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 19/03/2026 11:19:24)**
Departamento: Informatica
Area: Area IV: Pr. y Met. de Des. del Soft.

I - Oferta Académica

Materia	Carrera	Plan	Año	Período
PROGRAMACIÓN II	LIC.CS.COMP.	RD-3 -1/20 23	2026	1° cuatrimestre

II - Equipo Docente

Docente	Función	Cargo	Dedicación
BERON, MARIO MARCELO	Prof. Responsable	P.Adj Exc	40 Hs
ALBORNOZ, MARIA CLAUDIA	Responsable de Práctico	JTP Exc	40 Hs
ZALDUA, ANALIA MAGDALENA	Responsable de Práctico	JTP Simp	10 Hs
CABALLERO, WALTER DAMIAN	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**
3 Hs	Hs	Hs	4 Hs	7 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	105

IV - Fundamentación

En este curso se inicia a los alumnos en dos paradigmas:
Progr. Orientada a Objetos (POO) y Progr. Declarativa Funcional y se refuerzan y completan los conocimientos y aptitudes que adquirieron en Programación I en el paradigma de la programación estructurada. El contenido mencionado prepara al alumno para poder comparar las características de los distintos modelos formales subyacentes en el desarrollo de programas en los tres paradigmas. Las aptitudes desarrolladas a través de la práctica en esta materia, servirán de base tanto para el desarrollo de aplicaciones reales en los distintos paradigmas como para el análisis comparativo formal lenguajes.
La semántica clara de los lenguajes funcionales facilita la comprensión de las técnicas elementales de programación y el razonamiento formal; el alto nivel de expresividad de estos lenguajes permite abordar tempranamente problemas complejos de programación, trasladando el eje de la clase, desde el mecanismo de programación hacia la solución del problema lo cual favorece la estructuración del proceso de resolución de problemas, relegando la sintaxis del lenguaje funcional a un segundo plano y permitiendo al alumno concentrarse en conceptos generales de programación.
El paradigma de orientación a objetos es, en la actualidad, la principal tendencia para el desarrollo de software, ya que ha demostrado ser válido en la construcción de sistemas en toda clase de dominios de problemas, tamaños y complejidades. En la asignatura los conceptos de este paradigma se presentan en el lenguaje de programación Java, que se ha consolidado como

uno de los lenguajes de programación más utilizados para el desarrollo de aplicaciones multiplataformas en el ámbito profesional.
En la asignatura se aborda el desarrollo de un proyecto en Java en el que se implementan de manera incremental conceptos de la POO, culminando con el desarrollo de una aplicación Gráfica de un modelo simplificado de un sistema de Gestión particular. El proyecto mencionado, se codifica en un Entorno de Desarrollo Integrado (IDE) a elección de cada alumno a partir de una lista de sugerencias de la Cátedra. Cabe señalar que los IDE, al ser diseñados para su uso profesional y no con fines didácticos, configuran un elemento que dificulta el aprendizaje debido a la cantidad de herramientas y opciones que abruman a los estudiantes que se inician en la programación orientada a objetos. En la asignatura se desarrollan habilidades que los ayuden a enfrentar estas dificultades y lograr algún grado de competencia. El desarrollo de interfaces gráficas de usuario utilizando librerías de Java (Swing, Javafx, etc.) y herramientas del IDE, se encuentra articulado con la asignatura Ingeniería de Software.
También se explora el tema de la verificación y validación de software, investigado diferentes estrategias de prueba para código desarrollado en diferentes paradigmas.

V - Objetivos / Resultados de Aprendizaje

Objetivos Generales
- Profundizar el análisis de problemas resolubles con computadora, poniendo énfasis en la modelización, abstracción de funciones y en la modularización de los mismos utilizando diferentes paradigmas de programación.
- Promover conductas de aprendizaje activo y reflexivo.
- Promover el desarrollo de habilidades sociales y de comunicación

Objetivos Específicos
- Iniciar el estudio del paradigma de la programación funcional.
- Ejercitar desarrollo de programas basados en el paradigma de la programación funcional.
- Iniciar el estudio y aplicación del paradigma de la programación orientada a objetos.
- Ejercitar desarrollo de programas basados en el paradigma de la programación orientada a objetos.
- Iniciar el estudio de la verificación y depuración dinámica de software usando técnicas de pruebas estructurales y funcionales, usando herramientas para verificar software previamente desarrollado usando herramientas de código abierto.

Durante el dictado de la asignatura se abordan los siguientes ejes transversales:
- Identificar, formular y resolver problemas de informática.
- Utilizar de manera efectiva las técnicas y herramientas de aplicación en la informática.
- Desempeñarse de manera efectiva en equipos de trabajo.
- Comunicarse con efectividad de forma escrita, oral y gráfica.
- Actuar con ética, responsabilidad profesional y compromiso social.
- Aprender en forma continua y autónoma.
- Actuar con espíritu emprendedor.

VI - Contenidos

Contenidos mínimos (OCD 1/23)
Paradigma de la programación orientada a objetos: Características generales de los lenguajes orientados a objetos. Tipos de datos. Estructuras de datos. Tipos de Datos Abstractos (TDA). Tipos de datos polimórficos. Resolución de problemas usando un lenguaje que responda al paradigma. Eficiencia, legibilidad y reusabilidad de software.
Paradigma de la Programación Declarativa Funcional: Características generales de los lenguajes funcionales. Tipos de datos. Estructuras de datos. Tipos de Datos Abstractos (TDA). Tipos de datos polimórficos. Resolución de problemas usando un lenguaje que responda al paradigma.
Introducción a la Verificación y Depuración dinámica de software.
Técnicas de caja negra y caja blanca. Prueba de software desarrollado en diferentes paradigmas de manera manual y/o usando herramientas de código abierto.
1. Programación Orientada a Objetos
Conceptos Generales del paradigma:
Descripción del paradigma de orientación a Objetos. Conceptos: clases, objetos, servicios/métodos, estado/atributos, mensajes, super, self, encapsulamiento. Herencia. Control de herencia. Tipos de herencia. Sobrecarga y Polimorfismo. Eficiencia, legibilidad y reusabilidad de software.

Lenguaje a utilizar: Java
Conceptos particulares del paradigma en el lenguaje Java:
Características generales del lenguaje. Tipos de datos simples, estructurados y abstractos. Estructuras de control. Clases, objetos, variables y métodos. Declaración, creación y destrucción de Objetos. Control de acceso. Herencia. Sobrecarga vs. Polimorfismo. Compatibilidad de objetos. Conversión de tipos (Down/Up Casting). Tipos de Datos Abstractos (TDA) y polimórficos. Enlace estático vs dinámico de métodos. Definición e implementación de Interfaces. Gestión de excepciones en java. Programación de Interfaces de Usuario Textuales (IUT/) y Gráficas (IGU).
2. Programación Funcional
Conceptos Generales del paradigma:
Características de los lenguajes funcionales. Estilo de programación libre de puntos (pointfree) vs atada a puntos (poitwise). Evaluación perezosa vs Evaluación impaciente. Concepto de función, definición y aplicación. Coincidencia de patrones (Pattern matching). Funciones recursivas. Funciones de orden superior, definición y aplicación. Currificación y aplicación parcial de funciones. Composición de funciones. Manejo de listas. Listas por comprensión.
Lenguaje a utilizar: Haskell.
Conceptos particulares del paradigma en el lenguaje Haskell:
Entorno de trabajo, definición de programas, uso del intérprete. Módulos. Notación de listas por comprensión. Operadores infijos y prefijos. Reglas de precedencia.Tipos de Datos Abstractos (TDA) vs. tipos de datos provistos por el lenguaje. Funciones predefinidas para manejo de listas y tuplas. Definición y uso de Funciones de orden superior. Expresiones Lambda. Definición y manipulación de funciones y tipos polimórficos. Análisis de funciones recursivas. Recursión primitiva y estructural. Recursión de cola vs recursión controlada, uso de acumuladores.
Análisis y uso de las funciones de folding predefinidas para listas: foldl, foldl1, foldr, foldr1
3.Verificación de Programas
Introducción a la Verificación y Depuración de Software. Pruebas de errores del software. Características y fases de las pruebas. Pruebas Estáticas vs Dinámicas. Estrategias de prueba. Técnicas de prueba: prueba estructural y prueba funcional. Prueba Unitaria. Técnicas de caja negra y caja blanca. Prueba de software desarrollado en diferentes paradigmas de manera manual y/o usando herramientas de código abierto.

VII - Plan de Trabajos Prácticos

Los trabajos prácticos a desarrollar en la asignatura comprenden:
PROGRAMACIÓN ORIENTADA A OBJETOS
Desarrollo un proyecto en Java codificado en un Entorno de Desarrollo Integrado (IDE a elección de cada alumno a partir de una lista de sugerencias de la Cátedra). En dicho proyecto se desarrollarán tareas de ejecución y análisis de módulos provistos por la Cátedra para el aprendizaje de conceptos del paradigma, implementación y análisis de diferentes tipos de datos abstractos y desarrollo de una interface Gráfica de Usuario(GUI) utilizando librerías de Java a elección de cada alumno (Swing, Javafx, etc.).
Entrega de 2 laboratorios de aprobación obligatoria. El primer laboratorio se desarrolla y entrega de manera individual por cada estudiante. El segundo corresponde al desarrollo de la GUI y se desarrolla en grupos de trabajo colaborativo (máximo 3 integrantes). La aprobación del último laboratorio implica la presentación de un video con la participación de los alumnos que lo desarrollaron explicando las características y decisiones de diseño de la GUI implementada.
PROGRAMACIÓN FUNCIONAL
En los prácticos se enfatizan los aspectos relacionados con la concepción de la programación como una actividad rigurosa y formalizada, esto es, la noción básica de un programa como objeto matemático sobre el cual es posible razonar con toda precisión y utilizando argumentos lógicos, algebraicos y matemáticos.
El contenido de la práctica puede dividirse en dos partes, la primera tiene como objetivo la adquisición de habilidades para el razonamiento y desarrollo de soluciones usando el lenguaje Haskell y la segunda tiene como objetivo la aplicación de las habilidades desarrolladas para la programación de un módulo que implemente un sistema de Gestión simplificado.
- Parte 1: Desarrollo de un práctico con ejercicios de análisis y uso de funciones primitivas y de orden superior, desarrollo de funciones recursivas con diferentes tipos de recursión (de cola y controlada), simulación de la recursión de cola con el uso de acumuladores, uso y análisis de funciones de orden superior sobre listas, funciones de plegado de listas (folding).
- Parte 2: Desarrollo de un laboratorio de entrega y aprobación obligatoria. El mismo consiste en tareas de análisis y desarrollo de funciones que implementen la funcionalidad de un modelo simplificado provisto por la Cátedra.
VALIDACIÓN Y VERIFICACIÓN DE PROGRAMAS:

Desarrollo de un trabajo de investigación de los temas vistos en teoría. En grupos de 2 o 3 integrantes, los estudiantes deberán desarrollar un documento escrito (artículo de divulgación científica) que cumpla con las características indicadas por la cátedra.
A continuación, se describen el abordaje y evaluación de los distintos ejes transversales trabajados en la asignatura:
- Identificar, formular y resolver problemas de informática.
Como se aborda: Mediante el desarrollo de trabajos prácticos, dirigidos por guías de estudio desarrolladas por la cátedra, disponibilización de material teórico obligatorio y opcional (en este caso se incluyen links a videos actualizados públicos en la web, revisados por la Cátedra), clases teórico-prácticas donde se resumen los contenidos esenciales, se analizan, discuten y consultan temas específicos de cada unidad. Resolución de prácticos de aula y de ejercicios propuestos para mejorar el desempeño en las evaluaciones sumativas.
Como se evalúa:
Evaluaciones formativas: participación en foros de discusión, desarrollo de test de evaluación de aprendizaje disponibles en el sitio de la materia.
Evaluaciones sumativas: entrega de laboratorios de desarrollo de código y evaluaciones parciales con sus respectivas recuperaciones.
- Utilizar de manera efectiva las técnicas y herramientas de aplicación en la informática.
Como se aborda: Cada estudiante debe codificar los ejercicios prácticos propuestos y los laboratorios solicitados para cada unidad utilizando artefactos de software de los paradigmas Funcional y Orientado a Objetos obligatorios, como los lenguajes de programación (Haskell y Java) y a elección en lo referente al ambiente de desarrollo integrado (IDE) y la plataforma de programación (Linux, OSX, Windows).
El software disponibilizado y recomendado es de uso Libre.
Se proponen diferentes ambientes de trabajo dependiendo del paradigma. En el caso de la programación declarativa funcional, se propone el uso de: interprete de Haskell (Glasgow Haskell Compiler Interprete- GHCI) y un editor de programas a elección (Notepad++, AquaMacs, etc.) o el uso de ambientes integrados (WinGHCI de la Plataforma Haskell, NetBeans, Visual Estudio Code, etc).
Como se evalúa: Mediante la supervisión del uso eficiente de las herramientas involucradas en las clases teórico-prácticas y en el análisis del código presentado en los laboratorios solicitados. La presentación obligatoria de Laboratorios.
- Desempeñarse de manera efectiva en equipos de trabajo.
Como se aborda: Mediante el desarrollo de un proyecto de trabajo en grupos de a lo sumo tres estudiantes.
Como se evalúa: Presentación de una aplicación Gráfica desarrollada por cada grupo, donde cada integrante debe realizar una defensa oral para evaluar su participación en el desarrollo de la misma.
Comunicarse con efectividad de forma escrita, oral y gráfica.
Como se aborda: Se solicita la participación obligatoria en foros de discusión (en los que deben colocar conclusiones personales y las fuentes consultadas), la redacción obligatoria de respuestas a cuestionarios en distintos laboratorios, el desarrollo de respuestas a preguntas de integración en las evaluaciones parciales, se solicita el desarrollo de un documento escrito producto de la investigación realizada por los estudiantes referente a un tema particular. Se solicita el desarrollo de un video explicativo de la aplicación Gráfica desarrollada.
Como se evalúa: Mediante la corrección de los contenidos escritos, orales y gráficos del material presentado en la materia.
- Actuar con ética, responsabilidad profesional y compromiso social.
Como se aborda: Mediante la explicitación de la forma de trabajo en materia. Esto es, se realiza una especie de contrato al iniciar el curso, donde se acuerda:
• como será el comportamiento del equipo docente de la materia, y cuál deberá ser el comportamiento de los estudiantes en los diferentes tipos de interacción (clases teórico-prácticas, parciales, test on-line, foros, etc.)
• compromiso de cumplimiento de actividades en tiempo y forma (docentes y estudiantes).
• Formas adecuadas de solicitudes de extensión de plazos fijados.
Como se evalúa: con la observancia del cumplimiento del acuerdo mencionado por parte de los estudiantes y el equipo docente de la materia.
- Aprender en forma continua y autónoma.
Como se aborda: con el desarrollo de diferentes actividades que aseguren la atención y aprendizaje continuo de los estudiantes. Presentación de trabajos prácticos obligatorios. Participación en foros de discusión. Disponibilización en el sitio de la materia de material visual actual, links a blogs de contenido relacionado con los diferentes temas de la materia, etc.
Como se evalúa: se realizan correcciones y devoluciones personalizadas de cada actividad propuesta. Tanto las actividades propuestas y las correcciones realizadas en cada caso son gestionadas en el sitio de la materia.
- Actuar con espíritu emprendedor.
Como se aborda: mediante la posibilidad de elección de diferentes herramientas y niveles de desarrollo en sus actividades

prácticas.
Promoviendo la participación en actividades extracurriculares tales como la postulación a desafíos, competencias, becas de intercambio, etc. Motivando conductas proactivas mediante la transferencia de experiencias profesionales propias y/o ajenas (docentes y estudiantes).
Como se evalúa: Con la evaluación de una presentación presencial y grabada de una aplicación Gráfica desarrollada grupalmente, donde cada estudiante muestra sus elecciones y las herramientas elegidas para el desarrollo de la misma. Con la observancia de cambios aptitudinales individuales y colectivos de los estudiantes.

VIII - Regimen de Aprobación

- Para regularizar la asignatura:
el alumno debe aprobar dos exámenes parciales o sus correspondientes recuperaciones, y presentar y aprobar en tiempo y forma los laboratorios solicitados por la cátedra.
Se debe asistir al 70% de las clases teórico/prácticas.
Para aprobar la materia deberá rendir un examen final.
Se tomarán dos recuperaciones para cada parcial, acorde a la Ord.32/14 CS.
- Para promocionar la asignatura:
el alumno debe cumplir con las condiciones de regularización y aprobar los exámenes parciales o sus correspondientes recuperaciones, presentar y aprobar en tiempo y forma los laboratorios solicitados por la Cátedra y la evaluación integradora con una nota superior o igual a 7.
Se debe asistir al 80% de las clases teórico/prácticas.
Se tomarán dos recuperaciones para cada parcial, acorde a la Ord.32/14 CS.
Régimen de estudiantes libres:
La asignatura NO admite examen libre.
El desarrollo, tanto de los prácticos de aula como de los laboratorios, se realiza de manera incremental permitiendo una evaluación continua de los estudiantes. Esta metodología imposibilita la correcta evaluación de esta parte de la asignatura en una instancia de examen final.

IX - Bibliografía Básica

[1] [1] - Guía de estudio del Lenguaje Haskell. Material provisto por la Cátedra.
[2] [2] - Learn You a Haskell for Great Good!.A Beginner's Guide by Miran Lipovaca. No Starch press. April 2011, 400 pp.
[3] ISBN-13: 978-1-59327283-8, ISBN-10: 978-1-59327283-8. Adaptación al español gratuita on-line en http://aprendehaskell.es/).
[4] [3] - Real World Haskell, Bryan O'Sullivan, Don Stewart, and John Goerzen. Paperback: 700 pages, O'Reilly, November 2008, English, ISBN-10: 0596514980, ISBN-13: 978-0596514983. Disponible on line gratuito en
[5] http://book.realworldhaskell.org/
[6] [4] - Thinking in Java (4th Edition),Bruce Eckel, Prentice Hall; 4 edition (February 20, 2006), ISBN-10: 0131872486, ISBN-13: 978-0131872486.
[7] [5] - Aprender a programar Java desde cero. Autor: Mario Rodríguez Rancel. 200 páginas; Editorial: aprenderaprogramar.com, 2012. ISBN: 978-84-939427-0-0.
[8] [6] - Jorgensen, Paul C.; “Software Testing, a Craftsman’s Approach”, CRC Press, 1995.
[9] [7] - Daniel Bolaños y otros; “Pruebas de software y JUnit”, Prentice-Hall, 2008.

X - Bibliografia Complementaria

[1] [1] - Haskell: The Craft of Functional Programming (2nd Edition), Simon Thompson, Addison-Wesley, ISBN 0201882957, 2011.
[2] [2] - Introduction to programming in Java. Hume, J.N. Patterson; Stephenson, Christine; Holt Software Assoc Inc. 2000. ISBN: 0921598394.
[3] [3] - Predicate Calculus and Program Semantics (Monographs in Computer Science), Edsger W. Dijkstra, Carel S. Scholten
[4] Springer; 1 edition (December 18, 1989), ISBN-10: 0387969578, ISBN-13: 978-0387969572
[5] [4] - Logic in Computer Science modelling and reasoning about systems, Michael Huth and Mark Ryan; 427 pages (2nd edition). Cambridge University Press (link) in paperback only: ISBN 0 521 54310X

XI - Resumen de Objetivos

Desarrollar habilidades para el desarrollo de software en el paradigma de la programación funcional.
Desarrollar habilidades para el desarrollo de software en el paradigma de la programación orientada a objetos.
Desarrollar habilidades para la aplicación de técnicas de verificación de software.

XII - Resumen del Programa

Contenidos mínimos (OCD 1/23)
El paradigma de la Programación Funcional(PF) Características generales de los lenguajes funcionales. Estudio de un lenguaje del paradigma.
El paradigma de la Programación Orientada a Objetos(POO): Descripción del paradigma de orientación a Objetos. Estudio de un Lenguaje del Paradigma.
Verificación de Programas(VP): Técnicas de caja negra y caja blanca. Prueba de software desarrollado en diferentes paradigmas.
Plan de Trabajos Prácticos
Se deben desarrollar y aprobar:
Laboratorios (según lo disponga la cátedra) de POO desarrollados en lenguaje Java.
Un laboratorio de PF desarrollado en lenguaje Haskell.
Se debe desarrollar un trabajo escrito de investigación respecto al tema Verificación de Programas.

XIII - Imprevistos

XIV - Otros

Las vías de comunicación con los estudiantes son las siguientes:
- Correo electrónico docente: albornoz@email.unsl.edu.ar
- Oficina 27 – 1° piso – 2° Bloque

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