Fecha de aprobación del programa: 30/10/2018
Vencimiento: 30/10/2020

Programación con Objetos I

Actividad curricular: Programación con Objetos I

Prerrequisitos: Introducción a la Programación

Carga Horaria:

Carga horaria total 128 horas

Carga horaria práctica: 96 hs

- Formación Experimental: 16hs
- Resolución de problemas: 80hs

Carga horaria semanal: 8 horas por semana

Objetivos:

Que el 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. 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. Colecciones: conceptualización como objetos, caracterización a partir de los conceptos de protocolo y responsabilidad, protocolo, acceso a sus elementos. Testeo automático y repetible. Nociones básicas sobre manejo de errores. Interrupción del flujo de ejecución: modelado mediante estructuras de control, concepto de excepción.

Programa analítico:

Unidad 1: Planteo inicial del paradigma de objetos: objeto y mensaje.

Noción de objeto como entidad computacional que exhibe comportamiento. Noción de modelo computacional, entes en la realidad y en el modelo. Interacción basada en el comportamiento, cómo obtener información.

Definición de un objeto: métodos, necesidad de estado interno para recordar valores entre interacciones, atributos. Comprobación del comportamiento de los objetos construidos.

Unidad 2: Referencias entre objetos, polimorfismo.

Necesidad de interacción entre objetos que conforman un modelo. Interacciones basadas en el comportamiento. Referencias entre objetos como parte del estado interno de los mismos. Grafo de referencias.

Polimorfismo, surge naturalmente al aparecer distintos objetos con los que un tercero puede interactuar indistintamente. Noción de mensaje, separación entre mensaje y método. Capacidad de un programa de interactuar con objetos definidos a posteriori, noción de contrato.

Identidad de objetos, diferencia entre igualdad e identidad.

Unidad 3: Colecciones y *closures*.

Colecciones como herramienta que permite que un objeto mantenga referencias a una cantidad indeterminada de otros objetos.

Comportamiento básico de las listas: longitud, acceso y modificación de sus elementos, agregado y remoción.

Conjuntos, diferencias de comportamiento con las listas.

Cómo se reflejan las colecciones en los grafos de referencia.

Recorridos sobre colecciones: filtro, transformación, verificaciones. Uso de mensajes que involucran una función a evaluar sobre cada elemento. Noción de *closure*.

Unidad 4: Testeo automático.

Relevancia de la comprobación de programas. Concepto de test automático, ventajas respecto de la capacidad de modificar programas construidos previamente. Elementos de un test: objetos de prueba, configuración de escenario, aserciones.

Unidad 5: Clases.

Necesidad de unificar la definición de comportamiento de distintos objetos. Noción de clase. Instanciación, objetos que son instancias de una clase. Seudovariable *self*. Constructores.

Unidad 6: Herencia.

Subclasificación como herramienta para obtener variantes de definición para comportamientos similares pero no iguales. Superclase y subclase.

Method lookup, en particular para mensajes enviados a *self*. Seudovariable *super*, redefinición de métodos.

Técnica de *template method*, definición única de algoritmos genéricos con partes que admiten múltiples variantes.

Unidad 7: Manejo de errores.

Error básico: intentar interactuar con un objeto en una forma no admitida por éste.
Errores relacionados con condiciones. Posibilidad de controlar estos errores. Noción de excepción, manejo básico de excepciones.

Unidad 8: Nociones iniciales sobre diseño de software.

Problemáticas que surgen al resolver problemas que involucran objetos de varias clases. Técnicas de aprovechamiento del polimorfismo. Conveniencia de pensar en la interacción guiada por el comportamiento. Noción de programa como nube de objetos que interactúan entre sí.

Bibliografía obligatoria:

Secuencia de materiales sobre orientación a objetos, Fernando Dodino y otros. Disponible en http://www.wollok.org/documentacion/apuntes/ .

Guías de ejercicios y material adicional generado por el grupo docente de la Unviersidad Nacional de Quilmes; Carlos Lombardi, Leonardo Gassman y otros. Disponible en https://objetos1wollokunq.gitlab.io/ .

Herramientas

- Wollok, http://www.wollok.org/ .
  Se trata de un lenguaje de programación acompañado de un entorno integrado de programación, desarrollados en Argentina por docentes de distintas universidades públicas.
  Wollok combina un enfoque pedagógico orientado a estudiantes que están haciendo sus primeras armas en programación orientada a objetos, y con poca experiencia en programación. Al mismo tiempo, las características del lenguaje y del entorno de

programación son similares a las de lenguajes y herramientas ampliamente utilizados en la industria, como ser los lenguajes Java y JavaScript, y el entorno Eclipse.

Bibliografía de consulta:

- Programación Orientada a Objetos con Java usando Blue, 5ta edición, David Barnes y Michael Kölling, Pearson, 2014. ISBN(13): 9788483227916
- Programación Orientada a Objetos, Carlos Fontenla y Nicolás Páez, 2015. Disponible online en https://legacy.gitbook.com/book/nicopaez/poo/details .
- Introducción a la Programación Orientada a Objetos, Luiz R. Izquierdo. Disponible online en http://luis.izqui.org/resources/ProgOrientadaObjetos.pdf .
- Designing Object-Oriented Software, Rebecca Wirfs-Brock et al. – Prentice Hall, 1990. ISBN 0136298257
- Object Oriented Software Construction, Bertrand Meyer (2nd edition). Prentice Hall, 1997. ISBN (13) 9780136291558.

Organización de las clases:

El dictado está organizado a partir de la idea de integrar fuertemente la transmisión de contenidos (la llamada comunmente *teoría*) con la utilización de los mismos en la resolución de problemas (que puede asimilarse con lo conocido comunmente como *práctica*).

Luego de una enunciación inicial de las ideas fundantes del paradigma tal cual se transmite en esta propuesta, *i.e.* objeto y comportamiento, se desarrolla un esquema iterativo en el que cada tema que se presenta, está motivado por un problema que no admite soluciones apropiadas con las herramientas conceptuales presentadas previamente. De esta forma, se busca que el *saber* de los conceptos introducidos se refleje en el *saber hacer* de la resolución de problemas donde tales conceptos se aplican, y que el *hacer* se base efectivamente en los conceptos presentados.

Los problemas son resueltos directamente mediante la implementación de programas, lo cual resulta en una unificación de las ideas de “clase de resolución de problemas” y “clase de laboratorio”.

La amplia carga horaria de la materia habilita a dedicar parte del tiempo de cursada en la resolución de problemas en la computadora, donde cada estudiante trabaja con la asistencia de los docentes y de sus compañeros.

Se busca que las clases tengan una dinámica participativa, consultando a los estudiantes sobre la resolución de los problemas que se plantean en clase, y alentando las preguntas y debates.

La organización propuesta para la cursada está fuertemente basada en la existencia de una serie de guías de ejercicios, que están graduados de forma de acompañar la evolución de los estudiantes y la secuencia pedagógica propuesta.

Finalmente, se destaca que la complejidad creciente de los problemas planteados a lo largo de la cursada alienta una comprensión *en espiral* de los temas presentados, dado que varios de ellos confluyen en la resolución de las últimas guías.

Modalidad de evaluación:

Se realiza por medio de dos parciales.

En el primero se busca que los estudiantes afirmen la comprensión de algunas problemáticas puntuales: referencias, polimorfismo, herencia, method lookup. Se realiza en papel, se plantea el análisis de código y de la ejecución del mismo, y la propuesta de mejoras que estén en línea con los conceptos introducidos.

El segundo parcial es la resolución de un problema que involucra la definición de varias clases, y la utilización de varias técnicas de programación y diseño ejercitadas en la cursada. Se hace en computadora, se entrega en forma digital.

La calificación de cada evaluación se determinará en la escala 0 a10, con los siguientes valores: 0, 1, 2 y 3: insuficientes; 4 y 5 regular; 6 y 7 bueno; 8 y 9 distinguido; 10 sobresaliente. La materia podrá aprobarse mediante: régimen de promoción directa, exámenes finales regulares y exámenes libres.

> Régimen de promoción directa (sin examen final): los/las estudiantes deberán aprobar las materias con siete (7) o más puntos de promedio entre todas las instancias evaluativas, sean éstas parciales o sus recuperatorios, debiendo tener una nota igual o mayor a seis (6) puntos en cada una de éstas. Todas las instancias evaluativas tendrán una posibilidad de recuperación. En el caso de los ausentes en la fecha original, el recuperatorio operará como única fecha de examen. El examen recuperatorio permite mantener la chance de la promoción siempre y cuando respete las condiciones de calificación respectiva.
> Exámenes finales regulares: para aquellos/as estudiantes que hayan obtenido una calificación de al menos de 4 (cuatro) y no se encuentren en las condiciones de promoción, deberán rendir un examen final que se aprobará con una nota no inferior a 4 (cuatro) puntos.

La asistencia no debe ser inferior al 75% en las clases presenciales.

Cronograma

Se indican los temas y ejercicios prácticos a desarrollar en cada semana de clases.

En todos los casos, los ejercicios consisten en implementaciones de distintos enunciados, para ser desarrollados utilizando el lenguaje y el entorno Wollok que constituyen el soporte principal de la materia. Algunos de estos ejercicios deben ser entregados como condición para rendir parcial.

Los enunciados de todos los ejercicios pueden consultarse en
https://obj1-unahur.github.io/repo-index.html , un sitio abierto a la comunidad.

CRONOGRAMA DE CLASES, PARCIALES Y RECUPERATORIOS		
Nº SEM	TEMAS A DESARROLLAR	EJERCICIOS PRÁCTICOS A DESARROLLAR
01	Conceptos fundantes del paradigma: objeto, mensaje. Implementación de un objeto: método, mensaje.	Pepita la golondrina (parte de “multipepita”).
02	Polimorfismo, interacción entre varios objetos.	Se agregan aves a Pepita: Pepón el Gorrión, Pipa la paloma. También Roque el entrenador de aves. (partes de “multipepita”) Sueldos de empleados a partir de básico, adicionales y descuentos. (“sueldo de pepe”). Empanadas Giménez.
03	Closures y colecciones. Tests.	Camión de transporte. Pepita game.
04	Más sobre colecciones.	Golosinas.
05	Referencias, diagrama de objetos.	Viajes en auto. Casa de Pepe y Julián.
06	Clases, instancias , instanciación. Uso correcto de self.	Flota de rodados.
07	Creación dinámica de instancias, repaso de referencias usando instancias. Repaso de tests. Polimorfismo entre instancias de distintas clases.	Infraestructura ferroviaria.
08	Consolidación de lo visto hasta el momento.	Primer parcial.
09	Herencia.	Plagas. Naves espaciales
10	Casos complejos de herencia. Method lookup. Template method.	Naves espaciales (continuación).
11	Nociones de diseño. Aprovechamiento del polimorfismo. Manejo básico de errores.	Oktubre Fest.
12	Práctica con ejercicos complejos. Repaso de test. Asignación de responsabilidades.	Obreros (revisado). Vendedores.
13	Consolidación.	Segundo parcial.
14	Lenguajes industriales: Java, JavaScript.	Sueldo de Pepe en Java. Multipepita, implementación en JavaScript.
15	Cierre de la materia. Recuperación.	Recuperatorio de 1º parcial
16	Recuperación.	Recuperatorio de 2º parcial