PROGRAMA de Programación con Objetos 1

Carreras: Tecnicatura Universitaria en Programación Informática - Licenciatura en Informática

Asignatura: Programación con Objetos 1

Núcleo al que pertenece: Básico

Profesores: Máximo Prieto, Leonardo Gassman y Nahuel Palumbo

Asignaturas Correlativas: Introducción a la Programación

Objetivos:

El objetivo de esta materia es brindar los conceptos básicos de la Orientación a Objetos al grupo estudiantil. Esto se hace a través de una perspectiva paradigmática que ayuda a producir una manera específica de pensar al momento de construir software.

Para esto, se utiliza un enfoque pedagógico constructivista, que permite desarrollar, a lo largo del curso, el paradigma de objetos en forma pura —sólo hay objetos que pueden colaborar enviándose mensajes— y mínima —sólo los elementos esenciales que conforman su matriz disciplinar, los que, a su vez, permiten la construcción de elementos más complejos y sofisticados—.

Este enfoque permite aprender a pensar siempre en términos de objetos, lo que permite adecuarse a las limitaciones tecnológicas de las plataformas de desarrollo existentes en el mercado, sin perder las ventajas brindadas por el paradigma.

Los objetivos específicos son que le estudiante:

- Pueda modelar una situación sencilla, pero no trivial, usando correctamente las nociones básicas del paradigma de objetos: objeto, mensaje, clase, comportamiento, responsabilidad, protocolo.
- Entienda cómo se maneja el estado en el paradigma de objetos, esto es, haciendo que los objetos se conozcan entre sí.
- Conozca la forma básica de manejar el análogo a estructuras de datos, que son las colecciones.
- Entienda qué es el polimorfismo, y pueda usarlo adecuadamente en situaciones concretas. Adquiera cierta pericia en reconocer los tipos que puede tener un objeto, y en pensar en términos de tipo/interface.
- Entienda qué es la herencia, pueda distinguir formas adecuadas e inadecuadas de aprovecharla, y pueda usarla adecuadamente en situaciones concretas.

- Se enfrente a situaciones en las que resultan pertinentes formas de relacionar los objetos menos intuitivas/obvias, conozca el concepto de patrón de diseño, y pueda utilizar algunos patrones en situaciones concretas.

- Adquiera la costumbre de trabajar con testeo automático y unitario, vea que los tests también pueden verse como manuales de uso de los elementos que conforman un programa.

- Adquiera nociones básicas sobre manejo de excepciones: qué es un excepción y qué no, qué puede hacerse cuando en un programa se detecta una situación de excepción.

Contenidos mínimos:

- Conceptos fundantes del paradigma: objeto y mensaje. Visión externa del objeto: dispositivo computacional capaz de recibir mensajes y otorgar respuestas adecuadas. Relevancia de estos conceptos (con que objetos cuento, que mensajes le puedo enviar a cada uno) en el desarrollo de software.

- Concepto de polimorfismo en objetos, comprensión de las ventajas de aprovecharlo.

- Protocolo/interfaz, concepto de tipo en objetos, comprensión de que un objeto puede asumir distintos tipos. La interfaz como contrato al que se comprometen ciertos objetos, posibilidad de reforzar ese contrato.

- Estado en el paradigma de objetos: referencias, conocimiento, estado interno.

- Métodos, clases, herencia, method lookup.

- Conceptos de responsabilidad y delegación, su rol al definir una trama de objetos que responde a requerimientos determinados.

- Colecciones: conceptualización como objetos, caracterización a partir de los conceptos de protocolo y responsabilidad, protocolo, acceso a sus elementos.

- Testeo automático y repetible, test como comprobación tanto del correcto funcionamiento como de que los objetos definidos son efectivamente usables.

- Nociones básicas sobre manejo de excepciones: distinción entre excepción y valor de retorno, acciones posibles al detectar una situación de excepción. Interrupción del flujo de ejecución: modelado mediante estructuras de control.

Carga horaria semanal: 8 hs

Programa analítico:

Unidad 1: Paradigma y Modelo Computacional

Concepto de Paradigma. Concepto de Modelo Computacional. Relación entre ambos.

Unidad 2: Paradigma de Orientación a Objetos

Programa. Objeto. Identidad. Mensaje. Protocolo. Colaboraciones. Colaboradores (internos y externos). Nombres (locales, globales, particulares)
Método. Polimorfismo y Late Binding.

Unidad 3: Modelo Computacional de Objetos

Sintaxis: Asignación. Expresiones. Tipos de Mensajes. Semántica: Precedencias

Unidad 4: Representación de conocimiento

Creación de Objetos. Clases. Prototipos. Mecanismos de Abstracción
Niveles de abstracción. Objetos, Clases y Metaclases. Clasificación (Clases abstractas y concretas). Subclasificación. Redefinición de Protocolos. Mecanismos de Sharing. Herencia. Delegación. Tipos. Intercambiabilidad. Aserciones y Contratos. Excepciones. Jerarquías Polimórficas

Unidad 5: Modelos Básicos con Objetos

Destrucción de Objetos. Automática. Manual. Magnitudes. Lógica Booleana. Contextos de Ejecución. Colecciones

Unidad 6: Ambientes de Objetos

Inspectores. Generales. Especializados. Browser de Clases. Debugger.

Bibliografía obligatoria:

- Rebecca Wirfs-Brock et al., Object Design: Roles, Responsibilities and Collaborations, Addison-Wesley Professional, 2002 ISBN 0201379430

- Kent Beck, Test-Driven Development: By Example, Addison-Wesley Professional, 2003 ISBN 9780321146533

- David West, Object Thinking, Microsoft Press, 2004 ISBN 0735619654

- Stéphane Ducasse, Dimitris Chloupis, Nicolai Hess, and Dmitri Zagidulin. Pharo by Example 5 (https://books.pharo.org/updated-pharo-by-example/), 2018

Herramientas

- CuisUniversity (https://sites.google.com/view/cuis-university)

Bibliografía de consulta:

No posee

Organización de las clases:

1. Área teórica: El objetivo de las clases teóricas es brindar al grupo estudiantil los elementos conceptuales de arquitectura que ofrece la Orientación a Objetos. Entre los principales elementos básicos de arquitectura se encuentran los objetos en sí mismos, los protocolos, con sus consiguientes contratos, las clases, concretas y abstractas, y las jerarquías polimórficas.

La materia se inicia con una primera clase teórica que introduce lo mínimo indispensable para que puedan comenzar a programar de inmediato. Las siguientes clases teóricas se intercalan con las clases prácticas de modo tal de ofrecer los conceptos necesarios para resolver los problemas concretos con los que se enfrentan durante la realización de ejercicios de programación.

2. Área Práctica: Las clases prácticas se dan por medio de ejemplos concretos desarrollados en máquina y proyectados en una pantalla para que puedan seguirlos en sus propias computadoras, a fin de facilitar la resolución de ejercicios de programación, siempre en máquina, diseñados para que puedan experimentar cierto tipo de problemas que exigen la puesta en práctica, de manera iterativa e incremental, de los conceptos incluidos en la materia.

El ambiente de desarrollo utilizado, tanto por el grupo estudiantil para la resolución de ejercicios como por el cuerpo docente para desarrollar ejemplos en las clases teóricas y prácticas es Smalltalk. ¿Por qué Smalltalk? Porque es el ambiente de desarrollo que mejor acompaña al marco conceptual contenido en el paradigma de objetos, y porque es un ambiente dinámico, lo cual permite que se abstraigan de problemas propios de la programación que no están relacionados con el paradigma, y porque pueden entrar en contacto directo con los objetos por medio de inspectores en tiempo real (se evita la separación del tiempo de programación del de ejecución o run time). No es menos importante ya que es muy utilizado en la industria, lo que permite que se experimente con un ambiente de desarrollo profesional bien en el inicio de la carrera.

La distribución de Smalltalk elegida es CuisUniversity, dado que cubre todas las necesidades de la materia y es un ambiente gratuito y abierto, de fácil obtención y

desarrollado en Argentina por equipos docentes de diferentes universidades, entre ellas UBA (Universidad de Buenos Aires), UNQ (Universidad Nacional de Quilmes) y UCA (Universidad Católica Argentina).

3. Prácticos: En el transcurso de la cursada, al grupo estudiantil se les irán presentando una serie de actividades prácticas (Prácticos) donde en cada uno se presentan una serie de enunciados donde el grupo estudiantil debe ir resolviendo, estas tienen como fin afianzar y ejercitar el conocimiento que fue brindado en las clases teóricas. Los trabajos prácticos que deben realizar son:

1. *Práctico 1:* el objetivo de este práctico es familiarizarse con el ambiente de programación brindado y trabajar con *DenotativeObjects,* es decir, solo con objetos, sin ver la complejidad de la *clasificación*, presentando colaboraciones sencillas entre objetos, uso de colaboradores internos y externos.

2. *Práctico 2:* en este práctico, se comienza a trabajar con *clasificación* y *subclasificación*, con planteos jerárquicos sencillos, de los cuales permitirá ejercitar el diseño de jerarquías, la herencia y method lookup; utilización de colecciones con un manejo simple.

3. *Práctico 3:* como objetivo de este práctico se pretende afianzar los conocimientos necesarios para el *trabajo final*, de esto se desprende, planteo de jerarquías polimórficas más complejas, manejo de excepciones y violación de contratos, junto con un manejo de colecciones más complejo.

4. Trabajo Final: Hacia el último mes y medio de cursada, el grupo estudiantil se agrupará de a pares, y a cada grupo se le asignará une docente como supervisor de su trabajo, un tema para el desarrollo del trabajo y el enunciado correspondiente, el cual sirve como guía, pero no define límites estrictos, los cuales se van definiendo en consenso con le docente a cargo de cada grupo. A cada grupo se le exige que haga una entrega semanal. Estas entregas funcionan como puntos de control y permiten evaluar la evolución en lo que respecta a los resultados intermedios obtenidos, lo que, a su vez, muestra el proceso de programación seguido por el grupo. Es gracias al desarrollo de este trabajo final que el grupo estudiantil logra integrar todos los conceptos aprendidos y, de este modo, terminan de asimilarlos.

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 presencial, se evaluarán los contenidos de la asignatura mediante un examen por computadora y un trabajo final grupal donde se evaluarán todos los contenidos de la materia. En el caso de la instancia del primer integrador, se tomará un examen por computadora evaluando todo el contenido de la materia.

En la modalidad de libre, se evaluarán los contenidos de la asignatura mediante un examen por computadora, similar al integrador realizado en la modalidad presencial.

CRONOGRAMA TENTATIVO

Semana	Tema/unidad	Actividad*				Evaluación
		Teórico	Práctico			
			Res Prob.	Lab.	Otros Especificar	
1	Presentación de la materia. Concepto de programa. Discusión acerca de definición tipo Von Neuman. Necesidad de una definición más abstracta. Programa como representación (ejecutable) de conocimiento. Programación como proceso de observación y aprendizaje. Concepto de paradigma como construcción de una subjetividad colectiva. Repaso, iterativo e incremental, del concepto de programa y de paradigma (metáfora de juegos de mesa: Ajedrez). Base canónica y construcciones paradigmáticas. Consecuencias concretas de estas definiciones. Programa como eso que se ejecuta. Dominios de negocios vs. dominios naturales. Paradigma como puro y minimal. Definición de programa OO. Consecuencias conceptuales. Definición de objeto como representación esencial de una cosa de la realidad. Definición de cosa y esencia como utilidad, propósito, rol. Lenguaje natural y condensación. Definición de objeto como ente computacional: Comportamiento (mensajes y métodos) e Identidad.	X				
2	Construcción de un simple programa de manera iterativa e incremental. Cómo interpretar un enunciado, identificar objetos esenciales, mensajes que deberían entender. Foco en la parte metodológica. Ir comprendiendo el dominio, tomando simples decisiones de acuerdo al alcance. Colaboradores internos y externos. Tipos de mensaje en Smalltalk. Análisis detallado de colaboraciones.	x		x		
3	Repaso: división de responsabilidades, mensajes a self, refactors para simplificar	x		x		

	métodos y ganar legibilidad. Introducción a colecciones: qué es una colección, qué variantes hay, qué mensajes básicos entiende. Cómo utilizarlas en CuisUniversity. Concepto de programa como Representación de conocimiento. Programación como proceso de aprendizaje (iterativo e incremental). Importancia de comprender estos conceptos a fin de entender qué técnicas (pensando en TDD) y herramientas (Smalltalk, sUnit) necesitamos para programar bien y construir buenos programas, independientemente del lenguaje de programación. Relación en lenguaje natural y pensamiento. Necesidad de entrenar el lenguaje como medio para entrenar el pensamiento (cómo pensar en objetos). Concepto de paradigma o enfoque. Definición del enfoque de objetos: Programa OO, Objeto, mensaje, colaboración, estructura de colaboradores (internos e internos), método como conjunto de colaboraciones.					
4	Resolución de ejercicios en clase. Discusión de alternativas. Repaso de conceptos.	x		x		
5	Creación de objetos: clasificación vs prototipación. Subclasificación y herencia. Clases abstractas. Redefinición de protocolos	x		x		
6	Introducción a TDD y a las herramientas de CuisUniversity (System Browser, Debugger, SUnit). Resolución del ejercicio del Hormiguero haciendo mucho foco en metodología. SUnit, cómo definir tests, cómo escribir aserciones usando los mensajes #assert: y #deny:. Debugger, pieza fundamental para practicar TDD, creando métodos on-demand, inspeccionando el contexto de ejecución.			x		
7	Resolución de ejercicios en clase. Discusión de alternativas. Repaso de conceptos.	x		x		
8	Modelado de objetos que representan magnitudes, problemas asociados a ellas (igualdad e identidad). Ejemplo: ejercicio de distancias (metros y kilómetros). Resolución con TDD.	x		x		
9	Resolución de ejercicios en clase. Discusión de alternativas. Repaso de conceptos.	x		x		
10	Polimorfismo y Late Binding	x		x		
11	Polimorfismo: Consecuencias; Protocolos, contratos y tipos. Concepto de excepción	x		x		
12	Excepciones: qué son y para qué las usamos. Protocolo básico de una excepción, jerarquía de Exception. Ejemplos de excepciones: AssertionFailed, MessageNotUnderstood, CollectionIsEmpty, ZeroDivide. Cómo lanzar una excepción y cómo capturarla (mensaje #on:do:). Como verificar en un test que ocurrió una aserción	x		x		

	(mensajes #should:raise:, #shouldnt:raise:, #should:raise:withExceptionDo:). Ejemplos en CuisUniversity.					
13	Resolución de ejercicios en clase. Discusión de alternativas. Repaso de conceptos.	x		x		
14	Parcial					x
15	Comienzo del desarrollo del proyecto final. Discusión de enunciados	x		x		
16	Proyectos finales: alternativas de diseño	x		x		
17	Historia de OO. Recuperatorio	x				x
18	Squeakers. Entrega de proyectos finales	x				
19	Integrador. Cierre de Notas					x

***INDIQUE CON UNA CRUZ LA MODALIDAD**