UNIVERSIDAD NACIONAL DE CÓRDOBA
Facultad de Ciencias Exactas, Físicas y Naturales
República Argentina

Programa de:

Informática Avanzada

Código: 7438

Carrera: *Ingeniería Electrónica e Ingeniería en Computación* Escuela: *Escuela de Ingeniería Electrónica y Computación* Departamento: *Computación*.	Plan: *281-05* Carga Horaria: *72* Semestre: *Quinto* Carácter: *Obligatoria* Bloque: *Tec. Básicas*	Puntos: *3* Hs. Semanales: *4,5* Año: *Tercero*

Objetivos: Al terminar el curso el alumno:

- *Comprenderá los principios de la Informática y de Programación a Orientada a Objetos desde una perspectiva de la Ingeniería de Software.*
- *Diseñará la solución de problemas de ingeniería a partir de su conceptualización en estructuras de datos concebidas como clases de objetos que interactúan entre si para lograr un comportamiento complejo.*
- *Desarrollará y configurará aplicaciones de software reutilizando estructuras de datos flexibles a partir de bibliotecas, para resolver problemas que manejen la interfaz hombre-máquina.*
- *Utilizará herramientas de software para analizar, y desarrollar software, gestionando adecuadamente los errores de implementación y las sucesivas versiones*

Programa Sintético:

Parte I: Fundam. de la Orientación a Objetos

1. *Paradigmas y modelos de cómputo.*
2. *Objetos, clases, campos y métodos.*
3. *Interacción y agrupamiento de objetos.*
4. *Bibliotecas y documentación de software*
5. *Desarrollo y depuración de programas.*

Parte II: Estructuras de aplicación

6. *Diseño de clases.*
7. *Herencia. Polimorfismo.*
8. *Clases de datos abstractos, interfaces.*
9. *Interfaces de usuarios basadas en eventos.*
10. *Gestión de errores y excepciones.*
11. *Diseño de aplicaciones.*

Programa Analítico: de foja *6* a foja *7*.

Programa Combinado de Examen (si corresponde): de foja 9 a foja 9 .

Bibliografía: de foja *4* a foja *4*.

Correlativas Obligatorias: Informática.
Correlativas Aconsejadas: Métodos Numéricos.

Rige: *2008*

Aprobado HCD, Res.: Fecha:	Modificado / Anulado / Sust. HCD Res.: Fecha:

El Secretario Académico de la Facultad de Ciencias Exactas, Físicas y Naturales (UNC) certifica que el programa está aprobado por el (los) número(s) y fecha(s) que anteceden. Córdoba, / / .

Carece de validez sin la certificación de la Secretaría Académica:

PROGRAMA ANALITICO

LINEAMIENTOS GENERALES

Durante varios años desde la década de 1970 se concibieron diferentes estilos de programación que trataban de superar la visión de las Arquitecturas de Computadoras basadas en la visión de la Máquina de Von Neuman que se pueden percibir aún en los lenguajes de programación que tratan de abstraer los detalles de las máquinas físicas. Una primera clasificación nos permite distinguir los lenguajes simbólicos de los procedurales o imperativos. Según sea el tratamiento de las expresiones pueden seguirse dos líneas de clasificación en control de flujo, procedimientos, módulos, clases y objetos o en funciones, programación funcional o programación en lógica y finalmente convergiendo ambas líneas en la concurrencia de procesos que interatúan entre si.

La aparición durante la década de 1960 de la Programación Orientada a Objetos se inicia con el sistema de simulación denominado Simula y posteriormente recibe un gran impulso teórico a sus fundamentos con el grupo de investigación del Xerox Parc Place y su desarrollo del lenguaje Smalltalk durante la década de 1970. No obstante la pureza del paradigma implementado en Smalltalk se ha convertido en un problema para acceder al mismo ya que se debe abandonar completamente el paradigma procedural y también la sintaxis de los lenguajes descendientes del Algol como Pascal y particularmente la familia del C, constituida por el C mismo, C++, C# y el Java. Un caso particular es el de Eiffel más cercano a las ideas del Pascal y el Algol pero implementa el paradigma de objetos en forma pura, lo que lo lleva ser de difícil distribución masiva, no obstante muchas de sus características han sido implementadas actualmente en Java y C#.

Otro aspecto que ha colaborado en la difusión del paradigma orientado a objetos han sido los Sistemas Operativos con interfaces de usuario gráficas en las cuales la metáfora de los objetos y los eventos permite elaborar una interacción con el usuario natural y flexible.

Simultáneamente con el crecimiento de la Programación Orientada a Objetos se da el Análisis y Diseño Orientados a Objetos hasta desembocar en el Proceso Unificado de Desarrollo y el Lenguaje de Modelado (UML) que permite actualmente disponer de una Metodología que guíe el proceso de diseño e implementación.

El proceso de abstracción del diseño orientado a objetos se han descubierto, al igual que en los proyectos de arquitectura e ingeniería, que hay patrones de diseño a partir de los cuales construir el software sin necesidad de tener que reinventarlo y además se ha afirmado la construcción de software a partir de componentes reusables durante las décadas de 1990 y 2000.

También ha tenido lugar el reconocimiento de que el desarrollo de software puede someterse a los principios de la Ingeniería, dando lugar a la Ingeniería de Software como disciplina dentro de la Informática (Ciencia de la Computación) como así mismo como carreras universitarias de grado y postgrado de ingeniería.

A este contexto de la disciplina debe agregarse el contexto de las materias de Informática dentro de las carreras de Ingeniería de la Facultad de Ciencias Exactas, Físicas y Naturales, que a partir de la decisión de unificar un programa básico para todas las ingenierías, se llegó a un programa con contenidos excesivos al incluir los paradigmas procedimental y orientado a objetos, basados en los constructor del lenguaje C/C++ a los que se les agregaba el lenguaje Matalab/Octave para el desarrollo de prototipos de corte numérico y con facilidades para graficación.

En el contexto de las Ingenierías como las de Civil, Aeronáutica, Mecánica, Mecánica Electricista, Industrial y Química, una segunda materia avanzada en el desarrollo de software requeriría poner énfasis en Matalab/Octave y versiones modernas de Fortran 2003 que implementen el paradigma de orientación a objetos. No obstante esta asignatura nos se describe en términos de ningún lenguaje específico ya que se trata de una asignatura de Ciencias Básicas y no de Tecnologías Basicas y por lo tanto también se la puede dictar para dichas carreras.

El diseño de la asignatura se ha orientado a las carreras de Ingeniería Electrónica y en Computación que las requieren como partes estructurales de dichas carreras y se la podría caracterizar como de Tecnologías Básicas en ambas. Se ha seleccionado para ello un enfoque desde la Ingeniería de Software, cuyos conceptos básicos en la primera son insuficientes para encarar los Trabajos Finales y requieren un estudio de adaptación previa. Para Ingeniería en Computación será inestimable disponer de una formación previa en Programación Orientada a Objetos e introductoria a la Ingeniería de Software.

En cuanto a la pedagogía de la asignatura se ha seguido el concepto de introducir los Objetos Primero y para ello el alumno deberá acompañar el estudio teórico del diseño orientado a objetos con la práctica de la elaboración de un proyecto integral que justifique plenamente por su complejidad el paradigma de anális, diseño e implementación adoptados. También se pone énfasis en aceptar que el desarrollo industrial de software requiere metodologías de Ingeniería de Software y de herramientas de desarrollo que amplifiquen la eficiencia y la productividad. Esto no significa que se abordará en forma completa ningún lenguaje orientado a objetos en particular ya que se enfocará en los criterios de diseño.

METODOLOGIA DE ENSEÑANZA

Las etapas de construcción y elaboración de conocimientos son sustentadas mediante la exposición dialogada como estrategia didáctica y el empleo de proyección de diapositivas, filminas, pizarrón y proyector multimedia como materiales didácticos. Todos los materiales de estudio, incluyendo sistema de consultas, preguntas frecuentes, e-mail, evaluaciones, etc., se disponen en el sistema informático de aprendizaje del Departamento de Computación (Laboratorio de Enseñanza Virtual – LEV. http://lev.efn.uncor.edu)

La fase de ejercitación y aplicación de los contenidos de la signatura, se fundamenta tanto en el desarrollo teórico como en el práctico del presente curso. Se realizan dos tipos diferenciados de actividades en coordinación con el desarrollo de la autonomía de aprendizaje, consistentes en la solución de problemas acotados y en la elaboración de un proyecto informático integrador realizado en equipo. En estas instancias el trabajo individual y grupal, permite la conformación de ideas y el establecimiento de relaciones entre el conocimiento adquirido y situaciones nuevas planteadas desde otras problemáticas de la misma disciplina.

El dictado se realizará en 16 clases de 4hs 30min (reloj) consistentes en la presentación teórica de los temas por parte del docente, las que no podrán superar 1:30min en cada sesión.

La presentación de temas prácticos se realizará preferentemente, en el caso de disponer de equipos de computación, en el marco de tareas de laboratorio, previamente asignadas por el docente coincidentes con el tema teórico previo, asumiendo el docente el rol de tutor y mediante evaluaciones formativas en cada clase.

En el caso de no disponer del laboratorio se realizarán ejercitaciones y simulaciones de escritorio que permitan poner de manifiesto los objetivos de la asignatura.

El proceso de elaboración del proyecto integrador será seguido mediante entregas parciales pautadas en el LEV, así como la devolución de las evaluaciones parciales. El proyecto será defendido mediante una presentación pública para todo el curso.

Programación de actividades y bibliografía recomendada

Clase	Tema	Capítulo*
1	Unidad 1, 2	1
2	Unidad 3	2
3	Unidad 4	3
4	Unidad 5	4
5	Unidad 6	5
6	Unidad 7	6
7	1era evaluación parcial	
8	Unidad 8	7

Clase	Tema	Capítulo*
9	Unidad 9	8
10	Unidad 10	9
11	Unidad 11	10
12	Unidad 12	11
13	Unidad 13	12
14	Unidad 14	13
15	2da evaluación parcial	
16	Parcial de recuperación	

***** Corresponde con el Capítulo del material bibliográfico básico:

Barnes, David y Kölling Michael (2007). Objects First with Java A Practical Introduction using BlueJ.(3a. edición) Prentice Hall, Pearson Education.

EVALUACION

Evaluaciones Formativas

Las actividades de Laboratorio consistirán en la puesta en funcionamiento de los Ejercicios y Problemas de especificación de programas que acompañan al enunciado de los temas conceptuales y se considerarán como de realización necesaria para la acreditación del porcentaje de asistencia total. El estudiante pondrá a disposición de los profesores el trabajo en equipo, mediante el uso del LEV (Laboratorio de Educación Virtual).

1era Evaluación Parcial de Acreditación

Tiene por objeto acreditar que el alumno ha alcanzado, individualmente, las siguientes metas de aprendizaje, en relación a las unidades 1, 2, 3, 4 y 5 del programa analítico:

- Deberá interpretar el enunciado formal de un problema de ciencias básicas de la ingeniería o de información.
- Diseñará la solución de problemas de ingeniería a partir de su conceptualización en estructuras de datos concebidas como clases de objetos que interactúan entre si para lograr un comportamiento complejo.

Características generales:

- La evaluación se realizará durante el horario habitual de clases, pudiendo disponerse del tiempo asignado a las exposiciones teóricas y de laboratorio.
- Consistirá en la solución de ejercicios de diseño y especificación clases de objetos en el lenguaje asignado y con las herramientas de diseño apropiadas.
- La calificación será de 0 a 10 y el peso relativo del 30% del total.

2da Evaluación Parcial de Acreditación

Tiene por objeto acreditar que el alumno a alcanzado, individualmente, las siguientes metas de aprendizaje, en relación a las unidades 6, 7, 8, 9, 10, 11, y 13 del programa analítico y considerando los conceptos necesarios de las unidades anteriores:

- Deberá interpretar el enunciado formal de un problema de ciencias básicas de la ingeniería o de información.
- Desarrollará y configurará aplicaciones de software reutilizando estructuras de datos flexibles a partir de bibliotecas, para resolver problemas que manejen la interfaz hombre-máquina.

Características generales:

La evaluación se realizará durante el horario habitual de clases, pudiendo disponerse del tiempo asignado a las exposiciones téoricas y de laboratorio.
Consistirá en la solución de ejercicios de diseño y especificación de aplicaciones completas en el lenguaje asignado.
La calificación sera de 0 a 10 y el peso relativo del 30% del total.

Condición de regularidad

Para alcanzar la condición de ALUMNO REGULAR se deberán cumplir los siguientes requisitos excluyentes:

- Asistir al 80% de las clases teóricas y de laboratorio.
- Aprobar un examen parcial con nota cuatro (4) o superior.
- Aprobar el Proyecto de Programación.

Recuperación de parciales:

En el caso de no alcanzar la nota de 4 en ninguno de los dos parciales se deberá aprobar un único examen parcial de recuperación cuya nota deberá ser 4 o superior. A los fines de la nota final se reemplazará la nota original por la del parcial de recuperación.

Régimen de promoción

Aprobación de la materia:

Para lograr la promoción se deberán alcanzar los siguientes objetivos excluyentes:

- Asistir al 80% de las clases teóricas y de laboratorio.
- Aprobar los dos exámenes parciales con nota cuatro (4) o superior.
- Obtener una calificación final con nota cuatro (4) o superior

Recuperación de parciales:

En el caso de no alcanzar la nota de 4 en uno de los dos parciales se deberá aprobar un único examen parcial de recuperación cuya nota deberá ser 4 o superior. A los fines de la nota final se reemplazará la nota original por la del parcial de recuperación.

Calificación final:

La calificación es el promedio ponderado de las diferentes evaluaciones y su valor numérico se establece como:

Nota Final = Nota 1er Parcial * 0.30 + Nota 2do Parcial * 0.30 + Nota Proy. * 0.4

Este valor se redondeará al entero más próximo.

CONTENIDOS TEMATICOS

PARTE I: FUNDAMENTOS DE LA ORIENTACIÓN A OBJETOS

Unidad 1. Paradigmas y modelos de cómputo
Paradigmas de programación. Lógicas. Funciones y cálculo lambda. Sistemas de producciones. Autómatas.

Unidad 2. Objetos y clases

Objetos y clases. Llamado de métodos y sus parámetros. Tipos de datos. Instancia de objetos y estado. Interacción básica entre objetos. Valores de retorno en métodos. Objetos como parámetros de métodos.

Unidad 3. Declaración y definición de clases

Campos, constructores, y métodos (acceso, edición). Pasado de datos vía parámetros. Sentencias de asignación y condicional. Impresión desde métodos. Variables locales y parámetros.

Unidad 4. Interacción entre objetos

Abstracción, modularización de software. Diagramas de clases y de objetos. Tipos primitivos y tipos de objetos. Creación de objetos y constructores múltiples. Llamado interno y externo de métodos. La autorreferencia a un objeto. Uso de un depurador.

Unidad 5. Agrupamiento de objetos

Agrupamiento de objetos en colecciones de tamaño flexible. Elementos básicos de una biblioteca de clases. Clases genéricas. Enumeración y gestión de colecciones. Ciclos while, iteradores. Colecciones fijas de objetos, arreglos, ciclos for.

Unidad 6. Bibliotecas y documentación de programas

Documentación de bibliotecas de clases estándar. Interfaces o implementación de métodos. Lectura y redacción de documentación de clases parametrizadas. Importación y paquetes de código. Información y ocultamiento de campos y métodos. Variables de clase y constantes.

Unidad 7. Prueba, depuración y mantenimiento de software

Prueba y depuración. Depuración de unidades. Inspectores. Pruebas positivas y negativas. Automatización de las pruebas. Pruebas de regresión. Escenarios y registro de pruebas. Manuales. Uso de depuradores.

Unidad 8. Diseño de clases

Introducción al acoplamiento y la cohesión. Duplicación y extensión de código. Uso de encapsulado. Diseño basado en responsabilidades. Rrefabricación. Guías de diseño.

PARTE II: ESTRUCTURAS DE APLICACION

Unidad 9. Herencia de clases

Jerarquías de herencia de campos y métodos de clases. Derechos de acceso a la herencia. Inicialización de instancias con herencia. Subtipos y subclases. Subtipos y asignación. Subtipos y pasado de parámetros. Variables polimórficas. Casting. Clases predefinidas como jerarquías.

Unidad 10. Polimorfismo
Tipos estáticos y dinámicos. Sobrecarga de métodos. Búsqueda dinámica de métodos. Llamado a la superclase en los métodos. Polimorfismo de métodos. Métodos objeto. Acceso protegido.

Unidad 11. Acoplamiento flexible de funcionalidad

Clases abstractas. Superclases. Métodos abstractos. Interfaces y herencia múltiple. Interfaces como tipos. Interfaces como especificaciones.

Unidad 12. Construcción de interfaces gráficas de usuario

Construcción de Interfaces Gráficas de Usuario. Componentes y despliegue de la interfaz. Manejo de eventos. Bibliotecas de manejo abstracto de ventanas e Interfaces Gráficas de Usuario. Clases de procesamiento de imágenes.

Unidad 13. Gestión de errores y excepciones

Programación defensiva. Chequeo de errores, excepciones, manejo flexible de errores. Reporte de errores. Uso de aserciones. Herramientas de prueba automatizadas. Serialización de objetos.

Unidad 14. Diseño de aplicaciones

Análisis y diseño. El método de verbos/sustantivos. Descubrimiento de clases. Escenarios. Colaboración entre clases. Diseño de interfaz de clases y usuarios. Documentación. Cooperación. Modelos de desarrollo de software. Patrones básicos de diseño.

1. LISTADO DE ACTIVIDADES PRACTICAS Y/O DE LABORATORIO

Actividades Prácticas

1.- Actividades de Laboratorio

El alumno realizará actividades de programación en el Laboratorio de Computación que se corresponden con los ejercicios propuestos como actividades de práctica.

1. Objetos, clases, campos y métodos. Ambiente de desarrollo básico.
2. Interacción y agrupamiento de objetos.
3. Bibliotecas y documentación de software. Acceso a bibliotecas.
4. Desarrollo y depuración de programas. Herramientas de Depuración.
5. Diseño de clases.
6. Herencia. Polimorfismo.
7. Tipos y clases de datos abstractos, interfaces.
8. Interfaces de usuarios basadas en eventos. Desarrollo de interfaces.
9. Gestión de errores y excepciones.
10. Diseño de aplicaciones.

2.- Actividades de Proyecto y Diseño

Tiene por objeto acreditar que el alumno ha adquirido las siguientes habilidades y técnicas, relacionadas preferentemente a la totalidad de los contenidos de la asignatura:

- Aplicar la informática a un problema de ciencias básicas, de ingeniería o de sistemas de información, desde su formulación simbólico-matemática o de información hasta su implementación en un lenguaje informático.
- Adquirir la habilidad para la depuración de algoritmos y programas mediante una técnica basada en principios lógicos.
- Experimentar con diferentes criterios de diseño.
- Capacidad para el trabajo en equipo en la planificación y ejecución de un proyecto informático.

Características generales:

- El proyecto consistirá en el desarrollo de los algoritmos matemáticos y/o de información que den solución a un problema de ciencias o ingeniería.
- Se implementará la solución en el lenguaje definido y se probarán diferentes criterios de diseño y se presentarán todas las versiones de los archivos de código fuente. El diseño debe permitir definirlo como orientado a objetos.
- La aplicación resultante deberá poderse ejecutar en un ambiente de Windows o de Linux sin errores sintácticos ni lógicos y la interacción con el usuario estará basada en una interfaz gráfica controlada por eventos.
- Se documentará la presentación mediante una monografía sobre el tema, los criterios adoptados al respecto del diseño, como tarjetas CRC, escenarios, diagramas de clases de colaboración, de asociación y de acción, eficiencia algorítmica, interfaces con el usuario, etc.
- Constará de un manual de usuario o ayuda en línea
- Los grupos estarán constituidos por 4 alumnos como máximo.
- La presentación se realizará durante las clases de laboratorio correspondientes al último mes de clase.
- La calificación será de 0 a 10 y el peso relativo del 40% del total.

2. DISTRIBUCION DE LA CARGA HORARIA

ACTIVIDAD	HORAS
TEÓRICA	18
FORMACIÓN PRACTICA:	
o FORMACIÓN EXPERIMENTAL	18
o RESOLUCIÓN DE PROBLEMAS	27
o ACTIVIDADES DE PROYECTO Y DISEÑO	9
o PPS	
	72

DEDICADAS POR EL ALUMNO FUERA DE CLASE

ACTIVIDAD		HORAS
PREPARACION TEÓRICA		18
PREPARACION PRACTICA		
	o EXPERIMENTAL DE LABORATORIO	0
	o EXPERIMENTAL DE CAMPO	0
	o RESOLUCIÓN DE PROBLEMAS	27
	o PROYECTO Y DISEÑO	27
	TOTAL DE LA CARGA HORARIA	72

3. BIBLIOGRAFIA

Básica

- Barnes, David y Kölling Michael (2007). *Objects First with Java A Practical Introduction using BlueJ* (3a. Edición). Prentice Hall, Pearson Education.
- Llamas Bello, César (2004). *Introducción a la Informática. Modelos de Cómputo*. International Thomson editores Spain.

Recomendada

- Deitel, H. M., Deitel, P. J. (2004). *Cómo programa en JAVA* (5ta. Edición). Pearson Educación.
- Eckel, Bruce (2006). *Thinking in JAVA* (4ta edición).Prentice Hall
- Campione, Mary y otros (2006).*The Java Tutorial: A Short Course on the Basics* (4ta Edicion) . Prentice Hall
- Walrath, Katy y otrros (2004).*The JFC Swing Tutorial: A Guide to Constructing GUIs* (2nd Edition). Prentice Hall.