1. PRESENTACIÓN DEL ESPACIO CURRICULAR

Espacio curricular: Programación II					
Código SIU-guaraní: No dispone		**Horas Presenciales:** 75		**Ciclo lectivo:** 2024	
Carrera:	Licenciatura en Ciencias de la Computación		**Plan de Estudio:**	Ord. 04/23	
Dirección a la que pertenece	Licenciatura en Computación		**Bloque/ Trayecto**	Algoritmos y Lenguajes	
Ubicación curricular:	2do Sem	**Créditos** 8	**Formato Curricular**	Teoría/práctica	
Equipo docente					
Cargo: Adjunto	**Nombre:** Javier Rosenstein		**Correo:** javier.rosenstein@ingenieria.uncuyo.edu.ar		
Cargo: JTP	**Nombre:** Daniel Fontana		**Correo:** daniel.fontana@ingenieria.uncuyo.edu.ar		
Cargo: JTP	**Nombre:**		**Correo:**		

Fundamentación

La asignatura Programación II tiene como objetivo principal que los estudiantes aprendan todos los conceptos fundamentales la Programación Orientada a Objetos (POO). La POO es mucho más que una técnica de codificación; es una filosofía que fomenta el pensamiento modular, la reutilización de código y la creación de sistemas de software robustos y escalables. A lo largo del cursado de esta asignatura, se explorará cómo representar objetos del mundo real y sus interacciones, lo que permitirá diseñar y construir programas más comprensibles y mantenibles.

Durante el cursado el estudiante se irá sumergiendo en una variedad de conceptos clave de la POO:

1. Clases y Objetos: Aprenderás a definir clases para representar entidades y objetos del mundo real, así como a crear instancias de estas clases para interactuar con tus programas.
2. Encapsulación: Descubrirás cómo ocultar la implementación interna de tus clases y proporcionar una interfaz clara y segura para interactuar con ellas.
3. Herencia y Polimorfismo: Explorarás cómo crear jerarquías de clases y aprovechar el polimorfismo para escribir código más flexible y extensible.
4. Abstracción: Aprenderás a identificar y representar las características esenciales de objetos, simplificando así el proceso de diseño y desarrollo.
5. Diseño Orientado a Objetos: Te sumergirás en técnicas de diseño que te ayudarán a planificar y estructurar sistemas de software complejos de manera eficiente y eficaz.

A lo largo del curso, se promoverá la resolución de ejercicios prácticos y proyectos que permitirán a los estudiantes aplicar estos conceptos en contextos del mundo real. Además, se fomentará el desarrollo del pensamiento crítico y las habilidades de resolución de problemas, cualidades esenciales para forjar programadores competentes y versátiles. Es importante destacar que la POO es una competencia vital en el ámbito de la informática y la tecnología. Independientemente de si los estudiantes se interesan en el desarrollo de software, la inteligencia artificial, la robótica u otros campos afines, la POO será una herramienta esencial en su carrera.

Al finalizar el cursado el estudiante podrá resolver problemas sencillos del mundo real y contará con todas las bases para comenzar a aprender cualquier lenguaje de programación, esto gracias a que la lógica de programación que veremos es universal, y se aplica prácticamente a cualquier lenguaje.

Aportes al perfil de egreso (De la Matriz de Tributación)		
CE - Competencias de Egreso Específicas	*CE-GT* Competencias Genéricas Tecnológicas	*CE-GSPA* Competencias Sociales – Político - Actitudinales
CE 1.1. Especificar, proyectar y desarrollar sistemas de información. CE 1.3. Especificar, proyectar y desarrollar software.	CE-GT 1: Identificar, formular y resolver problemas de informática. CE-GT 2:Concebir, diseñar y desarrollar proyectos de informática. CE-GT 4 Utilizar de manera efectiva las técnicas y herramientas de aplicación en la informática. CE-GT 5: Contribuir a la generación de desarrollos tecnológicos y/o innovaciones tecnológicas.	CE-GSPA 6: Comunicarse con efectividad. CE-GSPA 9: Aprender de forma continua y autónoma.

Expectativas de logro (del Plan de Estudios)

Al acreditar el espacio curricular, las y los estudiantes serán capaces de:

- Identificar, formular y resolver problemas de informática del mundo real con software de calidad.
- Comprender y utilizar conceptos básicos del paradigma orientado a objetos para modelar y resolver problemas del mundo real aplicando buenas prácticas de programación.
- Utilizar de manera efectiva las técnicas y herramientas de aplicación en la informática para el desarrollo de aplicaciones informáticas.

Contenidos mínimos (del Plan de Estudios)

Conceptos básicos del paradigma orientado a objetos. Encapsulamiento. Modulador. Clase abstracta y concreta. Herencia y tipos de herencia. Polimorfismo y sobrecarga de métodos. Excepciones. Lenguajes de programación orientados a objetos. Buenas prácticas de diseño.

Correlativas (Saberes previos/ posteriores del Plan de Correlatividades)

Saberes previos: Programación I, Álgebra

Saberes posteriores: Algoritmos y Estructuras de Datos I, Ingeniería de Software I, Redes de Computadoras, Métodos Numéricos y Programación, Teoría de Base de Datos, Paradigmas de Programación.

2. RESULTADOS DE APRENDIZAJE

RA1 Desarrolla soluciones informáticas mediante un lenguaje orientado a objetos para dar solución a problemas del mundo real.

RA2 Diseña clases y objetos de manera efectiva para resolver problemas de programación específicos, cumpliendo con los principios de encapsulamiento y abstracción

RA3 Aplica la herencia y la encapsulación de manera efectiva para mejorar la reutilización del código.

RA4 Aplica el polimorfismo en la construcción de interfaces y abstracciones para promover la flexibilidad del diseño de software.

3. CONTENIDOS/SABERES (Organizados por unidades, ejes u otros)

PROGRAMACIÓN II
Unidad N°1: Fundamentos de la Programación Orientada a Objetos (POO) Conceptos básicos del POO. Historia y evolución de la POO. Ventajas y desventajas. Lenguajes de programación orientados a objetos. Principios fundamentales: encapsulamiento, abstracción, herencia y polimorfismo. Clases y objetos. Atributos y métodos. *Unidad N°2: Encapsulamiento y Modulación* Encapsulamiento: definición y ejemplos. Beneficios del encapsulamiento. Modulación: organización del código en módulos. Clases y objetos: creación y uso. Encapsulamiento: Modificadores de acceso: public, private, protected. Getters y Setters. *Unidad N°3: Herencia y Tipos de Herencia* Herencia: concepto y relación entre clases. Clase base y clases derivadas. Clase abstracta vs. clase concreta. Tipos de herencia: simple, múltiple y jerárquica. Polimorfismo y sus aplicaciones en herencia *Unidad N°4: Composición y agregación* Composición vs Agregación. Relaciones entre clases. Ejemplos de composición y agregación. *Unidad N°5: Polimorfismo y Sobrecarga de Métodos* Polimorfismo: definición y ejemplos. Interfaces y clases abstractas. Sobrecarga de métodos: definición y ejemplos. *Unidad N°6: Excepciones en Programación Orientada a Objetos* Tipos de excepciones. Captura y lanzamiento de excepciones. Creación de excepciones personalizadas. Uso de excepciones para el control de errores. Jerarquía de Excepciones. Buenas prácticas en el manejo de excepciones. *Unidad N°7: Buenas Practicas de Diseño en POO* Principios SOLID: Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion. Principio DRY (Don't Repeat Yourself). Patrones de diseño comunes.

4. MEDIACION PEDAGOGICA (metodologías, estrategias, recomendaciones para el estudio)

La metodología de enseñanza se centra en la combinación de clases teóricas y prácticas altamente participativas, con un énfasis significativo en uso de laboratorios. Este enfoque tiene como objetivo garantizar que los estudiantes adquieran los conocimientos teóricos y prácticos de la asignatura. Las clases se impartirán utilizando una variedad de métodos que incluyen:

1-Explicaciones conceptuales: Se presentarán los conceptos clave a través de definiciones y ejemplos claros.

2. Lectura individual dirigida: Los estudiantes serán guiados en su lectura para comprender y asimilar mejor el contenido.

3. Actividades grupales: Se fomentará la colaboración a través de actividades grupales de análisis, transferencia de conocimientos y validación colectiva

4. Recursos audiovisuales: Se utilizarán proyecciones y videos para ilustrar ciertos temas y enriquecer la comprensión.

5. Prácticas aplicadas: Los estudiantes participarán en ejercicios prácticos individuales y/o en grupos, aplicando los conceptos enseñados en las diferentes unidades temáticas. Se promoverá el trabajo en equipo.

Esta metodología teórico-práctica, continua e integral, tiene como objetivo que los estudiantes adquieran conocimientos, desarrollen actitudes, identifiquen aptitudes y mejoren sus habilidades para comprender y abordar situaciones nuevas, centrándose en la resolución de problemas. Dado que la adaptación a las nuevas tecnologías es esencial en la actualidad, proporcionaremos a los estudiantes los medios y oportunidades necesarios para explorar y estudiar las herramientas ofrecidas por las últimas tecnologías.

Metodología de las clases teóricas:

- Las clases teóricas tienen como objetivo introducir a los estudiantes en los conceptos teóricos fundamentales de la asignatura.
- Cada tema teórico se abordará en clase a través de ejemplos prácticos proporcionados por el profesor.
- La metodología variará entre clases expositivas, donde los profesores explicarán los temas, y clases interactivas que fomentarán la participación de los estudiantes a través de diálogos.

Metodología de clases prácticas:

- Las clases prácticas brindarán a los estudiantes la oportunidad de aplicar los conceptos teóricos, resolver dudas y aclarar conceptos en colaboración con los docentes.
- Los estudiantes trabajarán en ejercicios de forma individual o en grupos, mientras los profesores supervisarán su progreso y estarán disponibles para consultas personales.
- Cada práctica se relacionará con un núcleo temático específico de la materia.

5. INTENSIDAD DE LA FORMACIÓN PRACTICA

Ámbito de formación práctica	Carga horaria	
	Presencial	No presencial
Formación Experimental		
Resolución de problemas del mundo real	**45**	
Actividades de proyecto y diseño de sistemas informáticos		
Instancias supervisadas de formación en la práctica profesional		
Otras actividades		
Carga horaria total	**45**	

6. SISTEMA DE EVALUACIÓN

La asignatura será evaluada de manera continua, con lo cual a lo largo del cursado se considerarán lo siguientes criterios generales de evaluación:

- Asistencia obligatoria al 75% de las clases
- Participación
- Dedicación a la materia
- Responsabilidad

Las evaluaciones son teóricas-prácticas e incluyen la resolución de problemas algorítmicos donde se aplican los aprendizajes correspondientes a las unidades evaluadas. Las evaluaciones se realizan de acuerdo a los contenidos/saberes especificados en el programa.

6.1. Criterios de evaluación

Al momento de la evaluación y para garantizar los resultados de aprendizaje esperados por parte del alumno se tomarán en consideración los siguientes criterios de evaluación:

RA1 Desarrolla soluciones informáticas mediante el un lenguaje orientado a objetos para dar solución a problemas del mundo real.

1.1. Analiza problemas del mundo real y es capaz de identificar cómo se pueden abordar mediante la programación orientada a objetos.

1.2. Desarrolla soluciones informáticas utilizando un lenguaje orientado a objetos que satisfagan los requisitos y restricciones del problema planteado.

1.3. Demuestra la capacidad de traducir problemas del mundo real en modelos orientados a objetos

RA2 Diseña clases y objetos de manera efectiva para resolver problemas de programación específicos, cumpliendo con los principios de encapsulamiento y abstracción

2.1. Diseña clases de manera eficiente que reflejen conceptos del mundo real y sean adecuadas para resolver problemas específicos.

2.2. Aplica correctamente el principio de encapsulamiento al definir atributos y métodos de las clases, asegurando la protección y ocultamiento de la información.

2.3. Utiliza la abstracción de manera efectiva para crear jerarquías de clases y objetos que simplifiquen la representación y manipulación de conceptos complejos.

RA3 Aplica la herencia y la encapsulación de manera efectiva para mejorar la reutilización del código.

3.1. Utiliza la herencia de forma adecuada para crear relaciones de jerarquía entre clases, promoviendo la reutilización de código.

3.2. Demuestra comprensión y aplicación de los conceptos de encapsulación en la herencia, asegurando que los atributos y métodos sean gestionados adecuadamente en las subclases.

3.3. Diseña y estructura clases y objetos de manera que favorezca la extensibilidad y mantenibilidad del código.

RA4 Aplica el polimorfismo en la construcción de interfaces y abstracciones para promover la flexibilidad del diseño de software.

4.1. Implementa interfaces y abstracciones de manera efectiva, permitiendo que objetos de diferentes clases se comporten de manera intercambiable.

4.2. Utiliza el polimorfismo para crear código flexible y genérico que pueda adaptarse a diferentes situaciones y requisitos.

4.3. Diseña y documenta de manera clara y concisa las interfaces y abstracciones para que sean comprensibles y fáciles de utilizar en el desarrollo de software.

Estos criterios de evaluación se centran en las habilidades y competencias específicas que se esperan de los estudiantes en relación con los resultados de aprendizaje mencionados. Los criterios proporcionan una guía clara para evaluar el desempeño de los estudiantes en cada uno de los aspectos clave de la programación orientada a objetos.

6.2. Condiciones de regularidad

Las condiciones para la obtención de la regularidad son las siguientes:

- Aprobar la totalidad de las evaluaciones escritas o sus correspondientes instancias de recuperación (controles / parciales / globales) con un porcentaje mayor o igual al 60 %.
- Aprobar la totalidad de las prácticas obligatorias con una nota mayor o igual al 60%.
- Poseer como mínimo el 75% de la asistencia a clase.

6.3. Condiciones de promoción

Esta asignatura no tiene régimen de promoción directa.

6.4. Régimen de acreditación para

- **Alumnos regulares:**

Examen Final

Aquellos alumnos que regularizan la materia deberán rendir un examen final en alguna de las fechas establecidas en el calendario académico habilitadas para tal fin. El examen consiste de dos instancias, una instancia práctica donde el alumno debe diseñar e implementar la solución algorítmica a un problema que abarque los aspectos fundamentales de la asignatura, y una instancia teórica donde se evalúan los conceptos teóricos fundamentales de la asignatura. El examen final se rinde a programa completo, exigiendo su aprobación con una nota mayor o igual a 6 (seis) de acuerdo a lo establecido en la ordenanza N° 108 en cada una de las instancias de evaluación.

- **Alumnos libres:**

Se permitirá rendir en las mesas ordinarias y extraordinarias que sean asignadas en cada ciclo lectivo a todos los alumnos/as que queden en condición de:

D. Estudiante libre en el espacio curricular por pérdida de regularidad (LPPR), por haber rendido CUATRO (4) veces la asignatura, en condición de estudiante regular, sin lograr su aprobación.

El examen final será de similares características que el de alumno regular pero los contenidos serán evaluados más exahustivamente. El examen final se rinde a programa completo, exigiendo su aprobación con una nota mayor o igual a 6 (seis) de acuerdo a lo establecido en la Ordenanza N° 108.

7. BIBLIOGRAFIA

Bibliografía básica:

Titulo	Autor /es	Editorial	Año de Edición	Ejemplares Disponibles	Sitios digitales
Programación orientada a objetos en Java	Blasco, Francisco	Paracuellos de Jarama	2019	Electrónica	Enlace
Java 17 : programación avanzada	Vegas Gertrudix, José M	RA-MA	2022	Electrónica	Enlace
Java: Cómo programar	Deitel&Deitel	Pearson 9a edición	2012	1 (uno)	Inventario: OPE34580 Ubicación: 005 DEI

Bibliografía complementaria:

Titulo	Autor /es	Editorial	Año de Edición	Ejemplares Disponibles	Sitios digitales
Programación orientada a objetos	Moreno Pérez, Juan Carlos.	RA-MA	2015	Electrónica	Enlace
Fundamentos de la programación orientada a objetos : una aplicación a las estructuras de datos en Java	Ruiz Rodríguez, Ricardo	Miami : El Cid Editor,	2014	Electrónica	Enlace

7.1. Recursos digitales del espacio curricular (enlace a aula virtual y otros)

8. FIRMAS

Javier Rosenstein

V°B° DIRECTOR/A DE CARRERA RESPONSABLE A CARGO

Fecha

DOCENTE

Fecha: 12/03/2024