FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Planificaciones

9502 - Algoritmos y Programación III

Docente responsable: SUAREZ JOSE PABLO

OBJETIVOS

Esta materia busca que los alumnos adquieran los conceptos centrales de la programación orientada a objetos más algo de diseño y calidad de código, junto con algunas buenas prácticas básicas de construcción de software. Subsidiariamente, se estudia el desarrollo de aplicaciones de interfaz de usuario gráfica y algunas problemáticas típicas, como persistencia y concurrencia. Los alumnos deben aprender a programar en Java y Smalltalk.

CONTENIDOS MÍNIMOS

-

PROGRAMA SINTÉTICO

Primera parte: programación orientada a objetos y prácticas metodológicas
Resolución de problemas mediante objetos y mensajes. Diagramas de secuencia de UML. Comportamiento como noción central. Encapsulamiento. Polimorfismo (noción). Responsabilidades de los objetos. Uso en Smalltalk (ambiente Pharo).
Implementación del comportamiento de objetos con clases usando diseño por contrato y pruebas unitarias como especificaciones. Excepciones básicas. Automatización de pruebas unitarias. Principios de construcción de pruebas unitarias. Implementación en Smalltalk y SUnit.
Colaboraciones de objetos. Delegación y programación por diferencia. Relaciones entre clases: asociación y herencia. Diagrama de clases de UML. Implementación en Smalltalk.
Polimorfismo (profundización): Implementación en lenguajes con y sin chequeo estático e implementación en lenguajes sin clases. Interfaces en Java (noción).
Java como lenguaje con verificación estática. Polimorfismo en Java, con herencia y con interfaces.
Excepciones en general y excepciones chequeadas en Java.
Atributos de clase. Métodos de clase. Encapsulamiento y visibilidad. Inicialización, construcción, asociación y composición. UML y modelado (clases, secuencias, paquetes, estados).

Segunda parte: calidad de código, buenas prácticas y temas avanzados
Calidad de código. Diseño orientado a objetos (noción). Refactorización. TDD. Uso de dobles de prueba (noción). Buenas prácticas de XP. Integración y delivery continuos (noción).
Temas generales de diseño. Principios de diseño. Algunos patrones de diseño.
Construcción de aplicaciones con interfaz de usuario gráfica. MVC como patrón arquitectónico. Experiencia de usuario y usabilidad. Java FX.
Temas avanzados: información de tipos en tiempo de ejecución y reflexión, copia y clonación, genericidad.
Persistencia: serialización nativa y portable (nociones). Concurrencia y su implementación en Java.

PROGRAMA ANALÍTICO

Primera parte: programación orientada a objetos y prácticas metodológicas
Resolución de problemas mediante objetos y mensajes. Diagramas de secuencia de UML. Comportamiento como noción central. Encapsulamiento. Polimorfismo (noción). Responsabilidades de los objetos. Uso en Smalltalk (ambiente Pharo).
Implementación del comportamiento de objetos con clases usando diseño por contrato y pruebas unitarias como especificaciones. Excepciones básicas. Automatización de pruebas unitarias. Principios de construcción de pruebas unitarias. Implementación en Smalltalk y SUnit.
Colaboraciones de objetos. Delegación y programación por diferencia. Relaciones entre clases: asociación y herencia. Diagrama de clases de UML. Implementación en Smalltalk.
Polimorfismo (profundización): Implementación en lenguajes con y sin chequeo estático e implementación en lenguajes sin clases. Interfaces en Java (noción).
Java como lenguaje con verificación estática. Polimorfismo en Java, con herencia y con interfaces.
Excepciones en general y excepciones chequeadas en Java.
Atributos de clase. Métodos de clase. Encapsulamiento y visibilidad. Inicialización, construcción, asociación y composición. UML y modelado (clases, secuencias, paquetes, estados).

Segunda parte: calidad de código, buenas prácticas y temas avanzados
Calidad de código. Diseño orientado a objetos (noción). Refactorización. TDD. Uso de dobles de prueba (noción). Buenas prácticas de XP. Integración y delivery continuos (noción).
Temas generales de diseño. Principios de diseño. Algunos patrones de diseño.
Construcción de aplicaciones con interfaz de usuario gráfica. MVC como patrón arquitectónico. Experiencia de usuario y usabilidad. Java FX.

Temas avanzados: información de tipos en tiempo de ejecución y reflexión, copia y clonación, genericidad. Persistencia: serialización nativa y portable (nociones). Concurrencia y su implementación en Java.

BIBLIOGRAFÍA

Carlos Fontela, Texto de apoyo conceptual de Algoritmos y Programación III (Versión Beta 0.5 del libro "Programación Orientada a Objetos - 3ra. Edición"), 2017.
Andrew P. Black, Stéphane Ducasse, Oscar Nierstrasz, Damien Pollet, "Pharo por ejemplo", 2007.
Carlos Fontela, "Orientación a objetos con Java y UML", 2010.
Kent Beck, "Test Driven Development: By Example", 2003.
Bertrand Meyer, "Desarrollo de software orientado a objetos", 1999.
Steve McConnell, "Code Complete", 2004.
Robert Martin, "UML para programadores Java", 2004.
Carlos Fontela, "UML", 2011. Elizabeth Freeman, "Head-First Design Patterns", 2004. Martin Fowler, "Refactoring: Improving the Design of Existing Code", 2000. Kent Beck, "Extreme Programming Explained", 2004.

• Principios de diseño de Smalltalk, de Daniel H. H. Ingalls.
http://www.smalltalking.net/Papers/stDesign/stDesign.htm
• Apunte de Pruebas, Carlos Fontela y Pablo Suárez (disponible en campus de la materia)
• "Unit Testing Guidelines", http://geosoft.no/development/unittesting.html
• "8 Principles of Better Unit Testing", http://esj.com/Articles/2012/09/24/Better-Unit-Testing.aspx?p=1
• "Template Method & Strategy: Inheritance vs. Delegation", Robert Martin,
http://staff.cs.utu.fi/~jounsmed/doos_06/material/TemplateAndStrategy.pdf
• "Replace Conditional with Polymorphism", https://sourcemaking.com/refactoring/replace-conditional-with-polymorphism
• "Continuous Integration", de Martin Fowler, http://www.martinfowler.com/articles/continuousIntegration.html
• "What's a Model For?", Martin Fowler, http://martinfowler.com/distributedComputing/purpose.pdf
• "Generics in C#, Java, and C++ - a conversation with Anders Hejlsberg, by Bill Venners with Bruce Eckel",
http://www.artima.com/intv/genericsP.html
• "Extreme Programming", Kent Beck, capítulo 1,
http://ptgmedia.pearsoncmg.com/images/9780321278654/samplepages/9780321278654.pdf
• "Estado del arte y tendencias en Test-Driven development", Carlos Fontela,
http://web.fi.uba.ar/~cfontela/Fontela_EstadoDelArteTDD_UNLP_EIS.pdf
• Code Complete, Steve McConnell, capítulo 5 "Design in Construction", http://cc2e.com/File.ashx?cid=336xxx

RÉGIMEN DE CURSADA

Metodología de enseñanza

El plantel docente de la cátedra ha establecido para esta materia un enfoque de apropiación del aprendizaje por parte de los alumnos basado en las recomendaciones de enseñanza centrada en los estudiantes (LCT: Learner Centered Teaching).

Las clases de la materia son de diferentes tipos. En algunos cuatrimestres diferenciamos los tipos de clases en función de sus horarios, y en otros seguimos un esquema menos rígido. Esto se hace en función de la dotación docente y de la disponibilidad en cada horario.

Podemos clasificar las clases en 4 categorías: de consulta sobre trabajos prácticos; práctica interactiva; teórico-práctica; teoría pura.
Cada tipo de clase sigue el siguiente esquema:
Consulta. En estas clases, los docentes trabajan con los alumnos, guiándolos como tutores y/o evaluando el desempeño individual y/o grupal. Se trabaja siempre sobre computadoras, ya sean las que traen los alumnos o en laboratorios en la Facultad. No obstante, los alumnos desarrollan sus trabajos prácticos en forma autónoma y fuera del aula, contando con estas clases como puntos de referencia y control.
Práctica interactiva. En estas clases, los docentes presentan problemas a los alumnos, quienes los resuelven generalmente en computadora si requieren programar (no se "programa en papel"), de a pares y tratando que las parejas no sean las mismas en todas las clases. En algunas ocasiones, el docente puede comenzar mostrando cómo efectúa una determinada tarea, pero se limita a lo esencial, dejando luego lugar a los alumnos para que lo hagan ellos mismos. El compartir el ejercicio con un par facilita la apropiación del aprendizaje mediante la conversación, la discusión y la necesidad de entender los puntos de vista del otro.
Teórico-práctica. Estas son clases de contenidos, abordadas desde un punto de vista práctico. Para que los estudiantes aborden el proceso como propio, se van viendo los temas haciendo frecuentes puestas en común, con el docente haciendo las veces de moderador, y permitiendo a los alumnos opinar libremente y

discutiendo entre ellos, de forma de fomentar el sentido crítico. Las intervenciones del docente nunca son superiores a los 20 minutos, seguidas de una puesta en común, y la mayor parte de las veces de algo de ejercitación práctica. En algunas clases se realiza un role-play que facilita la apropiación del conocimiento. Teoría pura. Estás son también clases de contenidos, pensadas para aquellos temas que no requieren un abordaje práctico en el contexto de la materia. El profesor imparte, de manera teórica y tradicional algún tema y prepara a los alumnos para la siguiente unidad de conocimiento. En estas clases, el docente solicita realimentación a los alumnos en forma constante, tanto para conocer su nivel de comprensión como para que se involucren en la actividad.

En algunas ocasiones, antes de clase, se provee a los alumnos de un material para entrar en contacto con el tema, mediante lecturas (en su mayoría) o videos recomendados por los docentes. Esta es una actividad personal, fuera de las aulas, pero algunas de estas lecturas se analizan a fondo en clase luego de la tarea extra-áulica.

Los trabajos prácticos y los exámenes también son instancias de aprendizaje.

Modalidad de Evaluación Parcial

El examen parcial es escrito y en papel. No se evalúan las destrezas en programación sobre computadora, que están cubiertas por los trabajos prácticos. El mismo evalúa las habilidades de modelado orientado a objetos (UML, pruebas unitarias) y algunas cuestiones conceptuales mediante preguntas teóricas. En algunos casos, se pedirá mejorar una solución que contenga algún problema de diseño o conceptual.

El examen integrador tiene una parte de trabajo en computadora seguida de una evaluación oral. Los alumnos trabajarán sobre la mejora del diseño de un pequeño programa, en computadora, que luego será trabajada en forma oral con un docente. En esta parte oral, el docente podrá ahondar en los mismos temas evaluados en computadora, en otros temas de la materia, o pedir alguno de los trabajos prácticos desarrollados durante el cuatrimestre para charlar sobre mejoras o cambios al mismo. Una ventaja fundamental de la evaluación oral es que aumenta las posibilidades de ahondar en ciertos temas, tanto por parte del alumno como del docente, pudiendo determinar mejor cuánto sabe el alumno del tema.

CALENDARIO DE CLASES

Semana	Temas de teoría	Resolución de problemas	Laboratorio	Otro tipo	Fecha entrega Informe TP	Bibliografía básica
<1> 09/03 al 14/03	Presentación de la materia. POO y resolución de problemas. Objetos y mensajes. Diagramas de secuencia. Comportamiento y su implementación. Encapsulamiento y Polimorfismo (nociones).	Familiaridad con Smalltalk y el ambiente Pharo. Ejercicio en computadora por los alumnos.	Enunciado TP0		No determinado aún	Ver bibliografía en datos generales de la materia
<2> 16/03 al 21/03	Implementar comportamiento de objetos (clases sólo como una de las maneras de implementar), con contratos y pruebas. Se plantea desde la necesidad de especificar y surgen naturalmente el contrato y las pruebas.	Role play Máquina de café (para descubrir responsabilidades de los objetos, encapsulamiento y polimorfismo, delegación). Sunit y aplicación de diseño por contrato con pruebas unitarias sobre un ejercicio en computadora por los alumnos. Uso de SUnit sobre TP0.	Vence TP0		No determinado aún	Ver bibliografía en datos generales de la materia
<3> 23/03 al 28/03	Delegación. Resolver un ejercicio con delegación. Herencia. Resolver con herencia. Refactorización. TDD y subproductos de las pruebas. Diagramas de clases.	Ejercicio en computadora y otro de modelado a cargo de los alumnos, con puesta en común posterior.	Enunciado TP1S		No determinado aún	Ver bibliografía en datos generales de la materia
<4> 30/03 al 04/04	Polimorfismo en Smalltalk. Métodos y clases abstractos. Ídem en Java. Interfaces como mecanismo para polimorfismo sin herencia.	Revisión de las primeras 3 lecturas obligatorias (Principios de Diseño de Smalltalk, Unit Testing Guidelines y 8 Principles of Better Unit Testing). Ejercicio de refactorización			No determinado aún	Ver bibliografía en datos generales de la materia

Semana	Temas de teoría	Resolución de problemas	Laboratorio	Otro tipo	Fecha entrega Informe TP	Bibliografía básica
		que implique introducir polimorfismo.				
<5> 06/04 al 11/04	Excepciones en general y las chequeadas de Java. Atributos y métodos de clase. Encapsulamiento y visibilidad. Inicialización, construcción, asociación y composición.	Ejercicio de modelado con clases y secuencias. Programar una parte.			No determinado aún	Ver bibliografía en datos generales de la materia
<6> 13/04 al 18/04	Calidad de código. Algunas buenas prácticas tomadas de XP: refactoring, TDD, simplicidad… Integración y delivery continuos como ideas.	Ejercicio de funciones. Resuelven los alumnos y puesta en común. Luego se muestra otra resolución.			No determinado aún	Ver bibliografía en datos generales de la materia
<7> 20/04 al 25/04	UML: estados, paquetes	Mejorar el código de una aplicación. Repaso pre-parcial.	Vence TP1S		No determinado aún	Ver bibliografía en datos generales de la materia
<8> 27/04 al 02/05	Clase de recuperación o a cargo de invitados	Clase de recuperación o a cargo de invitados			No determinado aún	Ver bibliografía en datos generales de la materia
<9> 04/05 al 09/05	Parcial	Resolución del parcial. Mostrar resolución en Smalltalk y resolver en código y PC en Java.	Enunciado TP1J		No determinado aún	Ver bibliografía en datos generales de la materia
<10> 11/05 al 16/05	Principios de diseño y algunos patrones.	Patrones de diseño, llegando por aplicación de principios de buen diseño y refactorización. Revisión de la lectura "Inheritance vs Delegation" de Robert Martin.			No determinado aún	Ver bibliografía en datos generales de la materia
<11> 18/05 al 23/05	Temas avanzados. RTTI y reflexión. Copia y clonación. Generics.	Java FX parte 1. Configuración repositorios e integración continua.	Vence TP1 J. Enunciado TP2.		No determinado aún	Ver bibliografía en datos generales de la materia

Semana	Temas de teoría	Resolución de problemas	Laboratorio	Otro tipo	Fecha entrega Informe TP	Bibliografía básica
<12> 25/05 al 30/05	Recuperatorio	Role play MVC. Control de versiones.			No determinado aún	Ver bibliografía en datos generales de la materia
<13> 01/06 al 06/06	Programación e Ingeniería de Software. Metodología. UX.	Java FX parte 2	Entrega incremental TP2		No determinado aún	Ver bibliografía en datos generales de la materia
<14> 08/06 al 13/06	Clase de recuperación o a cargo de invitados	Clase de recuperación o a cargo de invitados	Entrega incremental TP2		No determinado aún	Ver bibliografía en datos generales de la materia
<15> 15/06 al 20/06	Concurrencia. Recapitulación.	Persistencia	Entrega incremental TP2		No determinado aún	Ver bibliografía en datos generales de la materia
<16> 22/06 al 27/06	Clase de recuperación o a cargo de invitados	Retrospectiva	Entrega incremental TP2		No determinado aún	Ver bibliografía en datos generales de la materia

CALENDARIO DE EVALUACIONES

Evaluación Parcial

Oportunidad	Semana	Fecha	Hora	Aula
1º	9	30/04		
2º	12	21/05		
3º	16	18/06		
4º				

Observaciones sobre el Temario de la Evaluación Parcial
El examen parcial es escrito y en papel. No se evalúan las destrezas en programación sobre computadora, que están cubiertas por los trabajos prácticos. El mismo evalúa las habilidades de modelado orientado a objetos (UML, pruebas unitarias) y algunas cuestiones conceptuales mediante preguntas teóricas. En algunos casos, se pedirá mejorar una solución que contenga algún problema de diseño o conceptual.

Otras observaciones
El horario de los parciales depende del curso.