Universidad Nacional de General San Martín
Escuela de Ciencia y Tecnología
Campus Miguelete
Martín de Irigoyen 3100
San Martín
Buenos Aires - Argentina
www.unsam.edu.ar
EscuelaCyT@unsam.edu.ar

UNSAM
UNIVERSIDAD NACIONAL DE SAN MARTÍN
CAMPUS MIGUELETE
ESCUELA DE CIENCIA Y TECNOLOGÍA

PROGRAMA		
MATERIA	***PARADIGMAS DE PROGRAMACION***	
CARRERAS	***Tecnicatura Universitaria en Programación Informática***	
SEDE	***Miguelete***	
CARÁCTER DE LA MATERIA	***OBLIGATORIA***	
PERÍODO DE VIGENCIA	***Segundo cuatrimestre 2010***	***DURACION: cuatrimestral***
DOCENTES	***Nicolás Passerini - Fernando Dodino***	
MATERIAS CORRELATIVAS	***Cursada de Algoritmos 3*** ***Final de Algoritmos 2***	
CARGA HORARIA	***Clases Teórico-prácticas: 3 horas semanales*** ***Laboratorios -1 hora semanal***	***Total de horas semanales: 4*** ***Total de horas de la materia: 64***

OBJETIVOS:

Que el alumno…

- Comprenda los fundamentos de los paradigmas básicos que son utilizados por los lenguajes de programación actuales.
- Conozca el modelo formal o semiformal subyacente de cada paradigma y la forma en que el mismo es incorporado correctamente en un lenguaje de programación.
- Aplique los diferentes paradigmas en la resolución de problemas.

CONTENIDOS MÍNIMOS

- Concepto de Paradigmas de Programación. Paradigmas Fundamentales.
- Paradigma Funcional.
    - Cálculo Lambda.
    - Lenguajes de Programación Funcional.
- Paradigma Lógico.
    - Lógica de Predicados de Primer Orden y Formas Restringidas.
    - Lenguajes de Programación Lógica.
- Revisión del Paradigma Orientado a Objetos.

    - Polimorfismo en lenguajes débil y fuertemente tipados en el modelo de objetos.
- Conceptos transversales a los diferentes paradigmas
    - Polimorfismo
    - Tipado
    - Binding
    - Orden Superior
    - Declaratividad
    - Efecto colateral
    - Representación de entidades y conjuntos
    - Recursividad

FUNDAMENTACIÓN

Este espacio procura que el alumno explore nuevas visiones para resolver problemas, tomando en cuenta ideas que la industria del software está incorporando en forma paulatina.

Entendemos que para ello es necesario que el alumno

1. conozca, experimente y comprenda las principales abstracciones de cada paradigma.
2. sea capaz de comprender qué paradigma se adapta a cada contexto en particular.
3. sea capaz de integrar y relacionar los conceptos comunes a todos los paradigmas analizados
4. reconozca la validez de aplicar un concepto en un determinado contexto

METODOLOGÍA DE EVALUACIÓN

Se dará en dos instancias, que permitirán evaluar:

- **Parciales Prácticos**, utilizar los conceptos adquiridos para diseñar una solución a un problema concreto y su capacidad para comunicar la solución propuesta aplicando los conceptos teóricos más salientes de cada paradigma.
- **Trabajos Prácticos**, resolver un problema poniendo en práctica los conceptos y herramientas aprendidos.

METODOLOGÍA DIDÁCTICA

Los contenidos del espacio se organizan en clases teóricas y prácticas. Los contenidos teóricos se introducirán siguiendo una metodología constructivista, precedidos por ejemplos que justifiquen y motiven cada nuevo concepto permitiendo que el alumno deduzca la necesidad y la utilidad de cada elemento de la teoría por sus propios medios. Una vez introducido el concepto ayudado por la intuición y la deducción, se procederá a la formalización y

generalización del mismo en diferentes contextos, comparando las características obtenidas en diferentes soluciones que utilicen o no los diferentes conceptos aprendidos.

Durante la clase práctica se realizarán ejercicios de diseño y programación, tanto en papel como en máquina, en forma individual o por grupos, con el objetivo de llevar a la práctica los conceptos aprendidos en la teoría, vinculando fuertemente ambos territorios.

También se prevé destinar tiempo de las clases prácticas para el seguimiento de los trabajos prácticos, respondiendo consultas de diseño y destrabando problemas que pudieran aparecer con las diferentes tecnologías utilizadas.

Una vez conocidos los paradigmas individualmente, en la parte final de la materia podemos estudiar a los paradigmas en su conjunto. Desde el punto de vista teórico se estudiará la aparición de conceptos similares en los distintos paradigmas y se compararán los beneficios obtenidos en uno y otro caso. Finalmente se mostrarán situaciones en las cuales se combinan los distintos paradigmas en un mismo programa, ya sea utilizando diferentes lenguajes de programación o utilizando lenguajes híbridos (que combinen elementos de más de un paradigma).

PROGRAMA ANALÍTICO

UNIDAD 1: Paradigmas de Programación

Concepto de paradigma de programación. Necesidad de la existencia de diferentes paradigmas de programación. Diferencia entre lenguaje y paradigma de programación. Concepto de tipo: representación de los tipos en los diferentes lenguajes de programación. Importancia del concepto de tipo en relación a la implementación de sistemas complejos y cambiantes. Ubicación de los mecanismos de control de flujo en un programa. Abstracción y modularización: definición y mecanismos de implementación. Comparación entre los diferentes paradigmas de programación.

UNIDAD 2: Paradigma de Objetos

Repaso de los conceptos principales del paradigma de objetos. Implementación del concepto de polimorfismo en lenguajes débil y fuertemente tipados.

Lenguaje asociado: se presentarán ejemplos en diferentes lenguajes: Smalltalk, javascript, Ruby, entre otros.

UNIDAD 3: Paradigma Funcional

Concepto de función. Definición de tipo y valor. Definición de funciones. Funciones definidas por ramas. Pattern matching. Inferencia de tipos. Funciones recursivas. Prueba por inducción.

Manejo de listas. Listas por comprensión. Funciones de orden superior. Currificación y aplicación parcial de funciones. Evaluación diferida y listas infinitas. Composición de funciones. Sistemas de tipos. Polimorfismo y tipos genéricos. Tuplas. Expresiones lambda. Tipos abstractos de datos. Definición de clases de usuario. Introducción al concepto de mónadas.

Lenguaje asociado: Haskell. Entorno de trabajo, definición de programas, uso del intérprete. Notación bidimensional. Módulos. Notación de listas [n..m]. Notación de listas por comprensión. Operadores infijos y prefijos. Reglas de precedencia. Prelude de Haskell. Funciones incorporadas en el prelude para manejo de listas, de tuplas, de funciones de orden superior, de mónadas.

UNIDAD 4: Paradigma Lógico

Fundamentación lógica. Predicados. Razonamientos y silogismos. Relaciones, hechos y reglas. Consultas. Tipos de consultas. Motor de inferencia, ubicación del control en un programa lógico. Diferencia entre una función y una relación. Unificación. Múltiples resultados. Inversibilidad de cláusulas. Aritmética, evaluación de expresiones aritméticas. Negación. Listas. Predicados recursivos. Predicados generadores. Pattern Matching. Uso y definición de Predicados de orden superior. Functores.

Lenguaje asociado: Prolog. Entorno de trabajo, manejo de archivos. Realización de consultas. Ayuda. Trace y debug. Limitaciones de inversiblidad: generación de valores.

UNIDAD 5: Integración de conceptos entre paradigmas

Concepto de programa: definiciones generales y específicas. Efecto colateral o efecto secundario. Concepto de variable. Sistema de tipos. Comparación de los diferentes esquemas de chequeo de tipos. Declaratividad: importancia de la separación del control de flujo de la lógica del dominio a modelar. Orden superior: concepto e implicancias en el desarrollo de programas. Utilización de las variantes del polimorfismo en los diferentes paradigmas. Diseño en cada uno de los paradigmas.

BIBLIOGRAFÍA

Lectura general sobre Paradigmas de Programación

- *Programming Languages Concepts and Paradigms*, David Watt, Prentice Hall.
- *Concepts, Techniques, and Models of Computer Programming*, Peter Van Roy and Seif Haridi, The MIT Press

Bibliografía específica del Paradigma de Objetos

- *Notas de clase,* Nicolás Passerini y Carlos Lombardi, disponible en www.pdep.com.ar

http://groups.google.com/group/archivos-pdep/web/apunte-objetos-1-4.doc

- *Smalltalk, Objects and Design*, Chamond Liu
- *Designing Object-Oriented Software*, Wirfs- Brock, R. y otros, Prentice Hall

Bibliografía específica sobre lenguajes del Paradigma de Objetos

Smalltalk

- *Smalltalk Best Practice Patterns,* Kent Beck
- *Smalltalk 80- The Language ,* Adele Goldberg and David Robson

Ruby

- *Refactoring: Ruby Edition,* Jay Fields, Shane Harvie, Martin Fowler, Kent Beck

Javascript

- *Object-Oriented JavaScript: Create scalable, reusable high-quality JavaScript applications and libraries*, Stoyan Stefanov

Bibliografía específica del Paradigma Funcional

- *Notas de clase*, Fernando Dodino
- Richard A. Bird y Philip Wadler, Introduction to Functional Programming, Prentice Hall.
- Introducción al lenguaje Haskell, José E. Labra G., Universidad de Oviedo

Bibliografía específica sobre lenguajes del Paradigma Funcional

Haskell

- *The Craft of Functional Programming*, Simon Thompson, Editorial Addison-Wesley, 1999

Los siguientes tutoriales, disponibles en el sitio www.haskell.org

- A Gentle Introduction to Haskell , Paul Hudak, John Peterson and Joseph H.Fasel
- Learn You a Haskell for Great Good! , Miran Lipovača
- Yet Another Haskell Tutorial, Hal Daumé III
- Real World Haskell , Bryan O'Sullivan, Don Stewart y John Goerzen, O'Reilly Media

Bibliografía específica del Paradigma Lógico

- *Notas de clase*, Fernando Dodino
- *Logic Programming and Knowledge Representation,* Chitta Baral and Michael Gelfond, University of Texas, Logic Programming and Knowledge Representation

Bibliografía específica sobre lenguajes del Paradigma Lógico

Prolog

- *Programming in Prolog: Using the ISO Standard*, William F.Clocksin & Christopher S.Mellish, Editorial Springer
- *Prolog*, Giannesini, Kanoui, Pasero y Van Caneghem, Addison, Wesley Iberoamericana.
- *The Arity/ Prolog Language Reference Manual*, Arity Corporation.