UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

ORIENTACIÓN A OBJETOS 1

Año 2014

Carrera/ Plan:
Licenciatura en Informática
Plan 2003-07 / Plan 2012
Licenciatura en Sistemas
Plan 2003-07 / Plan 2012
Analista Programador Universitario
Plan 2007

Año: *2°*

Régimen de Cursada: *Semestral*

Carácter: Obligatoria

Correlativas: Algoritmos, Datos y Programas

Coordinador: *Gustavo Rossi*
Profesor: *Roxana Giandini, Alicia Díaz*

Hs. semanales: 6 hs.

FUNDAMENTACIÓN

El paradigma de Orientación a Objetos es un paradigma de programación que se caracteriza por brindar herramientas de programación que promueven la reusbilidad, mantenibilidad, modificabilidad y fiabilidad de los programas; cualidades que deben ser rectoras del ejercicio profesional de un buen programador.

OBJETIVOS GENERALES:

Presentar formalmente el paradigma de objetos, sus características, ventajas y aplicaciones dentro del desarrollo de sistemas de software. Desarrollar prácticas concretas con lenguajes Orientados a Objetos. Establecer metodologías de análisis y diseño Orientados a Objetos.

CONTENIDOS MINIMOS:

- Objetos.
- Clases e instancias.
- Encapsulamiento.
- Jerarquías de clase.
- Herencia. Polimorfismo.
- Lenguajes y aplicaciones.

PROGRAMA ANALÍTICO

Unidad 1. La crisis del software. Problemas de las técnicas tradicionales (procedurales). Resolución de problemas complejos. El problema de la extensibilidad, el reuso y el mantenimiento.

Unidad 2. Conceptos básicos: Tipos Abstractos de Datos. Encapsulamiento. Information hiding. Objetos y Programa. Comportamiento de un Objeto. Mensaje y Método. Clasificación: Clases e Instancias. Instanciación. Jerarquías de Clases. Relación *isA* Generalización/Especialización. Herencia, Herencia Simple. Clases Abstractas. Hacia mayor genericidad de código: polimorfismo y binding dinámico.

Unidad 3. Relaciones entre Objetos. Relación de conocimiento. Relación `isPartOf`. Conocimiento versus composición.

Unidad 4. Lenguajes orientados a objetos: variantes. El lenguaje Smalltalk. Tipos de Mensajes. Variables de instancia. PseudoVariables: `self` y `super`. Método `new`. Biblioteca de clases, jerarquías predefinidas: clase Magnitude y su protocolo.

Unidad 5. Estructuras de Control: Clases `Boolean`, `False` y `True`. Métodos: `or:`, `and:` y `not`. Definición de bloques de código. Clase `Context`. Métodos: `value` y `value:`. Métodos `ifTrue:`, `ifFalse:`, `ifTrue: ifFalse:`, `whileTrue:`, `whileFalse:`.

Unidad 6. Estructuras de datos como Objetos. Objetos contenedores. Colecciones de Objetos. Clase `Collection` y sus subclases `Array`, `OrderedCollection`, `Set`, `Dictionary` y `SortedCollection`. Protocolo estándar. Iteradores: `to: do:`, `to: by: do:`, `timesRepeat:`. El iterador `do:`. Otros iteradores: `select:`, `detect:`, `reject:`, `collect:`, `inject: to:`.

Unidad 7. Introducción al lenguaje de Modelado Unificado (Unified Modeling Language). Diagramas de UML. Diagramas de Estructura Estática: Diagramas de Clases. Diagramas Dinámicos ó de Comportamiento: Diagramas de Interacción (Diagramas de Secuencia y Diagramas de Colaboración), Diagramas de Casos de Uso.

Unidad 8. Diseños complejos: uso de `self` y `super` combinados. Herencia versus composición. Doble dispatching.

METODOLOGÍA DE ENSEÑANZA

Expectativas de logro

Presentar formalmente el paradigma de objetos, sus características, ventajas y aplicaciones dentro del desarrollo de sistemas de software. Desarrollar prácticas concretas con lenguajes orientados a Objetos. Establecer metodologías de análisis y diseño orientados a objetos.

Existen tres objetivos particulares:

Escribir Programas Orientados a Objetos

Este objetivo implica:

- adquirir los conocimientos teóricos básicos de la Programación Orientada a Objetos: Tipos Abstractos de Datos. Encapsulamiento. Information hiding. Objetos y Programa Orientado a Objetos. Comportamiento de un Objeto. Mensaje y Método. Clasificación: Clases e Instancias. Instanciación. Jerarquías de Clases. Relación *isA*. Generalización/Especialización. Herencia, Herencia Simple. Clases Abstractas. Hacia mayor genericidad de código: polimorfismo y binding dinámico.
- experimentar con los conceptos teóricos en situaciones prácticas donde manifiesten las ventajas del paradigma orientado a objetos:
- utilizar un lenguaje orientado a objetos que permita codificar programas orientados a objetos. El aprendizaje de un lenguaje debe incluir la definición d clases, variables, Instanciación de objetos, mecanismos de herencia soportado por el lenguaje; biblioteca de clases. Se sugieren como lenguaje Smalltalk, Ruby o Java.
- familiarización con un ambiente de desarrollo orientado a objetos, como puede ser VisualWorks en el caso del lenguaje Smalltalk

Manejo Adecuado de una Notación

Es necesario contar con alguna notación que facilite la comunicación, documentación y desarrollo de un diseño orientado a objetos. Se introducirá también el uso de un lenguaje de modelado gráfico orientado a objetos (UML), que le permitirá construir diagramas especificando distintos aspectos de un sistema.

Esta notación también debe acompañar el procesos de desarrollo de sistemas orientado a objeto como es el proceso de desarrollo unificado basado en UML (RUP).

Dentro de este objetivo también se encuentra la familiarización con un ambiente que soporte el diseño de diagramas UML e incluso el proceso de desarrollo. Se recomienda el uso de ambientes del estilo de Rational Rose.

Procedimientos didácticos

En función de los objetivos planteados se propone organizar el dictado de la materia en clases teórico-practicas organizadas de la siguiente manera.

Clases Teóricas:

Las clases teóricas están destinadas a presentar los conceptos teóricos del programa de la materia. Los conceptos teóricos deben ser presentados a través de su motivación, su definición, relación e interacción con los demás conceptos. Estas actividades deben ser fuertemente soportadas por el uso de ejemplos concretos.

Las clases teóricas deben ser presenciales. El docente a cargo debe presentar el tema del día a través de la exposición oral del mismo, promoviendo la participación de los alumnos. La participación del los alumnos se logra a través de la discusión de situaciones concretas de aplicación de los conceptos teóricos en cuestión. Es responsabilidad del docente tener la habilidad de manejar dichas discusiones de manera que se planteen pros y contras de los distintos enfoque expuestos por los alumnos.

Clase practicas

Las clases prácticas deben ser dedicadas a aplicar los conceptos teóricos impartidos en las clases prácticas. Las mismas deben ser guiadas por un trabajo práctico. Cada trabajo practico debe identificar una temática y un conjunto de objetivos teóricos-prácticos a lograr con las ejercitaciones planteadas.

El desarrollo de la clase práctica debe contar con una explicación del trabajo práctico por el auxiliar docente a cargo, donde se le indiquen al alumno los objetivos de la práctica y los conceptos teóricos que se pretenden aplicar, mas un conjunto de guías para a resolución de los problemas planteados. Luego, los alumnos desarrollan la práctica llevando a cabo cada ejercitación y contando permanentemente con la posibilidad de trabajar en conjunto con un auxiliar docente que guíe su trabajo y evacue sus dudas.

Los ejercicios de los trabajos prácticos serán diseñados de manera que los mismo impliquen el diseño de un programa, su implementación en un lenguaje OO y su testeo a través de usar tecnicas de testeo de unidad. Para el diseño del programa se usarán los recursos del lenguaje de modelado UML, para la implementación se sugiere trabajar con el lenguaje Smalltalk y el ambiente VisualWorks. Para el testeo, cada practica será acompañada por un test de unidad.

Cada trabajo práctico debe concluir con la entrega para su visado de un ejercicio de manera que el alumno reciba de parte de los docentes comentarios que permitan retroalimentar su evolución en el proceso de aprendizaje.

De acuerdo a la temática desarrollada por cada trabajo práctico las clases prácticas se pueden desarrollar en máquina o no. Los prácticos en máquina tienen como objetivo desarrollar las habilidades de programación en un ambiente de desarrollo asistida por el auxiliar docente. Las prácticas que no son en máquina apuntan a desarrollar diseños los cuales son supervisados por el auxiliar docente.

EVALUACIÓN

Los alumnos deben aprobar los trabajos prácticos previo a someterse a un examen final para la aprobación de la materia.

Para la aprobación de los trabajos prácticos debería haber una evolución al final del curso donde se evaluarán los conceptos teóricos-prácticos enunciados en los contenidos de la materia.

El parcial podría ser en modalidad escrita, en máquina o ambas dependiendo de la disponibilidad de laboratorio y/o docentes. La prueba parcial deberían contar con al menos una posibilidad de recuperación.

Con respecto a la aprobación de la materia los alumnos pueden elegir entre dos modalidades de evaluación: rendir un examen final teórico-práctico globalizador o realizar un proyecto final. El proyecto consiste en el desarrollo de un sistema de software usando la tecnología de objetos a través del cual se evalúan los conocimientos teórico-prácticos adquiridos. El trabajo final debe ser defendido por el alumno en una exposición oral.

BIBLIOGRAFÍA OBLIGATORIA

THE OBJECT-ORIENTED THOUGHT PROCESS, Matt Weisfeld. Third Edition, Pearson Education, Addison Wesley. ISBN-13: 978-0-672-33016-2

INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING, AN (3rd Edition), Thimoty Budd,Addison Wesley; 3 edition (2001), ISBN-10: 0201760312

JOY OF SMALLTALK. IVAN TOMEK, HTTP://PLATO.ACADIAU.CA/COURSES/COMP/TOMEK/JOS.HTM

SMALLTALK BY EXAMPLE. ALEX SHARP, MCGRAW HILL; (1997), ISBN: 0079130364

UML GOTA A GOT. FOWLER MARTIN, SCOTT KENDALL, ADDISON-WESLEY IBEROA. EDICIÓN 1999 ISBN 9684443641

BIBLIOGRAFÍA COMPLEMENTARIA

DESIGNING OBJECT-ORIENTED SOFTWARE. Rebecca Wirfs-Brock, Brian Wilkerson (Contributor), Lauren Wiener Prentice Hall PTR; (January 1991), ISBN 0136298257

ANALISIS Y DISEÑO ORIENTADO A OBJETOS CON APLICACIONES. BOOCH GRADY, ADDISON-WESLEY IBEROA, Edición 1996, ISBN 9684443528

SMALLTALK: AN INTRODUCTION TO APPLICATION DEVELOPMENT USING VISUALWORKS. Trevor Hopkins, Bernard Horan, Prentice Hall, ISBN: 0133183874

SMALLTALK WITH STYLE. Suzanne Skublics, Edward J. Klimas, David A. Thomas, John Pugh (Foreword). Pearson Education POD; 1 edition (May 21, 2002), ISBN: 0131655493

SMALLTALK BEST PRACTICE PATTERNS. Kent Beck, Prentice Hall PTR; 1st edition (October 3, 1996) ISBN: 013476904X

EL LENGUAJE UNIFICADO DE MODELADO . BOOCH GRADY, JACOBSON IVAR , RUMBAUGH JAMES, ADDISON-WESLEY IBEROA, Edición 2000, ISBN 8478290281

CRONOGRAMA DE CLASES Y EVALUACIONES

	Contenidos/Actividades	Evaluaciones previstas
Semana 1	Introducción a la POO: • objeto • mensajes • método • Semanas	
Semana 2	Introducción a la POO: • jerarquía de Semanas • herencia	
Semana 3	Introducción a Smalltalk: • sintaxis de sentencias, • tipo de mensajes • Separadores de sentencias: . y ; • Orden de precedencia	
Semana 4	Introducción a Smalltalk: • Polimorfismo • Binding Dinámico	
Semana 5	Variables. • Tipos de variables excepto variables de Semana uso de setters y getters Instaciacion básico: new PseudoVariables: `self`, `super`, `nil`, `true`, `false`. • Ejemplo: self --: o *`trasferir: unMonto a: otraCuenta`* implementado en la Semana `CajaAhorro` Implementación del mensaje + de la Semana *Point:* *ifTrue: if False:* y *timesRepeat:* en forma sintáctica.	
Semana 6	*Introducción a UML.* • *Diagramas de Secuencia.* o *ejemplo con el mensaje* *`transferir: a:`* o Biblioteca de Clases en Smalltalk: jerarquías de	

	Clases. *Object*: `printString, =, = =, copy` *Magnitude*: Interger, Carácter, Time, Date, Point). • Protocolos básico: operadores lógicos, *between: and:* Diseño e implementación del `between: and:` y el `timesRepeat:` en la subjerarquía de `Magnitude` *Instanciación e Inicialización:* • `Rectangle new origin: corner` • `Rectangle origin: corner:`	
Semana 7	Biblioteca de Clase en Smalltalk: • `BlockClosure:` o *value* , `value:` y `value: value:` • `Boolean:` o la jerarquía e implemntacion del *and:*, `ifTrue:` y el *ifTrue: ifFalse*: • `whileTrue:` o qué es la condición? Donde esta implementado? Porqué? Cómo se implementa?	
Semana 8	self y super • *Ejercicio con self y super* • Super: Redifinicion del new Ejercicio Integrador: • mantener el registro de movimientos de una caja de ahorro. • Discutir la multiplicidad de la asociación entre cuenta y movimiento (registro de movimientos) Ejemplo: transacción con comprobante. *UML Básico*: Diagrama de Secuencia *Crear Objeto* (Revisar diagrama de secuencia para mostrar el new)	
Semana 9	Smalltalk: • Introducción a *Collection.* • Saben que el registro de movimientos es un objetos que contiene	

	otros objetos y entiende el *add:* • *SubsSemanas de Collection* o *Bag* o *Set* o *Array* o *OrderedCollection* o *SortedCollection* o *Dictionary* Smalltalk: `Collection` - Iteradores	
Semana 10	Diseño: Manejo de Collections con polimorfismo: ejemplo	
Semana 11	Herencia vs. Composición Ejemplos de herencia con uso de *self*: algo así como el template pattern. Polimorfismo	
Semana 12	Diseño	
Semana 13	Diseño	
Semana 14	double dispatching	
Semana 15	Repaso	
Semana 16		1° evaluación
Semana 17		2° evaluación
Semana 18		
Semana 19		3° evaluación

Contacto de la cátedra (mail, página, plataforma virtual de gestión de cursos):

alicia.diaz@lifia.info.unlp.edu.ar
gustavo.rossi@lifia.info.unlp.edu.ar
roxana.giandini@lifia.info.unlp.edu.ar

Firmas del/los profesores responsables: