Programa del curso

Diseño y Programación Orientada a Objetos

Versión PDF: Click Aquí

Nombre del curso:	Diseño y Programación Orientada a Objetos
Créditos:	3
Profesores:	• Germán Romero • John Casallas • Nicolás Cardozo • Mario Sánchez

Descripción

En este curso los estudiantes desarrollarán habilidades para diseñar e implementar software flexible y mantenible, haciendo explícitas las decisiones de diseño y las metodologías involucradas. El curso se concentra en el diseño orientado a objetos, pero se busca que las habilidades que se desarrollen sean útiles también para el trabajo con otros paradigmas o incluso en otras disciplinas de la ingeniería. En particular, será muy importante la argumentación alrededor de las decisiones de diseño que se tomen, el análisis de las ventajas y desventajas de cada decisión, y el uso de principios y patrones reconocidos. En este curso los estudiantes diseñarán programas y librerías y se enfrentarán a requerimientos cambiantes durante el proceso. En la mayoría de los casos, los procesos no terminarán con el diseño, sino que irán hasta las fases de implementación y pruebas. Para lograr esto, los estudiantes aprenderán a usar un lenguaje con un soporte sólido para la programación orientada a objetos (Java), en el cual se puedan apreciar las ventajas que trae este paradigma de programación para hacer mucho más sencillo el desarrollo de programas complejos.

Objetivos

Al final del curso los estudiantes:

1. Serán capaces de **diseñar** software (programas y librerías) para solucionar un problema:

- Partiendo del **análisis** para identificar los **conceptos del dominio** y su estructura, los **requerimientos** que deben implementarse y las **restricciones** con las que debe cumplir la solución.
- Siguiendo un **proceso iterativo** en el que apliquen estrategias de descomposición estructural y funcional en varios niveles de abstracción, practiquen el diseño de estructuras de datos, y concluyan con una validación del diseño contra requerimientos y restricciones.

- Utilizando los conceptos propios de la **programación orientada a objetos**.
- Adoptando algún ***framework*** y teniendo en cuenta los **principios** detrás de su diseño y los **compromisos** que implica.
- Incluyendo el **diseño de las pruebas** dentro del proceso global de diseño.

2. Serán capaces de **comunicar y explicar** el diseño de un programa o librería:

- En un **documento** que presente sus componentes y las relaciones estáticas y dinámicas entre ellos, usando artefactos y herramientas estándar (diagramas UML) y no estándar (bocetos de interfaces, diagramas de contexto).
- **Justificando las decisiones** que se hayan tomado en términos de características deseadas del software, de restricciones del problema, y de la aplicación de principios de diseño, explicando las ventajas, desventajas e implicaciones de cada alternativa.

3. Serán capaces de **leer críticamente** un diseño:

- Preparado por ellos o por alguien más.
- **Evaluándolo** y **analizando** las decisiones tomadas, para eventualmente corregirlo o mejorarlo.

4. Serán capaces de **implementar** un diseño:

- Utilizando con destreza un lenguaje de programación orientado a objetos.
- Para entender las consecuencias de las decisiones de diseño que se hayan tomado.

Metodología

En este curso la teoría es muy importante, pero sólo a través de la práctica se pueden desarrollar las habilidades planteadas dentro de los objetivos del curso. Para lograr esto, el curso combina la teoría y la práctica permanentemente: en casi todas las semanas habrá una sesión "teórica", en la que se estudiarán conceptos claves para el curso, y una sesión "práctica" (los talleres) en la que se pondrán en práctica esos mismos conceptos y otros que también estén relacionados. En el plan de actividades detallado se listan las actividades que se harán en cada una de las 32 sesiones del curso. Para **algunas** de las sesiones se presentan unas lecturas previas (no son sólo lecturas, también son videos y sitios web) : se espera que los estudiantes hayan hecho esas lecturas antes de la clase. Esto hará mucho más valiosa la sesión de clase y permitirá aprovechar mejor el tiempo. En varias sesiones se listan también lecturas complementarias: estas se pueden hacer antes o después de la sesión, no son obligatorias, pero tienen como principal objetivo complementar la sesión de clase y dar mucha más información sobre los temas que se están estudiando. Las lecturas estarán disponibles a través de Brightspace pero en casi todos los casos se encuentran disponibles en línea libremente o a través de la página de la biblioteca. La mayoría de los talleres serán evaluados: algunos serán individuales y otros se podrá hacer en parejas o en grupos de 3. En los talleres que no sean individuales se espera que todos los miembros del grupo participen y puedan defender cualquiera de los artefactos producidos. La principal razón para que los talleres no sean todos individuales es que todos tengan un interlocutor con el cual discutir las ideas y

problemas que surjan durante el desarrollo del taller. Las parejas y grupos pueden cambiar durante el semestre. Durante el curso se desarrollará un proyecto en 3 partes y cada parte tendrá una o más entregas (ver plan de actividades detallado). El proyecto deberá desarrollarse en parejas que deberán permanecer sin cambios durante todo el semestre. Cualquiera de los miembros de una pareja debería ser capaz de defender o explicar cualquier elemento de su proyecto.

Cronograma de actividades

No. semana	No. clase	Título de la actividad
1	1	Introducción al curso y al diseño O.O.
	2	Introducción a Java
	3	Conceptos centrales de OO
2	4	Taller 1: Arreglos, colecciones y asociaciones
	5	Más relaciones entre clases
3	6	Taller 2: Herencia, polimiformismo y genericidad
	7	Análisis y el modelo del mundo del problema
4	8	Taller 3: Análisis de dominio
	9	El proceso de diseño
5	10	Taller 4: Aspectos básicos del diseño de un programa
	11	Conceptos claves del diseño O.O.: roles, responsabilidades y colaboraciones
6	12	Taller 5: Roles, responsabilidades y colaboraciones

7	13	Mantenibilidad y flexibilidad
	14	Examen #1
	15	Interfaces gráficas
8	16	Taller 6a: GUIs con Swing
		SEMANA DE RECESO
	17	MVC - Modelo Vista Controlador
9	18	Taller 6b: GUIs con Swing
	19	Principios de diseño
10	20	Taller 7: Principios de diseño
	21	Patrones de diseño
11	22	Taller 8: Patrones
	23	Librerías y APIs
12	24	Taller 9: Diseño de APIs
	25	Estrategias para el manejo de errores
13	26	Taller 10: Excepciones
	27	Diseño de pruebas unitarias
14	28	Taller 11: Implementación de Pruebas unitarias
	29	Diseño de pruebas
15	30	Taller 12: Diseño de Pruebas
	31	Métricas de calidad en el diseño
16	32	Examen #2

Evaluación

- Examen 1 ..... 24%
- Examen 2 ..... 24%
- Proyecto 1..... 12%
- Proyecto 2..... 12%
- Proyecto 3..... 16%
- Talleres..... 12%

[tab title="Reglas de juego"]

Reglas de Juego

- Para la evaluación de los talleres se podrán realizar sustentaciones a discreción del profesor o del monitor del curso. En el caso de talleres que no se hayan realizado individualmente, el profesor o el monitor podrán seleccionar qué miembro del grupo deberá hacer la sustentación.
- La evaluación de los proyectos seguirá las mismas reglas que la evaluación de los talleres (podrían tener una sustentación). La nota obtenida en la sustentación se aplicará a todos los miembros del grupo, así no hayan participado en la sustentación.
- Los grupos del proyecto no pueden cambiar durante el semestre salvo casos de fuerza mayor.
- Los exámenes serán individuales y se realizarán durante la hora de clase. Los exámenes combinarán la teoría y la práctica.
- Es responsabilidad de cada estudiante tener un ambiente de trabajo completo y funcional para realizar todas las actividades del curso.

Bibliografía

[1] Apéndice II (Speaking the language of OO): D. McLaughlin, G. Pollice, D. West, Head-First Object-Oriented Analysis and Design, O'Reilly, 2006. [2] Capítulo 7 (Inheritance): M. P. Robillard, Introduction to Software Design with Java, Spring, 2017. [3] Capítulo 2 (Polymorphism): E. Sciore, Java Program Design, Principles, Polymorphism, and Patterns, Apress, 2019. [4] Capítulo 3 (Class Hierarchies): E. Sciore, Java Program Design, Principles, Polymorphism, and Patterns, Apress, 2019. [5] J. Villalobos, Modelos y Metamodelos, 2020. [6] Capítulo 1 (Introducción): M. P. Robillard, Introduction to Software Design with Java, Spring, 2017. [7] Capítulo 1 (Modular Software Design): E. Sciore, Java Program Design, Principles, Polymorphism, and Patterns, Apress, 2019. [8] M. Ubl, Design Docs at Google, https://www.industrialempathy.com/posts/design-docs-at-google/ , 2020. [9] R. Wirfs-Brock y A. McKean, Object Design: Roles, Responsibilities and Collaborations, 2002. Capítulos 3 al 6, especialmente las siguientes secciones: A Discovery Strategy (p. 78-79), What's in a name? (p. 88-93), What are responsbilities? (p. 110-111), Where Do Responsibilities Come From (p. 111-125), What is Object Collaboration? (p. 150-152), Collaboration Options (p. 153-158), What is Control Style? y Control Style Options (p.

196-198), Making Trade-offs (p. 198-205). [10] Capítulo 10 (Classes): R. C. Martin et al, Clean Code, A Handbook of Agile Software Craftsmanship, Prentice Hall, 2009 [11] Capítulo 5 (Construcción de la Interfaz Gráfica): J. Villalobos, Fundamentos de Programación - Aprendizaje Activo Basado en Casos, 2005. https://cupi2.virtual.uniandes.edu.co/libro-del-curso-pdf [12] M.P. Robillard and K. Kutschera. Lessons Learned in Migrating from Swing to JavaFX. IEEE Software, 2019. Reports on the experience of migrating the GUI to JavaFX. [13] M. Fowler, The Many Meanings of Event-Driven Architecture, https://www.youtube.com/watch?v=STKCRSUsyP0 [14] Capítulo 11 (Model, View, Controller): E. Sciore, Java Program Design, Principles, Polymorphism, and Patterns, Apress, 2019. [15] Capítulo 8 (Originality is Overrated): D. McLaughlin, G. Pollice, D. West, Head First Object-Oriented Analysis and Design, O'Reilly, 2006. [16] Capítulo 1 (Introduction): Design Patterns Elements of Reusable Object-Oriented Software by Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides [17] Capítulo 1 (Intro to Design Patterns): Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra, Head-First Design Patterns, O'Reilly. [18] Mel Ó Cinnéide and Paddy Fagan. 2006. Design patterns: the devils in the detail. In Proceedings of the 2006 conference on Pattern languages of programs (PLoP '06). Association for Computing Machinery, New York, NY, USA, Article 33, 1–9. [19] Capítulo 1 (APIs: What Are They?): S. Preibisch, API Development. Apress, Berkeley, CA, 2018. [20] J. Bloch. 2006. How to design a good API and why it matters. In Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications (OOPSLA '06). Association for Computing Machinery, New York, NY, USA, 506–507. DOI: https://doi.org/10.1145/1176617.1176622 [21] J. Bloch, How to Design a Good API and Why it Matters, 2007, http://www.newt.com/java/GoodApiDesign-JoshBloch.pdf [22] Capítulo 2 (API Design Best Practices): F. Doglio, REST API Development with Node.js. Apress, Berkeley, CA, 2018. [23] Capítulo 7 (Error Handling): R. C. Martin et al, Clean Code, A Handbook of Agile Software Craftsmanship, Prentice Hall, 2009 [24] Capítulo 5 (Unit Testing) - sin la sección 5.4: M. P. Robillard, Introduction to Software Design with Java, Spring, 2017. [25] Capítulo 3 (Fundamentals of Software Testing): G. O–Regan, Concise Guide to Software Testing. Undergraduate Topics in Computer Science. Springer, Cham, 2019. [26] Capítulo 2 (Software Testing): T.A. Majchrzak, Improving Software Testing. SpringerBriefs in Information Systems. Springer, Berlin, Heidelberg, 2012. [27] Ahmed Abd Elhalim Ibrahim, Amr Kamel, and Hesham Hassan. Object Oriented Metrics and Quality Attributes: A Survey. In Proceedings of the 10th International Conference on Informatics and Systems (INFOS '16). Association for Computing Machinery, New York, NY, USA, 312–319, 2016. [28] Seidl M., Scholz M., Huemer C., Kappel G. (2015) The Class Diagram. In: UML @ Classroom. Undergraduate Topics in Computer Science. Springer, Cham. [tab title="Aproximación"]

Reglas de aproximación

En este curso las calificaciones definitivas serán de uno cinco (1,5) a cinco (5,0), usando la siguiente escala de aproximación:

De 0 a 1,74	1,5
De 1,75 a 2,24	2,0

De 2,25 a 2,99	2,5
De 3,00 a 3,24	3,0
De 3,25 a 3,74	3,5
De 3,75 a 4,24	4,0
De 4,25 a 4,74	4,5
De 4,75 a 5,0	5,0