ORIENTACIÓN A OBJETOS II

Año 2017

Carrera/ Plan:
Licenciatura en Informática
Plan 2003-07/Plan 2012/Plan 2015
Licenciatura en Sistemas
Plan 2003-07/Plan 2012/Plan 2015
Analista Programador Universitario
Plan 2007/Plan 2015
Año: 3°
Régimen de cursada: *Semestral*
Carácter: Obligatoria
Correlativas:
Orientación a Objetos I
Taller de lecto-comprensión y traducción de Ingles
Coordinador: *Gustavo Rossi*
Profesor: *Alejandra Garrido*
Alejandro Fernández
Arturo Zambrano
JTP: *Leandro Antonelli*
Diego Torres
Hs. semanales: 6 hs.

FUNDAMENTACIÓN

OBJETIVOS GENERALES:

Profundizar los temas desarrollados por el alumno en Orientación a Objetos 1 e introducir conceptos fundamentales en la construcción de arquitecturas de software modulares, extensibles y reusables, a través de conceptos fundamentales como son: patrones de diseño, refactoring hacia patrones y frameworks orientados a objetos. Se profundizará también en el uso de un lenguaje de modelado gráfico orientado a objetos (UML), que le permitirá construir diagramas especificando distintos aspectos de un sistema. Los trabajos prácticos se realizarán usando el lenguaje de modelado y el lenguaje de implementación Smalltalk, que resulta el más apropiado de acuerdo a estos objetivos.

CONTENIDOS MINIMOS:

- Diseño Orientado a Objetos
- Patrones de diseño
- Construcción de aplicaciones con frameworks orientados a objetos.
- Refactoring. Testing. Metodologías de Diseño Ágiles.

PROGRAMA ANALÍTICO

I - Diseño Orientado a Objetos

1. La filosofía del proceso de desarrollo de software. Las etapas del proceso de desarrollo de software. Procesos de desarrollo iterativos e incrementales, basados

en modelos: utilidad de los modelos. Los modelos a través del proceso de desarrollo de software. Cualidades y clasificación de los modelos.

2. UML como Lenguaje de modelado. Diagrama de clases. Diagrama de estados. Diagrama de interacción.

II - Patrones de Diseño:

3. Introducción a Patrones. Definición de Patrón. Descripción de un patrón. Catálogo de Patrones.

4. Patrones de diseño. Definición. Descripción de un patrón de diseño. Organización del Catálogo de patrones de diseño. Utilidad de los patrones de diseño. Selección de los patrones de diseño. Uso de los patrones de diseño.

5. Patrones creacionales: Abstract Factory, Singleton.

6. Patrones estructurales: Composite, Decorador, Adapter, Proxy.

7. Patrones de comportamiento: Observer, State, Strategy, Template Method, Command.

III - Refactoring

8. Introducción a Refactoring. Utilidad del refactoring. Técnica de aplicación del refactoring. Casos de uso. Catalogo de refactoring.

9. Manipulación de métodos largos: Extract Method. lnline Method, Replace Temp with Queries, Replace Method with Method Object. Mover aspectos entre objetos: Move Method, Move Field, Extract class. Organización de datos. Self Encapsulate Field, Replace Data Value with Objects, Replace Type Code with Class / Subclass /State- Strategy. Simplificación de invocación de métodos: Rename Method, Replace Constructor with Factory Method, Parameterize Method. Simplificación de expresiones condicionales: Replace Conditional with Polimorfism. Manipulación de la generalización: Pull Up Method. Push Down Method, Extract Subclass, Extract Superclass, Form Template Method, Replace lnheritance with Delegation.

10. Refactoring hacia patrones. Unify interfaces with Adapter. Form Template Method. Replace conditional logic with strategy. Replace State-Altering Conditionals with State. Replace Hardcoded Notifications with Observer. Move Embellishment to Decorator.

IV - Frameworks

11. Introducción a Frameworks. Reutilización de software vs. reutilización de diseño. Clasificación de frameworks según su propósito.

12. Frameworks basados en herencia (white box frameworks). Frameworks basados en composición (black boxframeworks).

13. Elementos centrales en la implementación de un framework: Inversión de control, Hostposts, Frozenspots.

14. Las plantillas y los ganchos como generadores de hotspots e inversión de control. Su implementación con herencia y con composición.

15. Instanciación de frameworks; casos de estudio: Seaside, SUnit.

16. Diseño evolutivo de frameworks. El rol estratégico de los patrones, el refactoring, y los tests de unidad.

17. Documentación de frameworks: ejemplos, hotspot cards, y patrones

V - Test Driven Development

18. Testing. Importancia. Tipos de tests: de unidad, de integración, de aceptación. Metodología de desarrollo ágil TDD: "Test Driven Development". Relación entre refactoring y testing.

19. Patrones de tests de unidad: familia de frameworks XUnit.

20. El framework SUnit de test de unidad en Smalltalk.

METODOLOGÍA DE ENSEÑANZA

La cursada de la materia se organiza en clases teóricas y actividades prácticas.
En las clases teóricas se imparten los conceptos básicos de la materia, induciendo a la participación de los alumnos en la resolución de ejercicios de diseño que permite la incorporación activa de los conocimientos.

Hay dos tipos de clases prácticas, una donde se refuerzan los contenidos de la teoría con explicaciones de los JTP y donde se describe cada trabajo práctico, y otra donde los alumnos pueden acercarse a realizar consultas individuales sobre la resolución del trabajo práctico. Para ello se dispone de dos turnos (mañana y tarde) y algunas en turno especial. Además, las clases de consultas son en sala equipadas con computadoras y en aulas tradicionales.

EVALUACIÓN

La evaluación de los aspectos prácticos de la asignatura tiene lugar en un examen escrito, presencial con dos oportunidades de recuperación.

La aprobación final puede ser la realización de un proyecto y su defensa, el coloquio de promoción para quienes accedan a esta opción, o un examen final escrito.

Fechas tentativas de evaluación parcial

- Parcial: viernes 16/6, 18-21hs
- 1er Recuperatorio: viernes 30/6, 18-21 hs
- 2do Recuperatorio: viernes 4/8, 18-21 hs

A lo largo de la práctica los alumnos desarrollarán un trabajo integrador. El mismo cumplirá también la función de trabajo, optativo, de promoción. Los alumnos que cumplan en fecha y forma con las entregas pautadas accederán, en caso de aprobar la cursada, a un coloquio final de promoción. El coloquio será individual y versará sobre el trabajo realizado.

BIBLIOGRAFÍA OBLIGATORIA

1. Design Patterns. Elements of Reusable Objects Oriented Software. Garnma, Helm, Johnson, Vlissides, Addison-Wesley, Professional Computing Series.
2. Refactoring: lmproving the Design of Existing Codeo Fowler, Martin. Addison-Wesley, 1999.
3. Refactoring to Patterns. Joshua Kerievsky. Addison Wesley, 2004. ISBN: 0-321-21335-1
4. Implementing Application Frameworks: Object-Oriented Frameworks at Work (Hardcover). Mohamed E. Fayad (Editor), Douglas C. Schmidt (Editor), Ralph E. Johnson (Editor).

BIBLIOGRAFÍA COMPLEMENTARIA

1. Head First Design Patterns. Elisabeth Freeman, Bert Bates, Kathy Sierra - Computers -2004 -676 pages.
2. Extreme Programming Explained. Kent Beck and Cynthia Andres. Addison-Wesley, 2005.
3. Building Application Frameworks: Object-Oriented Foundations of Framework Design. Mohamed E. Fayad (Editor), Ralph E. Johnson (Author), Douglas C. Schmidt (Editor).
4. Domain-Specific Application Frameworks: Frameworks Experience by Industry (Hardcover). Mohamed E. Fayad (Editor), Ralph E. Johnson (Editor).

5. Johnson, R. E. 1997. Components, frameworks, patterns. In Proceedings of the 1997 Symposium on Software Reusability (Boston, Massachusetts, United States, May 17 - 20, 1997). M. Harandi, Ed. SSR '97. ACM Press, New York, NY, 10-17. [PDF]
6. Designing Reusable Classes. B. Foote, R. Johnson. Journal of Object-Oriented Programming, 1998.
7. D. Roberts and R. Johnson. Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks. Proceedings of Pattern Languages of Programs, Allerton Park, Illinois, September 1996. [PDF]
8. Kent Beck. Simple Smalltalk Testing: With Patterns. http://www.xprogramming.com/testfram.htm
9. Stephane Ducasse. SUnit Explained. http://www.iam.unibe.ch/~ducasse/

CRONOGRAMA DE CLASES Y EVALUACIONES

Clase	Contenidos/Actividades	Evaluaciones previstas
1	Presentación de la asignatura. Diseño OO. Características de un buen diseño OO.	
2	Patrones de diseño: introducción. El patrón Adapter. Prácticas de diseño en implementación.	
3	Patrónes Template Method y Composite. Práctica aplicando el patrón Adapter.	
4	Patrones Strategy y State. Práctica con Patrón Composite y Template method.	
5	Contrucción de aplicaciones web con el framework Seaside. Práctica con los patrones Strategy y State.	
6	Diseño de aplicaciones interactivas. Patrones de interacción web. Patrón Decorator. Práctica de construcción de aplicaciones con Seaside.	
7	Introducción a refactoring y Test Driven Development. Patrones de XUnit. Práctica de construcción de aplicaciones con Seaside.	
8	Refactoring: catálogo de refactorings de Fowler. El concepto de "bad smell". Bad smells clásicos. Práctica de testing de unidad.	
9	Refactoring hacia patrones de diseño. Refactoring to Template method. Refactoring to Strategy. Práctica de refactorings.	
10	Reuso; Librerías. Introducción a Frameworks Orientados a Objetos. Ejemplo de framework: Seaside, SUnit, entre otros. Práctica de refactoring hacia patrones.	

11	Plantillas y ganchos utilizando herencia y composición. Plantillas y ganchos en los hotspots. Prácticas integradoras.	
12	Frameworks blackbox: Ejemplo Roassal. Consulta previa al parcial.	
13	Integración de contenidos: Testing, Patrones, Refactoring, Frameworks. Prácticas integradoras.	Examen parcial
14	Prácticas integradoras. Clase de consulta.	
15	Prácticas integradoras. Clase de consulta.	Examen parcial recuperatorio.
16	Clase de consulta.	Examen parcial recuperatorio.

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

Consultas referidas a la teoría: Gustavo.Rossi@lifia.info.unlp.edu.ar, Alejandra.Garrido@lifia.info.unlp.edu.ar, Alejandro.Fernandez@lifia.info.unlp.edu.ar

Consultas referidas a la práctica: Arturo.Zambrano@lifia.info.unlp.edu.ar, Leandro.Antonelli@lifia.info.unlp.edu.ar, Diego.Torres@lifia.info.unlp.edu.ar

Firmas del/los profesores responsables: