Instituto de
Tecnología e Ingeniería

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

Programación con objetos II

Carrera: Licenciatura en Informática

Actividad curricular: Programación con Objetos II

Área: Algoritmos y lenguajes

Prerrequisitos: Programación con Objetos I

Carga Horaria:

- Carga horaria total 96 horas
- Carga horaria práctica: 72 hs
  - Formación Experimental: 8 hs
  - Resolución de problemas: 64 hs
- Carga horaria semanal: 6 horas por semana

Objetivos:

Que el estudiante:

- Dar continuidad al aprendizaje del modelo de programación con objetos iniciado en la materia previa.
- Incorpore técnicas que le permitan aprovechar distintas características de la orientación a objetos y la idea de diseño del software.
- Comprenda las consecuencias de trabajar en un entorno con chequeo estático de tipos.
- Adquiera habilidades para analizar las diferencias en el desarrollo de aplicaciones complejas.
- Desarrolle habilidades para la organización de los proyectos utilizando módulos, subsistemas y/o paquetes.
- Realice una formación experimental intensiva sobre un entorno de desarrollo integrado (IDE) utilizado en la industria.

Instituto de
Tecnología e Ingeniería

Contenidos mínimos:

- Aproximación al diseño de software. Noción de decisión de diseño, el diseño como 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. Características deseadas en un diseño de objetos. Patrones de diseño. Nociones sobre proceso de diseño. Eventos. Metaprogramación. Uso de un entorno integrado de software. Notación UML de los diagramas de clases, de objetos y de secuencia. Testeo unitario y automático. Manejo de errores, impacto del manejo de errores en el diseño.

Programa analítico:

Unidad 1: Introducción al diseño Orientado a Objetos.

Abstracción, modelos, El diseño orientado a objetos. Lenguaje UML. Clases. Relaciones de Herencia, Colaboración, Dependencia e Implementación. Cardinalidad y navegabilidad. Cohesión y Acoplamiento. Principios SOLID

Unidad 2: Diseño orientado a objetos como proceso

Diferentes formas de descripción del dominio. Casos de uso, user stories. 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

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. Tipos de herencia. Jerarquía. Generalización y especialización. Method Lookup. Herencia contrapuesta a la composición. Protocolo de mensajes. Objetos de diferentes clases con tipos similares. Polimorfismo: ventajas en el uso. Polimorfismo como herramienta de abstracción.

Unidad 5: Patrones

Conceptos. 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

Instituto de
Tecnología e Ingeniería

Tipos de tests. Tests funcionales. Test de unidad. Estructura de un test de unidad: set up, ejecución, verificación y tear down. Inputs indirectos, outputs indirectos. Test dobles. 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. Evolución de los frameworks

Bibliografía obligatoria:

- Designing Object-Oriented Software. Rebecca Wirfs-Brock, Brian Wilkerson (Contributor), Lauren
- Introduction to Object-Oriented Programming, An (3rd Edition), Thimoty Budd,Addison Wesley; 3 edition (2001), ISBN-10: 0201760312
- The Object-Oriented Thought Process, Matt Weisfeld, Third Edition, Pearson Education, Addison Wesley. ISBN-13: 978-0- 672-33016-2
- Design Patterns. Elements of Reusable Objects Oriented Software. Garnma, Helm, Johnson, Vlissides, Addison-Wesley, Professional Computing Series.
- Xunit Test Patterns: Refactoring Test Code, Gerard Meszarons. Addison-Wesley Signature Series.
- Test Driven Development: A Practical Guide. David Astels. Coad Series.

Bibliografía de consulta:

- The Java Language Specification, Third Edition. James Gosling, Bill Joy, Guy Steele, Gilad Bracha.
- UML gota a gota. Fowler Martin, Scott Kendall, ADDISON-WESLEY IBEROA. Edición 1999 ISBN 9684443641

- El lenguaje unificado de modelado. Booch Grady, Jaboson Ivar, Rumbaugh James. ADDISON-WESLEY IBEROA, Edición 2000, ISBN 8478290281
- Szyperski, Clemens. "Components vs. objects vs. component objects." Proceedings of OOP. Vol. 1999. 1999.
- Fayad, Mohamed, and Douglas C. Schmidt. "Object-oriented application frameworks."
- M. E. Fayad, D. C. Schmidt, and R. E. Johnson, Building Application Frameworks: Object-oriented

Organización de las clases:

Se combinarán clases teóricas con clases prácticas con ejercitación directa sobre PC en salas de informática o en laboratorio.

En función del tema y de la intensidad del mismo, las actividades prácticas serán de tipo:

- Resolución de problemas: 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. Se brindarán guías de trabajos prácticos para facilitar este proceso.
- Formación experimental: para profundizar los conceptos vistos en las clases teóricas y evaluar mediante pruebas mediante el uso del IDE. Estos ejercicios serán de tipo "laboratorio", de mayor envergadura que los ejercicios prácticos. Requieren mayor tiempo en el análisis, discusión y resolución. Estos proyectos están diseñados para realizarse en clase durante el horario de trabajos prácticos.

Semana	Tema/unidad	Actividad		
		Teórico	Práctica	
			Res Prob.	Form. Exp.
1	Introducción: UML / I Diseño	x		
2	Continuación proceso de diseño	x	x	
3	Integración desde la Prog. O. Objetos	x	x	
4	Intorducción y Manejo de tipos en Java		x	
5	Gestión del cambio	x	x	

6	Clases Abstractas / Herencia / SOLID	x	x	
7	Formación Experimental - Java			X
8	Primer Parcial			
9	Patrones de Diseño / Patrones de comportamiento	x		
10	Formación Experimental - Java			X
11	Patrones estructurales	x		
12	Formación Experimental - Java	x		X
13	Tests	x	x	X
14	Segundo Parcial / Seguimiento del TGAI			
15	Conceptos avanzados	x	x	x
16	Recuperatorios			

Uso del campus virtual.

El Campus Virtual es un espacio fundamental para el desarrollo de la asignatura. En el aula virtual se propondrá material educativo, apuntes de clase, bibliografía así como también el programa y cronograma de la asignatura y las guías de Trabajos Prácticos y ejercicios.

Modalidad de evaluación:

Consistirá en dos exámenes parciales con recuperatorios, según el cronograma previsto, de la totalidad de la materia descripta en el programa. Los mismos se realizarán en las fechas que establezcan en el cronograma correspondiente. Se contemplará también como nota de evaluación el trabajo prácitco final de implementación.

- 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.