Ministerio de Cultura y Educación **(Programa del año 2023)**
Universidad Nacional de San Luis
Facultad de Ciencias Físico Matemáticas y Naturales
Departamento: Informatica
Area: Area IV: Pr. y Met. de Des. del Soft.

I - Oferta Académica

Materia	Carrera	Plan	Año	Período
PROGRAMACION II	LIC.CS.COMP.	32/12	2023	1° cuatrimestre
PROGRAMACION II	PROF.CS.COMPUT.	02/16	2023	1° cuatrimestre

II - Equipo Docente

Docente	Función	Cargo	Dedicación
NECCO, CLAUDIA MONICA	Prof. Responsable	P.Asoc Exc	40 Hs
ALBORNOZ, MARIA CLAUDIA	Responsable de Práctico	JTP Exc	40 Hs
ZALDUA, ANALIA MAGDALENA	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**
4 Hs	Hs	3 Hs	2 Hs	9 Hs

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

Duración			
Desde	**Hasta**	**Cantidad de Semanas**	**Cantidad de Horas**
13/03/2023	24/06/2023	15	135

IV - Fundamentación

En este curso se inicia a los alumnos en dos paradigmas: prog. Orientada a Objetos y prog. 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 verificación de programas continuando con el estilo de demostración semántico funcional de Dijkstra, basado el la lógica axiomática de Hoare (introducido en materia anterior), usando alguna herramienta de software existente para realizar aserciones sobre programas simples, reforzando las bases para su posterior tratamiento formal tanto en el ámbito de la ingeniería del software como en el de teoría de la computación.

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 prog. funcional.
- Iniciar el estudio y aplicación del paradigma de la prog. orientada a objetos.
- Ejercitar desarrollo de programas basados en el paradigma de la prog. orientada a objetos.
-Continuar el estudio y aplicación de técnicas de verificación de software, en particular del estilo de demostración semántico functional de Dijkstra, basado en la lógica axiomática de Hoare, usando alguna herramienta para verificar software previamente desarrollado.

VI - Contenidos

Contenidos mínimos (OCD 32/12)
El paradigma de la Programación Orientada a Objetos: Descripción del paradigma de orientación a Objetos. Estudio de un Lenguaje del Paradigma.
El paradigma de la Programación Funcional: Características generales de los lenguajes funcionales. Estudio de un lenguaje del paradigma. Verificación de Programas: Introducción al estilo de demostración semántico funcional de Dijkstra.

1.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. Transparencia referencial. 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.
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 polimorficos. 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.

2. 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.
Lenguaje a utilizar: Java
Conceptos particulares del paradigma en el lenguaje Java:
Características generales del lenguaje. Tipos de datos básicos. Estructuras de control. Clases, objetos, variables y métodos. Declaración , creación y destrucción de Objetos. Control de acceso. Herencia. Polimorfismo. Compatibilidad de objetos. Conversión de tipos (Down/Up Casting). 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).

3.Verificación de Programas
Herramienta a utilizar: Key-Hoare Tool
Descripción y características generales de técnicas de verificación basadas en: casos de prueba, model-checking y en la utilización de métodos formales. Métodos formales: Lógica axiomática de Hoare y estilo semántico functional de Dijkstra. Transformadores de programas. Precondición más débil (weakest precondition WP). Determinismo y disyuntividad. Iteraciones. Aserciones, Invariantes.

VII - Plan de Trabajos Prácticos

Los trabajos prácticos a desarrollar en la asignatura comprenden:

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 de un sistema de Gestión.

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 5 laboratorios de aprobación obligatoria. Los primeros 4 laboratorios se entregan a partir de un proyecto desarrollado individualmente por cada alumno. El 5to. corresponde al desarrollo de la GUI y se desarrolla en grupos de trabajo colaborativo de 2 alumnos. 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.

DEMOSTRACIÓN DE ALGORITMOS
Ejecución de ejercicios de demostración de algoritmos, usando la herramienta Key-Hoare Tool o similar.

VIII - Regimen de Aprobación

- Para regularizar la asignatura:
el alumno debe aprobar dos exámenes parciales o sus correspondientes recuperaciones, y presentar y aprobar en forma y

tiempo los laboratorios solicitados por la cátedra.
Se debe asistir al 80% 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
El estudiante deberá desarrollar dos laboratorios, uno de programación funcional y otro de Programación orientada a objetos. Si los laboratorios son aprobados, se tomará un examen escrito de carácter práctico con ejercicios de integración sobre los contenidos de la asignatura.

IX - Bibliografía Básica

[1] Guía de estudio del Lenguaje Haskell. Material provisto por la Cátedra.
[2] Learn You a Haskell for Great Good!.A Beginner's Guide by Miran Lipovaca. No Starch press. April 2011, 400 pp. ISBN-13: 978-1-59327283-8, ISBN-10: 978-1-59327283-8. Adaptación al español gratuita on-line en http://aprendehaskell.es/).
[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 http://book.realworldhaskell.org/
[4] Thinking in Java (4th Edition),Bruce Eckel, Prentice Hall; 4 edition (February 20, 2006), ISBN-10: 0131872486, ISBN-13: 978-0131872486.
[5] Aprender a programar Java desde cero. Autor: Mario Rodríguez Rancel. 200 páginas; Editorial: aprenderaprogramar.com, 2012. ISBN: 978-84-939427-0-0.
[6] A Discipline of Programming, Edsger W. Dijkstra, Prentice Hall, Inc. (October 28, 1976), ISBN-10: 013215871X, ISBN-13: 978-0132158718
[7] A Hoare-Style Calculus with Explicit State Updates,Reiner Hahnle and Richard Bubel, "Course on Program Verification". Department of Computer Science, Chalmers University of Technology. Disponible en: https://www.researchgate.net/publication/241142337_A_Hoare-Style_Calculus_with_Explicit_State_Updates

X - Bibliografia Complementaria

[1] Haskell: The Craft of Functional Programming (2nd Edition), Simon Thompson, Addison-Wesley, ISBN 0201882957, 2011.
[2] Introduction to programming in Java. Hume, J.N. Patterson; Stephenson, Christine; Holt Software Assoc Inc. 2000. ISBN: 0921598394.
[3] Predicate Calculus and Program Semantics (Monographs in Computer Science), Edsger W. Dijkstra, Carel S. Scholten Springer; 1 edition (December 18, 1989), ISBN-10: 0387969578, ISBN-13: 978-0387969572
[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, en el estilo de demostración semántico funcional de Dijkstra, basado en la lógica axiomática de Hoare, usando alguna herramienta

XII - Resumen del Programa

Contenidos mínimos (OCD 32/12)
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): Introducción al estilo de demostración semántico funcional de Dijkstra.

Plan de Trabajos Prácticos
Se deben desarrollar y aprobar:
Un laboratorio de PF desarrollado en lenguaje Haskell
Cinco laboratorios de POO desarrollados en lenguaje Java
Se deben ejecutar ejercicios de demostración de algoritmos, usando la herramienta Key-Hoare Tool o similar.

Modalidad de Cursado:
La materia puede ser cursada de manera presencial para su regularización o promoción bajo la condición de alumno REGULAR, o puede ser aprobada mediante la aprobación de un examen en condición de alumno LIBRE.

XIII - Imprevistos

XIV - Otros