CARTA AL ESTUDIANTE

Código:	EIF400
Nombre:	Paradigmas de Programación
Requisitos:	EIF206 Programación 3
Naturaleza:	Curso teórico/práctico
Área disciplinaria:	Ingeniería de Software
Nivel:	II Ciclo del III Nivel
Ciclo lectivo:	II Ciclo, 2014
Profesores:	Carlos Loría Sáenz Maykol Guzmán
Coordinador:	Oldemar Rodríguez Rojas

Créditos	Horas semanales	Horas presenciales (pueden variar según los contenidos específicos)		Horas de estudio independiente
		Teoría	Práctica	
4	11	3(2)	1(2)	7

DESCRIPCIÓN

Al elaborar un modelo para resolver un problema mediante programación, existen diferentes enfoques sobre cómo se debe realizar la abstracción de los diferentes elementos de dicho problema. Así, dependiendo de la situación que se desea modelar, cada uno de estos distintos enfoques o paradigmas de programación tiene ventajas y desventajas, que facilitan o entorpecen la construcción de un programa. En el curso se estudian de manera comparativa los diferentes paradigmas de programación existentes, logrando de esta manera conocer los criterios más importantes para la selección de un lenguaje determinado. El curso busca también complementar el conocimiento de los estudiantes de ingeniería informática en los paradigmas y técnicas de modelado que no se han estudiado en los cursos regulares de programación.

OBJETIVO GENERAL

Ofrecer a los estudiantes un panorama general sobre los principales paradigmas de programación, los conceptos teóricos que los fundamentan y las técnicas fundamentales utilizadas en cada uno.

OBJETIVOS ESPECÍFICOS

Al terminar el curso, se pretende que el estudiante haya logrado adquirir el conocimiento y las habilidades necesarias para:

1. Esbozar la historia de los lenguajes de programación y comprender la manera en que estos han evolucionado para adaptarse a las necesidades de desarrollo de software.
2. Reconocer e identificar los tres principales paradigmas de programación existentes (según el modelo teórico que los sustenta) y sus características principales.
3. Identificar las características de cada lenguaje de programación, su implementación y ambiente de ejecución.
4. Comprender los problemas que existen para la traducción de lenguajes y las principales técnicas utilizadas en compiladores e interpretadores.
5. Identificar las particularidades de un lenguaje de programación que pueden afectar de una u otra manera la construcción de un programa.
6. Aprovechar las características de los lenguajes y las técnicas de programación orientada a objetos.
7. Establecer criterios y diseñar proyectos de software que permitan a un grupo de trabajo el uso de diferentes herramientas, lenguajes y técnicas.
8. Emplear técnicas funcionales en la construcción de aplicaciones generales.
9. Comprender la especificación formal de condiciones de salida para la comprobación de resultados de un programa.

CONTENIDOS

Se propone que el desarrollo de cada una de las cuatro partes en que se divide el curso tenga una duración aproximada de cuatro semanas y media (equivalente a 9 lecciones). Dependiendo del desarrollo del curso, es posible que se reasignen algunas lecciones para poder cubrir adecuadamente cada uno de los temas propuestos.

Parte 1 – Características generales de los lenguajes de programación

1. *Introducción a los lenguajes de programación (Objetivos 1 ,2, 4)*
    a. Historia de los lenguajes de programación.
2. *Traducción y compilación*
    a. Autómatas de estado Finito (FSA)
    b. Gramáticas y definición formal de lenguajes de programación.
    c. Analizadores sintácticos y generación de código.
3. *Lenguajes de programación y arquitectura (Objetivos 2, 4)*
    a. Tipos de datos y mecanismos de abstracción.
    b. Métodos de encapsulamiento.
    c. Paradigmas de programación.
        i. Modelos computacionales teóricos
        ii. Programación imperativa
        iii. Programación declarativa
            1. Programación funcional
            2. Programación lógica
        iv. Otros paradigmas de programación

Parte 2 – Programación funcional

4. Programación Funcional (Objetivos 3, 5, 7)

a. Introducción al cálculo lambda (λ-calculus)
b. Recursividad y funciones recursivas primitivas.
c. Recursividad simple, recursividad lineal y de cola.
d. Limitaciones de la programación funcional.
e. El problema del reconocimiento de patrones
f. Expresiones regulares y Evaluación de expresiones

Parte 3 – Programación lógica

5. Programación lógica (Objetivos 2,3,4,7)

a. Programación lógica.
b. Principios teóricos de la programación lógica.
c. Lógica cuantificada de primer orden.
d. El algoritmo de unificación y el método de resolución.
e. Declaraciones recursivas.

Parte 4 – Programación imperativa y Programación Orientada a Objetos

6. Programación imperativa (Objetivos 2, 3, 5)

a. Las estructuras básicas y la programación estructurada por control.
b. Comparación entre la programación imperativa secuencial y la programación algorítmica (estructurada).

7. Programación guiada por eventos (Objetivos 2, 3, 6)

a. Programación guiada por eventos (*event-driven programming*) y programación guiada por flujo (*flow-driven programming*).
b. Modelo básico de la programación guiada por eventos.
c. El problema del manejo de la estructura de control básica (ciclo de atención de eventos).
d. Relación entre la implementación de las interfaces gráficas de usuario (GUIs) y la arquitectura de una aplicación de software.
e. Serialización de interfaces de usuario.

8. Programación orientada a objetos (Objetivos 2, 6)

a. Conceptos fundamentales y propósito de la POO (Programación Orientada a Objetos).
b. Principios empíricos de la POO.
c. Ventajas y desventajas de la POO. Consecuencias del empleo de técnicas de POO en el desarrollo de sistemas de información.

METODOLOGÍA

El curso contará con la exposición magistral por parte del profesor de cada uno de los contenidos descritos. Además, se harán sesiones de resolución de problemas y prácticas de laboratorio para enfrentar al estudiante de manera directa con las principales dificultades y técnicas utilizadas en programación. De esta manera también se logra que el estudiante aplique los conceptos expuestos en situaciones específicas.

De la misma manera habrán proyectos de programación que requieren más tiempo del que se dispone durante las lecciones, donde los estudiantes resuelven en grupo ejercicios de dificultad media o alta, para conocer, estudiar y resolver problemas representativos.

También se realizarán investigaciones y se harán exposiciones sobre diferentes temas, cuyo estudio no puede hacerse dentro de las limitaciones de tiempo de la clase, pero se consideran importantes para conseguir cumplir cabalmente el objetivo general del curso.

EVALUACIÓN

Los exámenes en el curso buscan medir y evaluar la compresión de cada estudiante del material estudiado durante el curso y del trabajo realizado en los proyectos. Los **exámenes** deben realizarse y entregarse **individualmente**, aunque la realización de los **proyectos** pueden completarse en **grupos de dos personas como máximo**.

La suma de los porcentajes obtenidos por el estudiante en cada aspecto señalado determina su nota de aprovechamiento. Si ésta es superior o igual a 70%, el estudiante aprueba el curso. De lo contrario, el estudiante pierde el curso. Por ser un curso que incluye el desarrollo de proyectos prácticos, no existe la posibilidad de realizar un examen extraordinario.

Descripción	**Porcentaje**
Primer examen parcial	**30%**
Segundo examen parcial	**30%**
Quices y tareas Se realizarán exámenes cortos (quices) para evaluar el progreso del grupo. Los exámenes cortos se realizarán según el cronograma. También se asignarán tareas cortas que serán entregadas por el profesor con por lo menos una semana de anticipación.	**15%**
Trabajo de investigación Se hará un trabajo de investigación sobre algún tema relacionado con los objetivos del curso. Los estudiantes plantearán diferentes temas o problemas que podrían desarrollar. El profesor hará una valoración del tema seleccionado para lograr que su nivel de complejidad, aporte a los temas y objetivos del curso y desarrollo sea el adecuado. Se hará en grupos de a lo más dos personas. El grupo de trabajo para la investigación estará compuesto preferiblemente por los mismos integrantes que para la realización de los proyectos. Un aspecto importante del curso es que los estudiantes continúen desarrollando su capacidad de trabajo en grupo y mejoren sus habilidades para comunicar el resultado de sus investigaciones y desarrollos al resto de sus compañeras y compañeros.	**10%**
Proyecto(s) Los proyectos servirán para evaluar aspectos prácticos concretos de	**15%**

los temas estudiados en el curso. Pueden realizarse en grupos de a lo más dos personas. Los proyectos asignados serán preferiblemente programados, pero pueden tratar también de algún tipo de desarrollo teórico. **Los grupos de trabajo se formarán al inicio del curso, y permanecerán integrados de la misma manera para cada trabajo grupal asignado.**	

Por la naturaleza del curso, es inevitable que los contenidos desarrollados sean acumulativos para los exámenes y trabajos prácticos. Es decir, aunque en una evaluación (sea individual o grupal) se deba profundizar en uno o más temas específicos, esto no implica que no se puedan incluir temas anteriormente evaluados.

Al ser un curso cuya evaluación contempla aspectos prácticos, como los laboratorios y proyectos programados, no hay examen extraordinario. La suma de los porcentajes obtenidos por el estudiante en los rubros anteriores determina su nota de aprovechamiento (NA). El curso se aprueba con una NA igual o superior al 70%.

CRONOGRAMA

Punto de evaluación	**Fecha estimada**
Primer examen parcial	Semana 8 (En horas de clase)
Segundo examen parcial	Semana 16 (En horas de clase)
Exámenes cortos y tareas	Durante todo el curso, los exámenes cortos y la revisión de las tareas serán en horas de clase.
Trabajo de investigación	Durante todo el curso
Proyecto(s) El enunciado de los proyectos será entregado por cada profesor. Los estudiantes entregarán los proyectos el segundo día de clase de la semana correspondiente.	La revisión de los proyectos serán en horas de clase.

BIBLIOGRAFÍA

Material de referencia y consulta:

Aho A., Sethi Ravi, D. Ullman Jeffrey (2006). *Compilers: Principles, Techniques, and Tools.* Prentice Hall; 2nd edition.

Armstrong, Joe (2007). *Programming Erlang. Software for a Concurrent Wold.* Pragmatic Bookshelf; 1st Edition.

Bramer, M.A. (2010). *Logic Programming with Prolog.* Springer-Verlag.

Budd, Timothy (2001). *Introduction to Object-Oriented Programming.* Addison Wesley; 3rd edition

Cesarini, Francesco, Thompson, Simon (2009). *Erlang Programming.* O'Reilly, USA.

Clocksin, W. F. (1997). *Clause And Effect.* Springer-Verlag.

Clocksin, W. F., Mellish, C. S. (2003). *Programming in Prolog: Using the ISO Standard.* Springer-Verlag; 5th edition.

Hankin, Chris (2004). *An Introduction to Lambda Calculi for Computer Scientists.* King's College Publications. Londres.

Helo Guzmán, José E. (2005). *Introducción a la programación con Scheme.* Editorial Tecnológica, ITCR. Segunda edición, ISBN 9977-66-176-6.

Pratt, Terrence W., Zelkowitz, Marvin V. (2000). *Programming Languages: Design and Implementation.* Prentice Hall, 4th Edition.

Watt, David A. (1993). *Programming Language Concepts and Paradigms.* Prentice Hall, USA.

Además de los textos anteriores, se utilizarán los manuales respectivos de cada uno de los lenguajes estudiados, y bibliografía o material adicional que se anotará oportunamente durante el curso.

Software:

Para la elaboración de los proyectos se utilizará software provisto de manera oportuna por el profesor, que también podrá ser descargado de Internet sin costo ni necesidad de licencias de uso.

Las direcciones correspondientes para consultar documentación o descargar software son:

http://www.mingw.org/	gcc (para Windows)
http://oracle.com/	Java SDK
http://www.netbeans.org/	**NetBeans IDE**
http://racket-lang.org/	**Racket**
http://www.erlang.org/	**Erlang**
http://www.swi-prolog.org/	SWI Prolog

ESPECIFICACIONES GENERALES.

Es requisito indispensable para ganar el curso la presentación de todos los proyectos.

Los proyectos deben ser entregados según las indicaciones de cada profesor. Si el estudiante no presenta los trabajos en la fecha indicada en el cronograma del curso, queda a criterio del profesor el recibir los trabajos y establecer el porcentaje de la nota que será rebajado.

En caso de corroborarse algún fraude en la aplicación de cualquiera de las evaluaciones, la Escuela de Informática aplicará las sanciones establecidas en el reglamento de la Universidad Nacional.

El horario disponible para la atención a estudiantes será programado y comunicado por cada profesor. La asistencia oportuna y comprometida del estudiante le permitirá obtener del profesor en este espacio: orientación en los trabajos asignados durante el curso, evacuación de dudas en los temas estudiados y la articulación conjunta de ideas para el desarrollo de los trabajos. Este horario no descarta la posibilidad de que los estudiantes planteen dudas y soliciten orientación por otros medios, como el correo electrónico o el aula virtual.