UNIVERSIDAD NACIONAL DEL SUR BAHÍA BLANCA	1/5
DEPARTAMENTO DE CIENCIAS E INGENIERÍA DE LA COMPUTACIÓN	
TECNOLOGÍA DE PROGRAMACIÓN	**CÓDIGO: 7951** **ÁREA N°: I**

CARRERAS
Licenciatura en Ciencias de la Computación
Ingeniería en Sistemas de Información

PROFESOR RESPONSABLE:

Dr. Martín Larrea – Profesor Adjunto con Dedicación Exclusiva

CARGA HORARIA	Teoría	Práctica	Laboratorio	CANTIDAD DE SEMANAS	16
	64	32	32		

CORRELATIVAS

PARA CURSAR LA MATERIA		PARA APROBAR LA MATERIA	
APROBADAS	**CURSADAS**	**APROBADAS**	**CURSADAS**
Introducción a la Programación Orientada a Objetos	Estructura de Datos	Estructura de Datos	

DESCRIPCIÓN

Tecnología de Programación es la cuarta y última materia del bloque inicial de aprendizaje de programación. En ésta los alumnos adquieren un conocimiento más profundo sobre el paradigma de orientación a objetos, sus usos, beneficios y tecnologías asociadas. La materia permite articular los contenidos desarrollados en las materias del área de Programación con los temas que se abordarán a partir de tercer año en las asignaturas del área Desarrollo de Software.
La materia tiene tres objetivos primordiales:

- *introducir nociones elementales de ingeniería de software y el proceso de desarrollo.*
- *la relación del paradigma orientado a objetos con el área de ingeniería de software.*
- *el soporte de los conceptos centrales de orientación a objetos en diversos lenguajes de programación.*

El primer objetivo es mandatorio pues los alumnos completan en esta materia su formación en aspectos técnicos de programación y requieren conocimientos superiores sobre la administración del proceso general de desarrollo de software.
El segundo objetivo permite realizar un vínculo entre el camino realizado hasta el momento (el aprendizaje de la tarea de programar) con el camino a realizar en materias siguientes inmediatas que refieran al análisis y diseño de sistemas. Permite, además, valorizar el paradigma sobre el que han estado mayoritariamente entrenándose en materias anteriores.
El tercer objetivo permite ampliar el conocimiento de lenguajes de programación y destaca el hecho de que las ideas de orientación a objetos van más allá de algún lenguaje particular. Esto, además, enfatiza la visión del lenguaje como herramienta y no como un fín en sí mismo.
Son objetivos del curso también:

- *Capacitar a los alumnos para resolver problemas técnicos y/o de diseño recurrentes en la tarea de programar.* Se estudian patrones de diseño y frameworks orientados a objetos, junto con técnicas y formalismos usuales en la programación actual, como el tratamiento de excepciones, nociones de concurrencia y diseño de interfaces de usuario.
- *Entrenar a los alumnos en el auto-aprendizaje y en la lectura de artículos de investigación*

MC

científica, introduciendo temas de vanguardia en programación. En el transcurso del cuatrimestre los alumnos deben leer algunos artículos científicos, adecuados a su nivel de formación, sobre aspectos relacionados con temas de la materia. Estos artículos son tenidos en cuenta en las instancias de evaluación (exámenes parciales y final).

- *Entrenar a los alumnos en la gestión de proyectos de programación de pequeña o mediana escala utilizando tecnologías actuales en la industria del software.*

METODOLOGÍA DE ENSEÑANZA

En las clases teóricas se combinan métodos inductivos y deductivos para la presentación de los contenidos. En algunos temas se introducen conceptos, métodos y las herramientas que soportan estos métodos y luego se plantean aplicaciones concretas. En otros temas se introduce en primera instancia la especificación de un problema que motiva la presentación de métodos y herramientas adecuados para la resolución.

En ambos casos los contenidos desarrollados en las clases teóricas se retoman y refuerzan luego en las clases prácticas a través de problemas, ejercicios y proyectos. El diseño de estas actividades promueve un rol activo y desarrolla la capacidad de autoaprendizaje por parte de los alumnos. En particular los recursos provistos por los lenguajes de programación utilizados, no son presentados en detalle en clase, sino que el alumno debe aprenderlos en muchos casos de manera autónoma, aunque por supuesto puede consultar con los docentes.

Los proyectos tienen un rol protagónico en las clases prácticas y de laboratorio porque permiten aplicar lo contenidos conceptuales y sirven asimismo como mecanismo de evaluación. Se utilizan diversos lenguajes orientados a objetos, como Java, Eiffel o C#. Se profundiza el conocimiento sobre el Lenguaje de Modelado Unificado UML y se estudian herramientas gráficas que asisten en el diseño de la estructura del sistema a desarrollar.

La expectativa es profundizar capacidades generales como interpretar consignas, resolver problemas, elaborar informes y trabajar en grupo. También se pretende desarrollar capacidades más específicas como estructurar y documentar el código favoreciendo la legibilidad, utilizar herramientas de modelado, gestionar proyectos a través de la administración del tiempo y de los recursos.

En las clases prácticas se organizan intervenciones de profesionales del medio productivo que describen la aplicación concreta de los métodos y herramientas presentadas en la industria de software.

El programa de la materia, los trabajos prácticos, el mecanismo y cronograma de evaluación están disponibles en la página web de la materia, como así también las presentaciones de diapositivas utilizadas en las clases teóricas.

MU

UNIVERSIDAD NACIONAL DEL SUR BAHÍA BLANCA	3/5
DEPARTAMENTO DE CIENCIAS E INGENIERÍA DE LA COMPUTACIÓN	
TECNOLOGÍA DE PROGRAMACIÓN	CÓDIGO: 7951 ÁREA N°: I

MECANISMO DE EVALUACIÓN

Durante el cuatrimestre los alumnos deben rendir un examen parcial escrito y entregar proyectos de programación (usualmente cuatro o cinco). Para cursar es necesario aprobar todas estas instancias evaluativas. En caso de desaprobar el examen escrito debe aprobar el recuperatorio, en donde se evalúan los temas desaprobados.

El examen final regular consiste de un breve examen escrito con temas puntuales de teoría y algunos ejercicios de práctica, previamente seleccionados y establecidos en una guía para el alumno que se publica año a año. Adicionalmente deben entregar un proyecto final de programación de baja complejidad pero con tecnologías actuales, cuyo enunciado es publicado al finalizar el cuatrimestre. El enunciado del proyecto varía año a año.

El examen final libre consiste de dos exámenes escritos que incluyen todos los temas de teoría. Adicionalmente deben entregar un proyecto final de mediana complejidad con tecnologías actuales.

PROGRAMA SINTÉTICO

- Ingeniería de software. Proceso y ciclo de vida del software. Arquitectura y Diseño.
- Herramientas y técnicas para el modelado.
- Reutilización y extendibilidad.
- Confiabilidad.
- Conceptos avanzados de herencia.
- Componentes.
- Diseño reutilizable: patrones de diseño.
- Frameworks orientados a objetos.
- Comunicación hombre máquina.
- Sistemas con requerimientos especiales.
- Eventos. Excepciones. Concurrencia.

PROGRAMA ANALÍTICO

1. **Ingeniería de Software. Proceso y Ciclo de Vida del software.** Principios: calidad, confiabilidad, reusabilidad y extendibilidad. Procesos de desarrollo: tradicional o en cascada, iterativo, Proceso Unificado, Diseño Centrado en el Usuario. Arquitectura de un Sistema. Diseño. Nociones de administración y gestión de proyectos de software.
2. **Herramientas y técnicas para el modelado.** Lenguajes. Lenguaje de Modelado UML. Diagramas de Clases y Diagramas de Objetos. Diagrama de interacción. Herramientas

MU

gráficas para el modelado.

3. **Reusabilidad y Extendibilidad.** Herencia de tipos. Su rol en la reusabilidad y extensibilidad. Sistemas de tipos. Control de tipos. Polimorfismo paramétrico (o genericidad) y por inclusión. Clases genéricas. Casos de estudios: Java, Eiffel, C++, Smalltalk. Diagramas de Comportamiento.
4. **Confiabilidad.** Obtención y formalización de requerimientos. Validación. Aserciones. Fórmulas de correctitud: precondiciones, postcondiciones. Modelo cliente-servidor. Invariantes. Su rol en la calidad y confiabilidad de software. Excepciones. Conceptos básicos del manejo de excepciones. Modelos de manejadores de excepciones. Casos de estudio: Java, Eiffel. Presentación de los proyectos de programación.
5. **Conceptos avanzados de herencia.** Herencia de interface y herencia de implementación. Herencia concreta y Herencia abstracta. Herencia múltiple. Conflictos de la herencia múltiple: colisión de nombres y herencia repetida. Técnicas de los lenguajes para la resolución de estos conflictos. Clasificación de los distintos usos de la herencia. Su rol en la reusabilidad y extendibilidad. Herencia y aserciones. Casos de estudio: Java, C++, Eiffel, C#.
6. **Componentes.** Descomposición y modularidad. Componentes. Concepto y utilización. Casos de estudio: Java (Beans), C#.
7. **Diseño reutilizable: patrones de diseño.** Conceptos. Convenciones de descripción de patrones. Catálogo de patrones de diseño: estudio específico de los patrones creacionales, estructurales y de comportamiento. Implementación de los patrones en lenguajes orientados a objetos. Refactoring de código y diseño. Principios de diseño SOLID.
8. **Implementación: Concurrencia.** Concepto de Concurrencia. Primitivas de lenguajes y plataformas para el desarrollo de objetos concurrentes.Casos de estudio: Java, C#.
9. **Implementación: frameworks orientados a objetos:** conceptos y propiedades de los frameworks. Beneficios. Uso y evolución de frameworks orientados a objetos. Casos de estudio: Java2.
10. **Implementación: comunicación hombre-máquina.** Guías para el diseño de interfaces. Ejemplos. Uso de librerías de componentes GUI.
11. **Implementación: sistemas con requerimientos especiales.** Sistemas de bases de datos. Sistemas de Tiempo Real. Eventos. Excepciones. Concurrencia. Sistemas con Recursos Limitados. Ejemplos.

BIBLIOGRAFÍA

Bibliografía Básica:

1. *Object Oriented Software Construction*. B. Meyer. ISE, Inc. ISBN 0-13-629155-4.
2. *Design Patterns. Elements of Reusable Object-Oriented Software*. E. Gamma, R. Helm, R. Johnson, J. Vlissides. ISBN 0-201-63361-2 (1995).
3. *Component Software: Beyond Object Oriented Programming*. Szyperski, C. Addison-Wesley. ISBN:0201745720

MLL

4. *The Unified Modeling Language Reference Manual*. James Rumbaugh, Ivar Jacobson, Grady Booch. Addison-Wesley (2000).
5. *Handbook of Object Oriented Technology*. CRC Press. ISBN 0-8493-3135-8.
6. *Core Java 2, Volumen 1 y 2*. Cay S. Horstmann, Gary Cornell. Prentice Hall, ISBN 0130894680 / 0131118269.
7. *Applied Java Patterns.* S Stelting, O. Maassen. Prentice Hall. ISBN 0-13-093538-7.
8. *Applying UML and Patterns*. Craig Larman, Prentice Hall.
9. *Advanced Techniques for Java Developers*. Daniel J. Berg, J. Steven Fritzinger. John Wiley & Sons Inc.
10. *Refactoring: Improving the Design of Existing Code*. Martin Fowler. Addison-Wesley Object Technology. ISBN-10: 0134757599 - ISBN-13: 978-0134757599
11. *The Art and Science of Smalltalk.* Simon Lewis. Prentice Hall.
12. *Eiffel: An Introduction.* Switzer, R.
13. *Object-Oriented Programming in Eiffel*. Rist, R.-Terwilliger, R.

Bibliografía Adicional:

14. *Core J2EE Patterns.* Prentice Hall. ISBN 0-13-066586
15. *Component Software: Beyond Object Oriented Programming*. Szyperski, C.
16. *UML Distilled : Applying the Standard Object Modeling Language. (Fowler, M.-Kendall Scott)*

AÑO	FIRMA PROFESOR RESPONSABLE
2019	MARTÍN LARREA

VISADO

COORDINADOR ÁREA	SECRETARIO ACADÉMICO	DIRECTOR DEPARTAMENTO
	Dr. DIEGO C. MARTINEZ SECRETARIO ACADÉMICO DPTO. DE CS. E ING. DE LA COMP. UNIVERSIDAD NACIONAL DEL SUR	Dr. MARCELO A. FALAPPA DIRECTOR DECANO DPTO. DE CS. E ING. DE LA COMP. UNIVERSIDAD NACIONAL DEL SUR