Ministerio de Cultura y Educación **(Programa del año 2026)**
Universidad Nacional de San Luis **(Programa en trámite de aprobación)**
Facultad de Ciencias Físico Matemáticas y Naturales **(Presentado el 21/04/2026 20:30:00)**
Departamento: Informatica
Area: Area IV: Pr. y Met. de Des. del Soft.

I - Oferta Académica

Materia	Carrera	Plan	Año	Período
PROGRAMACION II	ING. INFORM.	OCD-3-2/2025	2026	1° cuatrimestre

II - Equipo Docente

Docente	Función	Cargo	Dedicación
BERON, MARIO MARCELO	Prof. Responsable	P.Adj Exc	40 Hs
BAIGORRIA FERNANDEZ, LORENA S.	Prof. Colaborador	P.Adj Exc	40 Hs
SANCHEZ, HECTOR ENRIQUE	Responsable de Práctico	JTP Exc	40 Hs
LEIVA OLGUIN, GASTON EMANUEL	Auxiliar de Práctico	A.1ra Simp	10 Hs

III - Características del Curso

Credito Horario Semanal				
Teórico/Práctico	**Teóricas**	**Prácticas de Aula**	**Práct. de lab/ camp/ Resid/ PIP, etc.**	**Total**
Hs	2 Hs	1 Hs	3 Hs	6 Hs

Tipificación	Periodo
B - Teoria con prácticas de aula y laboratorio	1º Cuatrimestre

Duración			
Desde	**Hasta**	**Cantidad de Semanas**	**Cantidad de Horas**
11/03/2026	23/06/2026	15	90

IV - Fundamentación

Los paradigmas de programación son uno de los núcleos basales de los profesionales en informática. Cada paradigma de programación proporciona métodos, estilos de programación, etc. que facilitan el abordaje de problemas informáticos. Por esta razón, impartir los conceptos relacionados con esta temática es primordial en las carreras de ingeniería. Uno de los paradigmas que ha adquirido mucho énfasis en los últimos tiempos, debido a sus fundamentos y a los métodos y herramientas que proporciona para la solución de problemas, es la Programación Orientado a Objetos (POO).
POO provee mecanismos interesantes que liberan al programador de detalles de implementación que incrementan la posibilidad de introducir errores de programación. Además, posibilita que el código desarrollado pueda ser reutilizado evitando de esta manera la pérdida de tiempo producida por la re-implementación de soluciones. Otro paradigma reciente que ha alcanzado mucha importancia es el proporcionado por la Programación Dirigida por Eventos. Esta forma de concebir la programación tiene como principal objetivo proporcionar los métodos y herramientas adecuadas para desarrollar sistemas donde el flujo de ejecución está determinado por el usuario, acciones internas del sistema u otros externos. El creador de un programa dirigido por eventos debe definir los eventos que manejarán su programa y las acciones que se realizarán al producirse cada uno de ellos, lo que se conoce como el administrador de evento. Los eventos soportados estarán determinados por el lenguaje de programación utilizado, por el sistema operativo e incluso por eventos creados por el mismo

programador. La Programación Dirigida por Eventos es la base para desarrollar interfaces de usuario aunque también es útil para desarrollar otros tipos de aplicaciones como por ejemplo: módulos del núcleo de un sistema operativo. La programación basada en scripts es importante en el ámbito de procesamiento por lotes y archivos de comando en sistemas operativos y así como también su evolución hacia lenguajes de propósito general. Como se puede observar en las breves descripciones presentadas en los párrafos precedentes, los paradigmas Orientados a Objetos, Dirigidos por Eventos y la programación Script son fundamentales para el desarrollo de aplicaciones industriales y científicas y por consiguiente son tópicos muy importantes que deben conformar el trasfondo de conocimientos de todo Ingeniero.

V - Objetivos / Resultados de Aprendizaje

Objetivos / Resultados de Aprendizaje
Al finalizar el curso se espera que el alumno sea capaz de:
* Conocer y aplicar correctamente los conceptos fundamentales del paradigma de programación orientada a objetos (POO).
* Desarrollar una visión clara del tipo de situaciones en las que el paradigma POO es adecuado y la forma en que los conceptos de clase, herencia, polimorfismo y ligadura dinámica de mensajes interactúan.
* Desarrollar y ejecutar programas cortos usando un lenguaje representativo del POO.
* Comprender los conceptos fundamentales de la programación dirigida por eventos (PDE).
* Aplicar esta forma de programación en ejemplos concretos que involucren eventos y manejadores de eventos.
* Comprender los conceptos fundamentales de la programación de la programación basada en scripts.

Durante el dictado de la asignatura se abordan los siguientes ejes transversales:
* Concepción, diseño y desarrollo de proyectos de de ingeniería en sistemas de información/informática.
* Utilización de técnicas y herramientas de aplicación en de ingeniería en sistemas de información/informática.
* Generación de desarrollos tecnológicos y/o innovaciones tecnológicas.
* Fundamentos para el aprendizaje continuo.
* Fundamentos para el desempeño en equipos de trabajo.
* Fundamentos para una comunicación efectiva.
* Fundamentos para una actuación profesional ética y responsable.

VI - Contenidos

Contenidos Mínimos

Programación Orientada a Objetos. Tipos de datos abstractos. Ocultamiento de la información y encapsulamiento. Definición de clases. Control de Acceso. Herencia. Subclases. Herencia simple y múltiple. Tipos de datos elementales y estructurados en POO. Estructuras de control. Polimorfismo y ligadura dinámica. Clases y Métodos abstractos. Ejemplos de un lenguaje POO particular. Paquetes. Interfaces. Excepciones. Entrada-salida. Ambientes de programación. Desarrollo de aplicaciones usando librerías.
Introducción a la Programación dirigida por eventos (PDE). PDE en POO. PDE e interfaces de usuario gráficas. Su uso en un lenguaje que responda al paradigma POO. Programación basada en scripts. Conceptos básicos de scripts. Características principales de un lenguaje script Phyton o simiJlar.

Para cada unidad se deja disponible el material correspondiente a los contenidos de la unidad, las diapositivas de clase, el apunte teórico y su correspondiente trabajo práctico en el repositorio digital. Para una mejor organización, tanto de los estudiantes como del equipo docente, se presenta un cronograma en donde se detalla la información de la materia: días y horarios de clase, horarios de consulta, fechas de parciales y sus respectivas recuperaciones, fecha de inicio y fin de los laboratorios.

Unidad I: Introducción a Java
Sintaxis y semántica de Java. Definición de variables. Visibilidad y Vida de las variables. Tipos de dato primitivos. Conversión de tipos. Uso de Operadores mediante ejemplos sencillos en laboratorio: Aritméticos, Asignación, instanceof, condicional, incrementales, relacionales, lógicos, concatenación de caracteres. Precedencia entre operadores. La clase String. Integer, Float y Number. Arreglos. Wrappers. Estructuras de control de flujo: Selección if else, Selección switch, Bucle

while, Bucle for, Bucle do while, Sentencia break, Sentencia continue. Comentarios en Java. Caracteres especiales.

Unidad II: Programación Orientada a Objetos - Lenguaje Java
Programación Orientada a Objetos y Java. Clase. Ejemplos de construcción de una clase en Java. Clases y Objetos en Java. Atributos. Métodos en Java. Pasaje de parámetros. Encapsulamiento. Control de acceso. Constructores. Herencia en Java. Sub-clases y super-clases. This. Super. Herencia múltiple. Redefinición de métodos heredados. Accesibilidad a clases o interfaces, variables. Polimorfismo. Ligadura Dinámica. Clase abstracta. Abstracción en Java. Interfaz; ejemplos en Java. Métodos Abstractos. Paquetes. Relaciones entre paquetes.

Unidad III: Colecciones
El framework Collection en Java: Interfaces, implementaciones y algoritmos. La interfaces Collection, List, Queue, Dequeue, Map, Set y sus respectivas implementaciones y algoritmos. Iteradores.

Unidad IV: Manejo de Archivos en Java
Archivos. Conceptos Fundamentales. Apertura, lectura y cierre de archivos. Archivos en Java. FileReader. FileWriter. Buffers. BufferedReader. BufferedWriter.

Unidad V: Mecanismos de Manejo de Excepciones en Java
Manejo de Excepciones. Ventajas. Manejo de excepciones en Java: La clase Throwable; bloques try, catch y finally. Tipos de Excepciones: Excepciones verificadas y no verificadas. Creación de excepciones propias. La sentencia throw. La clase Exception como superclase. La cláusula throws. Estudio comparativo y mediante ejemplos en laboratorio.

Unidad VI: AWT y Swing
La interfaz gráfica. AWT. Concepto y uso. Componentes y eventos soportados. Jerarquía de componentes, eventos y sus relaciones. Estructura de una aplicación AWT. Swing. Estructura de una aplicación Swing. Swing vs. AWT. Componentes Swing: Contenedores. JFrame. JComponent JDialog. JApplet. Jpanel. Organización en el IDE. Manejo de ventanas y layout. Introducción a la Programación Dirigida a Eventos. El modelo de eventos de Java. Programación de eventos GUI en Java.

Unidad VII: Programación Dirigida por Eventos
El Paradigma de Programación Dirigida a Eventos. Diferencias entre las distintas formas de paradigmas de programación vistos. El usuario dirigiendo el flujo del programa. Detección de eventos. Problemática. Interfaces gráficas con el usuario en un entorno dirigido a Eventos. Herramientas visuales de desarrollo. Lenguajes utilizados en la Programación Dirigida a Eventos. Referencias. Enlaces externos. Definición de Eventos y notificaciones. Subscripción de Eventos. Proceso de Suscripción. Modelos. Protocolos de Entrega de Notificación.

Unidad VIII: Programación de Scripts
Programación basada en scripts. Conceptos básicos de scripts. Orígenes de la programación script (shell scripts). Desarrollo moderno de lenguajes de script. Tipos de lenguajes de scripting. Características generales. Introducción al lenguaje Python. Tipos: entero, flotantes, caracter, booleano, strings, listas, tuplas y diccionarios. Sentencias. Funciones. Definición de funciones. Invocación de funciones. Aplicaciones.

VII - Plan de Trabajos Prácticos

Metodología de la Enseñanza

Los trabajos prácticos correspondientes a las unidades del programa consisten en problemas que deben ser resueltos por los estudiantes guiados por los docentes. En cada trabajo práctico los estudiantes deben resolver ejercicios que están relacionados con la teoría que se ha dictado en ese momento. Los ejercicios están organizados por orden de complejidad comenzando por ejercicios más simples, luego los de complejidad media y por último los más complejos. En cada práctico, con el fin de realizar una evaluación formativa, se resuelven ejercicios (correspondientes a cada nivel de complejidad) en el pizarrón de forma tal de fomentar la comunicación, hacer explícita la relación de conceptos, y detectar los inconvenientes que los estudiantes puedan tener.
En caso de detectar un problema, el docente resuelve ejercicios del mismo tipo y propone otros nuevos con la finalidad de

que el estudiante pueda asimilar los conceptos y sus relaciones. Además, con este tipo de actividad, se busca que el estudiante empiece a expresarse de manera escrita utilizando un vocabulario acorde a los contenidos vistos, a socializar y reflexionar sobre sus respuestas y a detectar los conceptos no comprendidos.
Las clases están pensadas para que el alumno participe activamente respondiendo preguntas que el docente realiza las cuales pueden ser de teóricas o prácticas y resolviendo ejercicios en el pizarrón. A medida que se van desarrollando las clases, el docente va realizando análisis del desempeño de los estudiantes de forma tal de poder establecer el nivel de grupo y proponer actividades que resuelvan las dificultades que se puedan presentar.
Todos los ejercicios pueden ser validados utilizando herramientas (IDEs, compiladores e intérpretes) que permiten la ejecución de los ejercicios (dado que en la materia se enseñan lenguajes de programación utilizados tanto en el ambiente académico como en el profesional). Con el fin de promover una relación más estrecha con el mercado laboral y también con el de investigación, en la materia se utilizan herramientas de uso profesional asociadas a los lenguajes que se enseñan en la materia (los cuales también son muy utilizados en el contexto académico y el empresarial.
Durante todo el desarrollo de la materia los docentes hacen énfasis en conceptos del paradigma en sí remarcado como se concretan en los lenguajes estudiados e informando que el proceso de capacitación es continuo, en primera instancia es guiada por los docentes de la materia y luego conducido por los propios estudiantes. Se refuerza esta idea basándose en que los lenguajes de programación evolucionan constantemente lo que conduce a que aparezcan nuevos lenguajes y por consiguiente se deban estudiar para que la inserción en el ambiente laboral sea más sencilla. También se promueve el trabajo en equipo, el desarrollo de soluciones innovadoras de los prácticos y el uso de vocabulario técnico, preciso y claro cuando los estudiantes explican los prácticos en clase.
La materia consta de laboratorios en los cuales el estudiante integra los conocimientos adquiridos en los prácticos de aula. En estos laboratorios también se le solicita al estudiante que use técnicas de desarrollo de sistemas y de resolución creativa de problemas para promover la innovación. Una vez finalizado cada laboratorio los estudiantes realizan una presentación la cual es evaluada por los docentes. En la evaluación se tiene en cuenta el vocabulario utilizado, la forma de comunicación, la calidad de la solución y las tareas realizadas por cada miembro del equipo.

Práctico I: Introducción a Java
Se introducen los conceptos iniciales del lenguaje de programación Java. Los ejercicios están orientados a realizar programas simples que usen clases primitivas, expresiones, sentencias de flujo de control.

Objetivo: Con este práctico se pretende que el alumno conozca las construcciones de programación provistas por el lenguaje de programación Java.

El práctico incluye ejercicios donde se pretende que el estudiante:

* Utilice las sentencias (secuencia, selección e iteración) que provee el lenguaje java para resolver ejercicios básicos de programación.

* Aplique en las soluciones algoritmos simples

En este práctico se realiza el siguiente laboratorio:

Laboratorio I: Implementación y ejecución de programas cortos usando el lenguaje orientado a objetos Java.

Práctico II: Programación Orientada a Objetos - Lenguaje Java
En este práctico se introducen los conceptos necesarios para la creación de clases y de las relaciones posibles entre ellas en el Lenguaje de Programación Java.

Objetivo: En este práctico se pretende que el alumno defina clases, herencia y polimorfismo.

El práctico incluye ejercicios donde se pretende que el estudiante:

* Identifique e implemente clases con todos los miembros necesarios.

* Identifique e implemente relaciones de herencia entre las clases.

* Aplique soluciones polimórficas a problemas típicos de programación.

* Integre todos los conceptos vistos en la unidad.

En este práctico se desarrolla el siguiente laboratorio:

* Laboratorio II: Implementación y ejecución de programas que impliquen el uso de las jerarquías de clases disponibles para el lenguaje orientado a objetos Java.

Práctico III: Colecciones
En este práctico se desarrollarán ejercicios con la utilización de las interfaces de: list, map, etc.

Objetivo: En este práctico se pretende que el estudiante:

* Sepa utilizar las estructuras de datos provistas por Java como así también utilizar los algoritmos apropiados para resolver problemas de programación

* Pueda definir sus propias estructuras de datos a partir de la infraestructura de estructuras de datos provistas por Java.

El práctico incluye ejercicios donde se pretende que el estudiante:

* Utilice de forma directa las estructuras de datos provistas por el lenguaje de programación.

* Seleccione y utilice las estructuras de datos provistas por el lenguaje de programación Java en sus soluciones.

* Cree estructuras de datos simples y las incorpore, cuando el problema lo amerite, a la jerarquía de clases del lenguaje de programación.

* Elabore criterios de selección de estructura de datos y algoritmos.

En este práctico se desarrolla el siguiente laboratorio:

* Laboratorio III: Implementación y ejecución de programas que impliquen el uso de la infraestructura de colecciones provistas por el lenguaje de programación Java.

Práctico IV: Manejo de Archivos en Java
En este práctico se desarrollarán ejercicios que utilicen las funciones provistas por Java para la manipulación de archivos.

Objetivo: En este práctico se pretende que el estudiante:

* Aprenda a administrar la memoria secundaria.

El práctico incluye ejercicios donde se pretende que el estudiante:

* Aprenda a aplicar los pasos que son necesarios para trabajar con archivos.

* Abra archivos usando los diferentes modos que provee el lenguaje de programación.

* Escriba archivos en memoria secundaria.

En este práctico se desarrolla el siguiente laboratorio:

Laboratorio IV: Construcción y ejecución de programas que:

* Almacenen en memoria secundaria información ingresada por un usuario.

* Escriban en memoria secundaria información ingresada por el usuario.

* Recuperen y procesen información de memoria secundaria, procesen la información recuperada y escriban información la información resultante del procesamiento en memoria secundaria.

Práctico V: Mecanismos de Manejo de Excepciones en Java
Los ejercicios desarrollados en el práctico correspondiente a la unidad III serán mejorados introduciendo rutinas de manejo de excepciones, errores, etc.

Objetivo: En este práctico se pretende que el estudiante:

* Aprenda a aplicar el concepto de excepción.

* Combine los mecanismos de manejo de errores tradicionales con los provisto por las excepciones.

* Elabore criterios respecto a cuando puede aplicar: i) El mecanismo de manejo de errores tradicional; ii) El de excepciones, o iii) Una combinación de los mismos.

El práctico incluye ejercicios donde se pretende que el estudiante:

* Utilice el mecanismo de manejo de excepciones provisto por el lenguaje de programación.

* Analice el funcionamiento del mecanismo de excepciones.

* Use las excepciones provistas por el lenguaje de programación.

* Defina sus propias excepciones.

* Combine las estrategias de manejo de errores tradicionales con las de manejo de excepciones.

En este práctico se desarrolla el siguiente laboratorio:

Laboratorio V: Ejecución y corridas de programas donde el estudiante utilice excepciones provistas por el lenguaje de programación y defina sus propias excepciones.

Práctico VI: AWT y Swing

Objetivo: En este práctico se pretende que el estudiante:
Construya interfaces gráficas utilizando AWT y SWING.

El práctico incluye ejercicios donde se pretende que el estudiante:

* Utilice el mecanismo utilice las widgets provistas por AWT y SWING.

* Analice el funcionamiento de las componentes de AWT y SWING.

* Defina sus propias componentes.

En este práctico se desarrolla el siguiente laboratorio:

Laboratorio VI: Entorno de desarrollo en Java (AWT-SWING). Implementación de programas con manejo de GUIs manejada por eventos.

Práctico VII: Programación Dirigida por Eventos

Objetivo: En este práctico se pretende que el estudiante:

* Aprenda a utilizar los eventos provistos por AWT y SWING.

* Defina sus rutinas de manejo de eventos.

El práctico incluye ejercicios donde se pretende que el estudiante:

* Utilice el mecanismo utilice los eventos provistos por AWT y SWING.

* Analice el funcionamiento de los eventos y sus formas de propagación.

En este práctico se desarrolla el siguiente laboratorio:

Laboratorio VII: Desarrollo de programas cortos utilizando la programación dirigida a eventos con rutinas de manejo de eventos de complejidad media.

Práctico VIII: Programación de Scripts

Objetivo: En este práctico se pretende que el estudiante:

* Aplique los conceptos de programación de scripts.

* Aprenda el lenguaje de programación Python – Paradigma Imperativo.

El práctico contiene ejercicios donde se pretende que el estudiante:

* Utilice las sentencias y construcciones del lenguaje de programación

* Elabore soluciones utilizando estrategias de desarrollo de sistemas.

* Utilice diferentes módulos provistos por el lenguaje.

* Defina sus propios módulos.

En este práctico se desarrolla el siguiente laboratorio:

Laboratorio VIII: Desarrollo de scripts utilizando el lenguaje Python.

A continuación se explica como se abordan y evalúan los ejes transversales empleados en la materia:

* Concepción, diseño y desarrollo de proyectos de de ingeniería en sistemas de información/informática
¿Cómo se aborda?
Se aborda mediante la realización de proyectos de laboratorio, donde los estudiantes deben aplicar de manera correcta los métodos, técnicas y herramientas trabajados en la materia. Se promueve que cada proyecto transite las etapas de básicas del

proceso de desarrollo de software, favoreciendo la integración de conocimientos teóricos y prácticos. Este enfoque busca que los estudiantes adquieran competencias para planificar, implementar y probar soluciones informáticas, consolidando así su capacidad de trabajo en equipo y su preparación para enfrentar desafíos propios de la ingeniería en informática.

¿Cómo se evalúa?
La evaluación se realizará a través de la valoración integral de los proyectos desarrollados en los laboratorios. Se considerará la correcta aplicación de los métodos, técnicas y herramientas trabajados en la materia, así como la coherencia en el tránsito por las etapas básicas del proceso de desarrollo de software: concepción, diseño, implementación y pruebas. Se prestará especial atención a la capacidad de los estudiantes para integrar conocimientos teóricos y prácticos, la pertinencia de las soluciones propuestas y la calidad del trabajo en equipo. El equipo docente efectuará correcciones y sugerencias orientadas a fortalecer la planificación, la ejecución y la validación de los proyectos, consolidando así competencias esenciales para el ejercicio profesional en la ingeniería en informática.

* Utilización de técnicas y herramientas de aplicación en de ingeniería en sistemas de información/informática

¿Cómo se aborda?
Fomentando, durante todo el desarrollo de la materia, el uso adecuado de las técnicas de programación y la aplicación de herramientas reconocidas en el mercado. Se promueven prácticas correctas de modularización de sistemas de software, integrando teoría y práctica para que los estudiantes adquieran competencias que les permitan desarrollar soluciones informáticas eficientes, escalables y acordes a los estándares profesionales vigentes.

¿Cómo se evalúa?
Mediante prácticos en los que los estudiantes apliquen técnicas de programación y herramientas reconocidas en el mercado, demostrando modularización, eficiencia y precisión de los conceptos dados en la materia. Se valorarán tanto la calidad técnica del código como la integración teoría–práctica y la capacidad de trabajar colaborativamente.

* Generación de desarrollos tecnológicos y/o innovaciones tecnológicas
¿Cómo se aborda?
Se aborda promoviendo, especialmente en los espacios de laboratorio, el diseño de soluciones que integren creatividad y pertinencia técnica. Se incentiva la elaboración de propuestas innovadoras que contemplen la intervención del usuario, fortaleciendo la capacidad de los estudiantes para transformar conocimientos en desarrollos aplicables y relevantes. De este modo, se fomenta la formación de profesionales capaces de generar aportes tecnológicos originales y de impacto en su entorno.

¿Cómo se evalúa?
La evaluación se realizará considerando la originalidad de las propuestas, la pertinencia de las soluciones planteadas y la calidad técnica de los desarrollos. Se valorará especialmente la capacidad de los estudiantes para integrar la intervención del usuario en sus proyectos, así como el uso adecuado de herramientas informáticas y la aplicación correcta de los conceptos. El equipo docente efectuará correcciones y sugerencias en todo momento con el fin de establecer base sólidas para la formación de profesionales capaces de generar aportes tecnológicos innovadores.

* Fundamentos para el aprendizaje continuo
¿Cómo se aborda?
Se aborda mediante la integración de repasos sistemáticos de contenidos previos en las actividades teóricas y prácticas, promoviendo la participación activa de los estudiantes a través de consultas y reflexiones. Se enfatiza la evolución constante de la tecnología y la necesidad de adquirir estrategias de estudio e investigación que faciliten la incorporación de nuevos conocimientos. De este modo, se fomenta la autonomía y la disposición permanente para el aprendizaje, competencias esenciales en la formación de los futuros ingenieros en informática.

¿Cómo se evalúa?
Se realizará considerando la participación de los estudiantes en los repasos de contenidos previos y en las consultas que acompañan el inicio de cada clase. Se valorará la capacidad de relacionar saberes adquiridos con nuevos desarrollos, así como la disposición para incorporar estrategias de estudio e investigación que favorezcan la actualización permanente frente a la evolución tecnológica. El equipo docente efectuará correcciones y sugerencias orientadas a fortalecer la autonomía y la

actitud crítica de los estudiantes, consolidando así la competencia de aprendizaje continuo como parte esencial de su formación profesional.

* Fundamentos para el desempeño en equipos de trabajo

¿Cómo se aborda?
Se desarrollan prácticos donde los estudiantes ejercitan la coordinación de roles, la responsabilidad compartida y la toma de decisiones colectivas. De este modo, se fortalece la capacidad de actuar eficazmente en equipos interdisciplinarios, condición esencial para el ejercicio profesional.

¿Cómo se evalúa?
Se evalúa mediante una estrategia que contemple la coordinación de roles, la responsabilidad compartida y la toma de decisiones colectivas. Se valorará la participación equitativa de los integrantes, la calidad de la comunicación, el cumplimiento de los objetivos grupales y la capacidad de reflexión sobre la dinámica del equipo. De este modo, la evaluación permitirá reconocer tanto el logro técnico alcanzado como el fortalecimiento de competencias esenciales para el ejercicio profesional en contextos interdisciplinarios.

* Fundamentos para una comunicación efectiva

¿Cómo se aborda?
Se aborda integrando la práctica comunicacional en todas las actividades académicas, tanto escritas como orales. Se promueve el uso riguroso de la terminología y notaciones propias de la informática, junto con la claridad en la exposición de conceptos de programación. La pertinencia de las intervenciones es un criterio central, y el equipo docente acompaña el proceso mediante correcciones y sugerencias que orientan hacia una comunicación precisa y adecuada al contexto profesional. De este modo, la comunicación se convierte en un componente formativo inseparable del desarrollo técnico.

¿Cómo se evalúa?
En todas las actividades que implican la participación activa de los estudiantes, tanto escritas como orales, se prestará especial atención al empleo de terminología y notaciones propias de los desarrollos informáticos, así como a la claridad con que se expresen los conceptos de programación utilizados. Además, se verificará que las intervenciones de los estudiantes sean pertinentes. En todos los casos, el equipo docente realizará las correcciones y/o sugerencias necesarias para una correcta comunicación, según el contexto.

* Fundamentos para una actuación profesional ética y responsable

¿Cómo se aborda?
Se aborda mediante la incorporación de prácticas que promueven la disciplina académica, la transparencia y el respeto por la autoría intelectual. La organización de cronogramas de evaluaciones y actividades con plazos definidos fomenta la responsabilidad en la gestión del tiempo y el cumplimiento de compromisos. Los requisitos de asistencia y la presentación de certificados para justificar excepciones refuerzan la honestidad y la formalidad durante el cursado. Asimismo, se impulsa el reconocimiento explícito de la autoría en el uso de algoritmos y desarrollos ajenos, consolidando valores de integridad y respeto profesional. De este modo, la ética se integra como un componente esencial en la formación de los futuros ingenieros en informática.

¿Cómo se evalúa?
Se compartirá un cronograma de evaluaciones y otras actividades. Para todas ellas, se establecerán plazos y formas de entrega. Se exigirán requisitos de asistencia a clases para regularizar y/o promocionar la materia. Se requerirá la presentación de los certificados correspondientes a quienes soliciten algún tipo de flexibilidad excepcional con causa que lo justifique. Se fomentará el reconocimiento de autoría cuando se usen algoritmos realizados por programadores que no forman parte del equipo de trabajo.

VIII - Regimen de Aprobación

La materia se desarrolla con la modalidad de promoción sin examen final. Existen dos niveles:
a) Regularización solamente: Para regularizar la materia se deberá:
1.- Tener como mínimo un 80% de asistencia a clases prácticas.
2.- Tener los prácticos, solicitados por la cátedra, aprobados.
3.- Aprobar el parcial o sus respectivas recuperaciones con un mínimo del 60%.
b) Promoción sin examen final: Para regularizar y aprobar la materia se deberá:
1.- Cumplir con los requisitos a.1 y a.2.
2.- Aprobar el parcial o sus respectivas recuperaciones con un mínimo del 70%.
3.- Aprobar una evaluación integradora global con un mínimo del 70%.
Se otorga dos recuperaciones para el parcial y el práctico de máquina según reglamentación vigente.
Aquellos alumnos que sólo regularicen la materia deberán rendir un examen final, en los turnos establecidos.

Alumnos Libres: Según la reglamentación vigente (Art. 27 de Ord. 13/03 CS).

IX - Bibliografía Básica

[1] Gervais, Luc. Aprender Programación Orientada a Objetos con Java. 2Da Edición. ENI Ediciones. 2025. ISBN: 978-2-409-05028-2.
[2] Sznajdleder, Pablo. Java a Fondo – Curso de Programación (5ª edición). Alfaomega. 2024. ISBN: 9789878983776.
[3] Slatkin, Brett. Effective Python 2da Edition. Addison-Wesley Professional. 2025. ISBN: 978-0-13-485398-7.
[4] Ramalho, Luciano. Fluent Python 2da Edición. O'Reilly Media. 2025. ISBN: 978-1-492-09189-7.
[5] Cachero, Cristina, Ponce de León, Pedro J., Saquete, Estela. "Introducción a la Programación Orientada a Objetos". Universidad de Alicante. Publicación Alicante. 2006. ISBN: 978-84-7908-873-6
[6] Liang, Daniel. Introduction to Java Programming, comprehensive version. Prentice Hall. 8 de. 2010. ISBN-13: 978-0-13-213080-6.
[7] Grant Palmer. Java Event Handling. Prentice Hall. ISBN 0-13-041802-1. 2001.
[8] David Luckham. The Power of Events - An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley. ISBN: 0-201-72789-7. 2002.
[9] Kathy Sierra, Bert Bates, and Trisha Gee.Head First Java™. Third Edition. O'Reilly. ISBN: 978-149-191077-1. 2022.
[10] Eric Matthes. Curso Intensivo de Python: Introducción Práctica a la Programación Basada en Proyectos. ISBN-10: 0-7645-9654-3. Anaya Multimedia. 2021. ISBN13 9788441543348.
[11] Bertrand Meyer. Touch of Class: Learning to Program Well with Objects and Contracts. Springer, 2009. ISBN: 978-3-540-92144-8.
[12] Bertrand Meyer. The power of abstraction, reuse and simplicity: an object-oriented library for event-driven design, in Festschrift in Honor of Ole-Johan Dahl eds. Olaf Owe et al. Springer-Verlag. Lecture Notes in Computer Science .2004.
[13] The Java Tutorial. http://java.sun.com/docs/books/tutorial/index.html
[14] Herbert Schildt. Java: A Beginner's Guide. Third Edition. McGraw-Hill/Osborne. 2005.
[15] Python Programming Language. Official Website http://www.python.org/doc/.
[16] Mark Lutz. Programming Python. O'Reilly Media, Inc. 4th Edition. 2010. ISBN: 9780596158101.
[17] Peter Norton, Alex Samuel, David Aitel, et.al. Beginning Python. Wiley Publishing, Inc. 2005. ISBN-10: 0-7645-9654-3
[18] Material provisto por la cátedra.

X - Bibliografia Complementaria

[1] Quentin Charatan, Aaron Kans. Java in two Semesters. Fourth Edition. Srpinger. 2019. ISSN1868-0941.
[2] Blich, Joshua. Effective Java (3ª Edición).Addison-Wesley Professional. 2017. ISBN-13: 978-0134685991.

XI - Resumen de Objetivos

Al finalizar el curso se espera que el alumno sea capaz de:
- Conocer y aplicar los conceptos fundamentales del paradigma de programación orientada a objetos (POO).

- Comprender y aplicar los conceptos fundamentales de la programación dirigida por eventos (PDE).
- Comprender y aplicar los conceptos fundamentales de la programación de script.

XII - Resumen del Programa

-Programación Orientada a Objeto. Ocultamiento de la información y encapsulamiento. Definición de clases. Control de Acceso. Herencia. Subclases. Ejemplos en un lenguaje de POO particular. Herencia simple y múltiple. Tipos de datos elementales y estructurados en POO: Java. Estructuras de control. Polimorfismo y ligadura dinámica. Clases y Métodos abstractos. Ejemplos de un lenguaje POO particular. Paquetes. Interfaces. Excepciones. Entrada-salida. Ambientes de programación. Desarrollo de aplicaciones usando librerías.
- Programación dirigida por eventos (PDE). Desde la programación (secuencial) estructurada a la programación dirigida por eventos. Eventos. Creando y ligando manejadores de eventos. PDE en POO. PDE e interfaces de usuario gráficas.
- Programación basada en scripts. Conceptos básicos de scripts. Desarrollos modernos de lenguajes tipo script. Tipos de lenguajes de scripting. Características generales. Introducción al lenguaje Python Script.

XIII - Imprevistos

Los imprevistos serán resueltos por la cátedra en la medida que aparezcan.
Correo de Contacto: Mario Berón -- mberon@email.unsl.edu.ar

XIV - Otros

Las vías de comunicación con los estudiantes son las siguientes:
*Correo electrónico del responsable de la materia: mberon@email.unsl.edu.ar
*Oficina: 3 Bloque II - Primer Piso
*Teléfono: +54 (266) 4520300 - Interno: 2103

ELEVACIÓN y APROBACIÓN DE ESTE PROGRAMA	
	Profesor Responsable
Firma:	
Aclaración:	
Fecha:	