	No Starch Press		2011
Design Patterns Elements of reusable object-oriented software	Gamma, Erich and Helm, Richard and Johnson, Ralph and Vlissides, John	Addison Wesley		2009
Head first design patterns	Eric Freeman, Kathy Sierra, Bert Bates, Elisabeth Robson	O'Reilly Media, Inc.		2008

Otros recursos obligatorios [Videos, enlaces, otros. Incluir una fila por cada recurso]	
Nombre	

Otros recursos complementarios [Videos, enlaces, otros. Incluir una fila por cada recurso]	
Nombre	