PROGRAMA de Programación con Objetos 2

Carreras: Tecnicatura Universitaria en Programación Informática - Licenciatura en Informática

Asignatura: Programación con Objetos 2

Núcleo al que pertenece: Básico

Profesores: Diego Torres, Diego Cano y Matías Butti

Asignaturas Correlativas: Programación con Objetos 1

Objetivos:

Que les estudiantes:

- Adquieran ideas y técnicas que le permitan aprovechar distintas características de la orientación a objetos (como ser comportamiento, responsabilidad, polimorfismo, uso de patrones de diseño) para dar soluciones adecuadas a los desafíos planteados que apliquen, ya sea al proceso de diseño (como atacar un dominio) ya sea al software que se está diseñando (en particular, respecto de los conceptos de acoplamiento y cohesión).

- En particular, adquieran cierta fluidez en el manejo del concepto de tipo como herramienta para estructurar un programa: qué tipo se espera de un valor que se recibe, qué tipos puede ofrecer un valor que se define.

- Comprendan las consecuencias de trabajar en un entorno con chequeo estático de tipos.

- Comprendan la idea de diseño de software.

- Comprendan los desafíos que aparecen al construir aplicaciones más grandes, en las que el dominio a modelar es más complejo, y cuyo desarrollo lleva más tiempo.

- Conozcan formas de organizar los componentes de un programa en unidades mayores a los bloques de construcción básicos; p.ej. utilizando módulos, subsistemas y/o paquetes.

- Comprendan la necesidad de controlar el acoplamiento entre los componentes que forman un sistema de software.

- Se enfrenten a la necesidad de adaptar un producto de software en construcción o construido a cambios en los requerimientos que afectan al diseño definido.

- Hagan una experiencia de uso intensivo de un entorno de desarrollo integrado del estilo de los usados habitualmente en la industria.

Contenidos mínimos:

- Aproximación al diseño de software: en qué dimensiones puede crecer un proyecto de software, problemáticas que devienen de este crecimiento, necesidad de pensar en la organización de un sistema como elementos relacionados, pensando en la funcionalidad de cada elemento y de qué relaciones se establecen. Noción de decisión de diseño, considerando que el diseño es un proceso de toma de decisiones.

- Conceptos de acoplamiento y cohesión. Problemas que derivan de un grado de acoplamiento inadecuado.

- Vinculación entre las ideas básicas de diseño y el paradigma de objetos: objetos como elementos, conocimiento como relación, responsabilidades como funcionalidad de cada elemento; tipo y polimorfismo para comprender que ciertos elementos son intercambiables a efectos de lograr un diseño flexible y escalable.

- Características deseadas en un diseño de objetos: no repetición de implementación de lógica, capacidad de separar entre grupos de objetos cohesivos con responsabilidades aplicables al grupo.

- Patrones de diseño: idea de patrón, consecuencias del uso de algunos patrones respecto de las características del diseño y de las cualidades pretendidas del producto.

- Nociones sobre proceso de diseño: foco en las responsabilidades, pensar los problemas desde las características básicas del paradigma, pertinencia de iterar entre diseño en papel, codificación y test, relevancia de los diagramas de objetos y de clases.

- Metaprogramación, características reflexivas de un lenguaje de programación.

- Uso de un entorno integrado de software del estilo de los utilizados ampliamente en la industria, funcionalidades que provee, aprovechamiento de sus facilidades.

- Notación UML de los diagramas de clases, de objetos y de secuencia.

- Profundización del trabajo sobre testeo unitario y automático.

- Profundización del trabajo sobre manejo de errores, impacto del manejo de errores en el diseño.

Carga horaria semanal: 6 hs

Programa analítico:

Unidad 1: Introducción al diseño Orientado a Objetos.

¿Qué es diseñar? Abstracción, modelos, simplificación del mundo real. El diseño orientado a objetos. Descripción de un diseño como forma de comunicación. Introducción al lenguaje UML.Clases. Relaciones de Herencia, Colaboración, Dependencia e Implementación. Cardinalidad y navegabilidad en las relaciones. Modificadores de visibilidad y de modo (abstracto y de clase). Cohesión y Acoplamiento. Principios SOLID

Unidad 2: Diseño orientado a objetos como proceso

El diseño como un proceso iterativo desde el análisis del problema hasta la codificación en un lenguaje orientado a objetos. Diferentes formas de descripción del dominio. Casos de uso, user stories. Cohesión y Acoplamiento. Virtudes en la definición de objetos y métodos cohesivos. Problemas que devienen de un acoplamiento inadecuado. Diagrama UML de caso de Uso, documentación de un caso de uso. Diseño por capas. Capa de presentación. Capa de modelo. Capa de persistencia.

Unidad 3: Lenguajes de programación orientado a objetos con manejo de tipos

Introducción a un lenguaje orientado a objetos tipado. Introducción a Java. Principales características. La plataforma Java. Compilación e interpretación en la máquina virtual. Clases, objetos, métodos, constructores y paquetes en Java. Tipos primitivos y tipos de referencia.

Comparación con los lenguajes orientado a objetos no tipados. Colecciones. Herencia y visibilidad en Java. Manejo de Excepciones.

Unidad 4: Reutilización: Herencia y Composición

Herencia. Herencia de comportamiento. Herencia de estructura. Mensajes abstractos. Jerarquía. Generalización y especialización. Method Lookup. Herencia contrapuesta a la composición. Protocolo de mensajes. Noción de Tipo. Relación de las implicancias del concepto de tipos en lenguajes tipados y en lenguajes sin tipos. Objetos de diferentes clases con tipos similares.

Polimorfismo: ventajas en el uso. Polimorfismo como herramienta de abstracción. Reutilización por medio de objetos altamente cohesivos.

Unidad 5: Patrones de diseño de programación orientada a objetos.

Idea de patrón. Consecuencia del uso de algunos patrones respecto de las características del diseño y de las cualidades pretendidas del producto. Patrones de diseño en la ingeniería de software. Catálogo de patrones de diseño orientados a

objetos. Patrones de diseño creacionales, patrones de diseño estructurales, patrones de diseño de comportamiento.

Unidad 6: Desarrollo dirigido por tests

Desarrollo dirigido por Tests (TDD). Tipos de tests. Tests funcionales. Test de unidad. Estructura de un test de unidad: set up, ejecución, verificación y tear down, sistema cubierto por el test. Inputs indirectos, outputs indirectos. Testeo con dobles (Test doubles). Tipos de Test Dobles (Fake, Dummy, Spy, Mock). Framework de testing. JUnit y Mockito. Test de integración.

Unidad 7: Conceptos avanzados de diseño y programación

Framework de colecciones. Interfaces. Casting. Clases vs. Tipos. Generics. Uso básico de Annotations. Patrones de Diseño en el contexto particular de Java. Excepciones. Redefiniciones de métodos importantes: equals y hashCode.

Unidad 8: Frameworks

Reuso de código. Componentes, Aplicaciones y Frameworks. Frameworks de caja blanca. Frameworks de caja negra. Inversión de control. Hot spot. Frozen spot. Frameworks de caja negra en contraposición con los de caja blanca. Evolución de los frameworks

Bibliografía obligatoria:

- Rebecca Wirfs-Brock, Brian Wilkerson (Contributor), Lauren Wiener. Designing Object-Oriented Software. Prentice Hall PTR; 1991. ISBN 0136298257.
- Rebecca Wirfs-Brock and Alan McKean. Object Design: Roles, Responsibilities, and Collaborations. Addison-Wesley 2003. ISBN 0201379430.
- Thimoty Budd, Introduction to Object-Oriented Programming, An (3rd Edition), Addison Wesley; 3 edition. 2001, ISBN-10: 0201760312.
- Matt Weisfeld, The Object-Oriented Thought Process, Third Edition, Pearson Education, Addison Wesley. 2013. ISBN-13: 978-0- 672-33016-2.
- Garnma, Helm, Johnson, Vlissides, Design Patterns. Elements of Reusable Objects Oriented Software. Addison-Wesley, Professional Computing Series. 1995.
- Gerard Meszarons. Xunit Test Patterns: Refactoring Test Code, Addison-Wesley Signature Series. 2007.
- David Astels. Test Driven Development: A Practical Guide. Coad Series. 2007.
- Fayad, Mohamed, and Douglas C. Schmidt. "Object-oriented application frameworks.". 1997. Communications of the ACM 40.10: 32-38.
- Martin C. Robert, Martin Micah. Agile Principles, Patterns, and Practices in C#. Prentice Hall. 2006. ISBN-10: 0-13-185725-8.

Bibliografía de consulta:

- Fowler Martin, Scott Kendall, UML gota a gota. Addision Wesley IBEROA. Edición 1999 ISBN 9684443641.
- Booch Grady, Jaboson Ivar, Rumbaugh James. El lenguaje unificado de modelado. Addision Wesley IBEROA, Edición 2000, ISBN 8478290281.
- Suzanne Skublics, Edward J. Klimas, David A. Thomas, John Pugh (Foreword) Smalltalk With Style, Pearson Education POD; 1 edition (May 21, 2002) ISBN: 0131655493.
- James Gosling, Bill Joy, Guy Steele, Gilad Bracha. The Java Language Specification, Third Edition. 1996.
- Ian Sommerville. Software Engineering. Addison-Wesley; 9 edition. 2010
- Szyperski, Clemens. "Components vs. objects vs. component objects." Proceedings of OOP. Vol. 1999. 1999.
- M. E. Fayad, D. C. Schmidt, and R. E. Johnson, Building Application Frameworks: Object-oriented Foundations of Framework Design. New York, NY, USA: John Wiley & Sons, Inc., 1999.
- Roberts, Don, and Ralph Johnson. "Evolving frameworks: A pattern language for developing object-oriented frameworks." Pattern languages of program design 3. 1996: 471-486.
- W. Pree, “Hot-spot-driven framework development,” in Summer School on Reusable Architectures in Object-Oriented software Development, 1995, pp. 123–127.
- Johnson, Ralph E., and Brian Foote. "Designing reusable classes." Journal of object-oriented programming 1.2. 1988: 22-35.

Organización de las clases:

La materia se organiza en dos tipos de clases: clases teóricas y clases prácticas. Estas últimas pueden ser de tipo trabajos prácticos o trabajos de laboratorio. Las clases teóricas serán de una carga semanal de 3 horas, en las mismas se introducirán y explicarán los temas indicados en el programa analítico. Las clases teóricas darán el marco general de discusión y se complementarán con la bibliografía obligatoria.

La mecánica de cursada para les alumnes consiste en asistir a las clases teóricas y luego resolver una serie de trabajos prácticos que tratarán los temas teóricos en diferentes formatos de actividades:

- Ejercicios teórico / práctico para que puedan profundizar los conceptos vistos en las clases teóricas y los contrapongan con la bibliografía obligatoria de la materia.

- Ejercicios prácticos para resolver aplicando los conceptos teóricos en la resolución de problemas o situaciones. Estos ejercicios pueden resolverse en parte en el horario de trabajos prácticos y en parte fuera de los horarios de la materia.

- Ejercicios de laboratorio, de mayor envergadura que los ejercicios prácticos y que requieren mayor tiempo en el análisis, discusión y resolución. Estos proyectos están diseñados para realizarse en clase durante el horario de trabajos prácticos.

Trabajos Prácticos

TP 1 Introducción al diseño Orientado a Objetos (OO). Los objetivos de este trabajo práctico son profundizar qué involucra el proceso de diseño orientado a objetos.

Formas de comunicar un diseño. Realizar diseños utilizando el lenguaje UML con la notación básica para diagramas de clases: Relaciones de Herencia, Colaboración, Dependencia e Implementación, cardinalidad y navegabilidad en las relaciones. Modificadores de visibilidad y de modo (abstracto y de clase). Incorporar los conceptos de Cohesión y Acoplamiento en el diseño OO . Introducir los principios SOLID

TP 2: Diseño orientado a objetos como proceso

Los objetivos de este práctico son comprender el diseño como un proceso iterativo desde el análisis del problema hasta la codificación en un lenguaje orientado a objetos. Adquirir habilidades para analizar y describir el dominio mediante Casos de uso, user stories. Profundizar en las buenas prácticas del diseño OO: métodos cohesivos. , acoplamiento adecuado, diseño por capas (Capa de presentación. Capa de modelo. Capa de persistencia).

TP 3: Lenguajes de programación orientado a objetos con manejo de tipos.

Los objetivos de este trabajo práctico es introducir los lenguajes orientados a objetos tipados, a través del lenguaje Java. Quienes cursen lograrán conocer las características principales de este lenguaje, el funcionamiento de la compilación y la interpretación con máquinas virtuales. Además, se presenta como objetivo incorporar la sintaxis para la programación de clases, métodos, constructores, colecciones, visibilidad, herencia manejo de colecciones y paquetes en Java, incluyendo las características de tipos primitivos y de referencia. Como último objetivo, el tp posee, iniciar la discusión en la comparación de lenguajes tipados vs. no tipados.

TP 4: Reutilización: Herencia y Composición

El objetivo de este práctico es incorporar los conceptos de herencia y composición en las decisiones de diseño. Tomando como ejes la herencia de comportamiento y de estructura, la implicancia de comportamiento abstracto. Esto incluye poder incorporar los conocimientos sobre jerarquía, method lookup, protocolos y tipos de objetos. Además, se busca que se incorpore el concepto de polimorfismo como herramienta de abstracción y la relación con la reutilización de objetos altamente cohesivos.

TP 5: Patrones de diseño Adapter y Strategy

El objetivo de este práctico es introducir al / a la estudiante en la aplicación de patrones de diseño, tomando dos de los patrones más sencillos e intuitivos: los patrones Adapter y Strategy. Sobre la base de estos dos patrones de diseño, se busca que quienes cursen comprendan los lineamientos generales y la estructura de documentación de patrones, como así también su aplicación e implementación en un lenguaje concreto de programación. Se busca también que el estudiantado tome conciencia de que ya posee los conocimientos fundamentales que conforman la definición de mayor abstracción que constituye un patrón.

TP 6: Patrones de diseño Composite y State

El objetivo de este práctico es profundizar el estudio de patrones de diseño concentrándose en algunos de mayor complejidad. Mediante estos dos patrones se enfrentarán a desafíos en cuanto al uso de recursión, tanto en la definición de estructura de datos como de algoritmos, asimismo hará un uso exhaustivo de técnicas fundamentales del paradigma orientado a objetos, como lo son la delegación, el polimorfismo y la utilización de jerarquías polimórficas.

TP 7: Patrones de diseño Template Method y Observer

En este práctico se trabaja profundamente el concepto de abstracción mediante el patrón Observer, culminando en implementaciones concretas y flexibles en un lenguaje de programación. Quien curse la materia, podrá tangibilizar el poder de abstracción del paradigma identificando situaciones en las que puede abstraerse no sólo del “cómo” y del “quién” implementa un determinado comportamiento sino también del “qué” es lo que se implementa. Por otro lado, el patrón Template Method permite ahondar en las abstracciones algorítmicas que pueden darse dentro de una misma jerarquía de clases.

TP 8: Desarrollo dirigido por tests

En este práctico se trabajará en la práctica real de la metodología Desarrollo dirigido por Tests (TDD). Quien curse la materia, realizará desarrollos concretos guiados por esta metodología, y deberá aplicar técnicas de testeo avanzadas que requerirán el uso de distintos tipos de Test Doubles. Para esto deberá programar tests de unidad mediante frameworks y herramientas de amplia utilización en la industria actual, como lo son JUnit y Mockito. Pasados estos tests, el estudiantado realizará tests de integración de las distintas partes del modelo para consolidar la integridad funcional del mismo.

TP 9. Conceptos avanzados de diseño y programación: framework de colecciones.

En este práctico se pondrán en práctica las estructuras y la API que provee el framework de colecciones Java.

TP 10. Conceptos avanzados de diseño y programación: Interfaces y casting.

Se pondrá en práctica el concepto de interfaces como instrumento para favorecer el polimorfismo de clases no necesariamente relacionadas y para diseño de contratos. También se aborda la necesidad de disponer de herramientas de casting en lenguajes estáticamente tipados y las diferencias entre el downcasting y el upcasting. Por último se pone pone en práctica la diferencia entre identidad e igualdad y la redefinición de equals y hashCode.

TP 11: Frameworks.

Ejercicios que permiten a quienes cursen la integración de annotations, generics, frameworks de colecciones y patrones de diseño para el diseño y desarrollo de frameworks. Se presentan ejercicios que permitan distinguir frameworks de caja blanca y de caja negra. Incluye también ejercicios que permitan al estudiantado entender la importancia de la Inversión de control.

Modalidad de evaluación:

Los mecanismos de evaluación en modalidades libre y presencial de esta asignatura están reglamentados según los siguientes artículos del Régimen de estudios de la UNQ (Res. CS 201/18).

En la modalidad de libre, se evaluarán los contenidos de la asignatura con un examen escrito, un examen oral e instancias de evaluación similares a las realizadas en la modalidad presencial.

CRONOGRAMA TENTATIVO

Semana	Tema/unidad	Actividad*				Evaluación
		Teórico	Práctico			
			Res Prob.	Lab.	Otros Especificar	
1	Conceptos preliminares de UML e introducción al Diseño	x				
2	El diseño como un proceso iterativo	x	x	x		
3	Cohesión y Acoplamiento desde la POO	x	x	x		
4	Lenguajes Orientados a Objetos con Manejo de tipos: Java		x	x		
5	Evolución del diseño con la aparición de cambios	x	x	x		
6	Clases Abstractas – Herencia vs Composición - SOLID	x	x	x		
7	Articulación en lenguajes tipados como Java / Trabajo grupal articulatorio integral (TGAI)		x	x		
8	Primer Parcial					x
9	Conceptos preliminares de Patrones de Diseño / Patrones de comportamiento	x				
10	Recuperatorio del primer parcial					x
11	Patrones de estructurales / Seguimiento del TGAI	x	x	x		x
12	Patrones estructurales	x	x	x		
13	Desarrollo dirigido por tests.	x				
14	Segundo Parcial / Seguimiento del TGAI					x
15	Test doubles	x	x	x		
16	Recuperatorio del segundo parcial					x
17	Conceptos avanzados de diseño y programación	x				
18	Recuperatorio flotante del primer y segundo parcial / Finalización del TGAI					x

***INDIQUE CON UNA CRUZ LA MODALIDAD**

Firmado digitalmente por: SUAREZ Mariana Alejandra
Directora del Departamento de Ciencia y Tecnologla
UNIVERSIDAD NACIONAL DE QUILMES