FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

PROGRAMA DE CURSO

Código	Nombre
CC3001	Metodologías de Diseño y Programación
Nombre en Inglés	
Design and programming methodologies	

SCT	Unidades Docentes	Horas de Cátedra	Horas Docencia Auxiliar	Horas de Trabajo Personal
6	10	3	1.5	4.5

Requisitos	Carácter del Curso
CC1001, CC3001(S)	Obligatorio para: Licenciatura en Ciencias mención Computación

Resultados de Aprendizaje

Este curso entrega los fundamentos necesarios para desarrollar software orientado a objetos fácil de entender, extender y mantener en el tiempo. En particular, enseña a diseñar y programar buenos objetos, a usar la herencia sólo cuando ésta provee ventajas reales, a integrar objetos para resolver un problema complejo, a diseñar y resolver problemas usando patrones de diseño, a evaluar diseños usando métricas y a enfrentar desarrollo de software de pequeña y mediana complejidad usando metodologías estándares.

Al final de este curso el alumno debe ser capaz de:

- entender los conceptos más importantes de las etapas de análisis, diseño y programación orientada a objetos y aplicarlos correctamente en el desarrollo de software de pequeña y mediana complejidad
- diseñar y programar objetos usando “buenos” objetos
- usar en concepto de herencia de manera apropiada
- reconocer y aplicar patrones de diseño
- programar en distintos lenguajes que soportan orientación a objetos tales como java y c++
- conocer y aplicar métricas para identificar problemas de diseño en programas orientados a objetos

Metodología Docente	Evaluación General
Clases de cátedra, clases auxiliares, trabajo individual y en grupo. Parcialmente se usa la metodología de aprendizaje basado en problemas.	2 controles 5 evaluaciones usando la metodología de aprendizaje basada en problemas (Se elimina 1). Equivale al control 3. Tareas individuales computacionales. Examen NF = 0.6 NC + 0.4 NT

Unidades Temáticas

Número	Nombre de la Unidad	Duración en Semanas
1	Introducción	1

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Revisión del programa del curso • Introducción a la complejidad en el desarrollo de software • Metodologías para enfrentar la complejidad en el desarrollo de software	Entender los problemas existentes en el desarrollo de software en general Conocer algunas metodologías usadas para el desarrollo de software	[1]

Número	Nombre de la Unidad	Duración en Semanas
2	Clases y objetos	2

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Clases y objetos. • Objetos: estado, comportamiento, identidad. • Cómo definir buenos objetos? • Diseño de buenos tipos de datos abstractos • Contratos • De tipos de datos abstractos a clases. • Ejemplos en java y c++ • Diseño de una buena clase	Programar usando clases y objetos Dominar los conceptos básicos de orientación a objetos Aprender a diseñar e implementar buenos objetos Entender código escrito java y en c++ Programar correctamente en java o en c++	[1], [3],[8], [9],[11],[12]

Número	Nombre de la Unidad	Duración en Semanas

3	Herencia y subtipos	3,5

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Motivando el uso de herencia • Polimorfismo y enlace dinámico. • Tipos y subtipos • Relaciones de subtipos entre: conjuntos, tipos estructurados y funciones • Contraste entre herencia y subtipos • Tipos de herencia • Contratos y herencia	Programar usando herencia Dominar los conceptos tipos y subtipos y su relación con la herencia Definir contratos usando herencia Aprender a diseñar e implementar buenos objetos usando herencia	[1], [3],[8], [9],[11],[13]

Número	Nombre de la Unidad	Duración en Semanas
4	Diseño de software orientado objetos	4

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Cómo encontrar objetos? • Cómo definir si una funcionalidad debe ser un método o un objeto? • Relaciones entre clases • Diagramas de clases (UML) • Cuando usar la relación de composicion? • Cuando usar la relación de herencia • Patrones de diseño: adaptador, iterador, observador, puente, singleton y fábrica abstracta, entre otros. • Evaluación de Diseños: métricas	Diseñar y programar usando buenas prácticas sistemas de software de medianos Identificar relaciones entre clases y construir diagramas de clases apropiados en UML Reconocer patrones de diseño Aplicar adecuadamente patrones de diseño en el desarrollo de una aplicación Conocer las distintas métricas existentes para la evaluación de software orientado a objetos, aplicarlas a un programa y analizar losresultados	[4],[10]

5	Validación de software orientado a objetos	1,5

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Métodos de testing • Testing y herencia • Niveles de testing • Testing el comportamiento de objetos • Verifición de consistencia en tiempo de ejecución • Validación estática	Aprender técnicas para hacer debugging de programas orientados a objetos Usar un debugger	[3],[7]

6	Proceso de desarrollo de software orientado a objetos	3

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Proceso de desarrollo unificado (RUP): Guiado por los casos de uso, centrado en la arquitectura, iterativo e incremental. • Casos de uso • Diagramas análisis, de secuencia e interacción de objetos • Aspectos básicos de interfaces a usuario • Enfrentando un problema de mediana envergadura.	Aprender a enfrentar el desarrollo de sofware orientado a objetos desde su etapa de análisis. Especificar casos de uso Crear diagramas de análisis y secuencia desarrollar software orientado a objetos construyendo cada una de sus etapas	[2],[7]

Bibliografía

[1] Bertrand Meyer. Object Oriented Software Construction. Prentice Hall. 1997. Second Edition.
[2] Ivar Jacobson, Grady Booch, James Rumbaugh. The unified software development process. Addison Wesley, 2000.
[3] A Eliëns. Principles of object oriented software development. Addison Wesley, 1995.
[4] Setrag Khoshafian, RazmikAbnous. Object orientation: concepts, languages, databases and user interfaces. John Wiley&Sons Inc. 1990.
[5] Erich Gamma, Richard Helm, Ralph Johnson, John vlissides. Design Patterns: Elements of reusable object oriented software. Addison Wesley. 1995.
[6] Cay Horstmann.Object-Oriented Design & Patterns John Wiley & Sons, Inc., 2004.
[7] Timothy C. Lethbridge, Robert Laganiere. Object oriented software engineering. McGraw-hill Education. 2001
[8] Bjarne Stroustrup. What is object oriented programming?
IEEE Software. 1988.

[9] Bertrand Meyer. Applying design by contract. Computer. 1990
[10] Daniel Halbert y Patrick O'Brien. Using types and inheritance
in object oriented languages. European conference on object-oriented programming on ECOOP '87. 1987.
[11] Developing software for the user interface. The SEI series in software Engineering. Adisson Wesley. 1991.
[12] Bjarne Stroustrup. The {C}++ Programming Language}. Addison-Wesley, 2002. Edicion especial.
[13] Java
http://sunsite.dcc.uchile.cl/SunSITE/java/docs/tut-java122-sun/index.html

Vigencia desde:	Agosto 2010
Elaborado por:	Nancy Hitschfeld Kahler