



GUÍA DOCENTE
Curso 2011-2012

Titulación:	Grado en Ingeniería Informática			Código :	801G
Centro:	FCEAI				
Dirección:	Edificio CCT C/Madre de Dios, 51			Código postal:	26006
Teléfono:	+34 941 299 607	Fax:	+34 941 299 611	Correo electrónico:	decanato.cai@unirioja.es
Director del Grado:	Ángel Luis Rubio García				
Teléfono:	+34 941 299 449	Correo electrónico:	direstudios.informatica@unirioja.es		
Despacho:	231	Edificio:	Vives		

Fdo.: Ángel Luis Rubio García

En Logroño a 1 de julio de 2011

GUÍA DOCENTE

Curso 2011-2012

Titulación:	Grado en Ingeniería Informática		801G
Asignatura:	Programación Orientada a Objetos		801205012
Materia:	Programación		
Módulo:	M3 Programación		
Carácter:	Obligatorio	Curso: 2º	Semestre: 1º
Créditos ECTS:	6	Horas presenciales: 60	Horas de trabajo autónomo estimadas: 90
Idiomas en los que se imparte:	Castellano		
Idiomas del material de lectura o audiovisual:	Castellano e inglés		

Departamentos responsables de la docencia:

Matemáticas y Computación		R111
Dirección:	Luis de Ulloa, s/n	Código postal: 26004
Teléfono:	+34 941 299 452	Fax: +34 941 299 460
	Correo electrónico:	dpto.dmc@unirioja.es

Profesores

Profesor responsable de la asignatura:	Julio Rubio García	
Teléfono:	+34 941 299 448	Correo electrónico: julio.rubio@unirioja.es
Despacho:	204	Edificio: Vives
Horario de tutorías:		
Nombre profesor:	Jesús María Aransay Azofra	
Teléfono:	+34 941 299 438	Correo electrónico: jesus-maria.aransay@unirioja.es
Despacho:	235	Edificio: Vives
Horario de tutorías:		
Nombre profesor:	Eduardo Sáenz de Cabezón Irigaray	
Teléfono:	+34 941 299 458	Correo electrónico: eduardo.saenz-de-cabezon@unirioja.es
Despacho:	213	Edificio: Vives
Horario de tutorías:		

Descripción de contenidos:

- Revisión y carencias de la programación estructurada.
- Clases, objetos, estado, métodos, modos de acceso.
- Herencia, subtipos, polimorfismo.
- Clases abstractas e interfaces.
- Lenguajes de programación orientada a objetos.
- Enlazado dinámico y estático.
- Diferencias y similitudes entre lenguajes que admiten orientación a objetos.

Requisitos previos:

Se aconseja tener competencias y conocimientos básicos en programación de computadores.

Relación de asignaturas que proporcionan los conocimientos y competencias requeridos:

Metodología de la Programación y Tecnología de la Programación

Contexto

La asignatura pertenece al módulo "Programación" propio de la titulación, al primer semestre del segundo curso. Dentro de dicho bloque, es la primera asignatura que deben realizar los alumnos. Sin embargo, en la asignatura se asume cierta familiaridad de los alumnos con algunas de las nociones básicas sobre programación, lenguajes de programación (en particular, C++) o implementación de tipos de datos. Esta familiaridad permitirá poner la asignatura en perspectiva y realizar una comparación más completa de los paradigmas de programación orientada a objetos y de programación estructurada. Por ejemplo, la asignatura explica las nociones de orientación a objetos tanto en C++ como en Java, asumiendo que los alumnos ya han trabajado previamente con C++. También hace uso de ciertos ejemplos de tipos abstractos de datos que se implementarán dentro del paradigma de orientación a objetos que se suponen ya conocidos por los alumnos.

Dentro del módulo "Programación", la asignatura tiene que ser capaz de dotar a los alumnos de las competencias necesarias para ser capaces de implementar un programa en Java o en C++ usando las nociones propias de orientación a objetos, a partir de un diseño ya dado. Por tanto, los alumnos deben aprender la sintaxis de los lenguajes de programación utilizados en la asignatura (Java y C++), conocer la noción de clase (atributos, métodos), utilizar modificadores de acceso, distinguir el estado y el comportamiento de los objetos, ser capaces de abstraer la noción de clase a partir de objetos de dicha clase, reconocer e identificar las relaciones más comunes entre clases (herencia, asociación, agregación), y en particular identificarlas a partir de un diagrama UML dado, entender la noción de polimorfismo y prever el comportamiento de un sistema de información que haga uso del mismo, distinguir las nociones de enlazado dinámico y estático y comprender su funcionamiento en Java y C++, declarar métodos abstractos, entender su utilidad, implementar clases abstractas e interfaces, implementar sistemas de información que contengan todos los elementos anteriores, y finalmente aprender a gestionar y definir excepciones en Java. Asignaturas posteriores dentro del mismo módulo se destinarán a enseñar a diseñar sistemas de información propios, dentro del paradigma de orientación a objetos, así como a programar interfaces de usuario.

Competencias:
Competencias generales

- | | |
|-----|--|
| CG2 | Estar capacitado para, utilizando el nivel adecuado de abstracción, establecer y evaluar modelos que representen situaciones reales. |
| CG3 | Estar capacitado para encontrar, relacionar, estructurar e interpretar datos, información y conocimiento provenientes de diversas fuentes. |
| CG7 | Haber desarrollado aquellas habilidades de aprendizaje necesarias para continuar su formación. |

Competencias específicas

- | | |
|-----|--|
| CE1 | Capacidad para concebir, redactar, organizar, planificar, desarrollar y firmar proyectos en el ámbito de la ingeniería en informática que tengan por objeto, la concepción, el desarrollo o la explotación de sistemas, servicios y aplicaciones informáticas. |
|-----|--|

CE4	Capacidad para definir, evaluar y seleccionar plataformas hardware y software para el desarrollo y la ejecución de sistemas, servicios y aplicaciones informáticas.
CE5	Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones informáticas empleando los métodos de la ingeniería del software como instrumento para el aseguramiento de su calidad.

Resultados del aprendizaje:

Abordar la solución de problemas aplicando el paradigma de la programación orientada a objetos.
Conocer y usar algunos lenguajes que faciliten la orientación a objetos y que sean de uso extendido.
Conocer y manejar distintos entornos y plataformas de desarrollo de software.
Saber determinar la adecuación o no del uso de distintas plataformas y/o lenguajes para resolver un problema de desarrollo de software.
Comprender el tratamiento de los conceptos de programación orientada a objetos en distintas plataformas y lenguajes de desarrollo.

Temario

Tema 1. Nociones de clase y objeto en programación orientada a objetos.

- 1.1 Representación de la información por medio de objetos
- 1.2 Atributos o estado
- 1.3 Métodos o comportamiento
- 1.4 Abstracción de objetos en clases
- 1.5 Necesidad y relevancia de los constructores de clase: constructor por defecto, constructores propios
- 1.6 Métodos de acceso y modificación del estado de un objeto
- 1.7 Atributos y métodos estáticos o de clase
- 1.8 Modificadores de acceso: relevancia y necesidad de los modificadores público y privado
- 1.9 Ocultación de la información: distintas formas de representar una misma clase manteniendo su comportamiento
- 1.10 Introducción al lenguaje de especificación UML: utilización para representar clases y objetos
- 1.11 Lenguaje de programación C++: declaración de clases y construcción de objetos
- 1.12 Lenguaje de programación Java: declaración de clases

Tema 2. Relaciones entre clases. Herencia entre clases.

- 2.1 Comunicación entre distintas clases
- 2.2 Clases que contienen objetos como atributos: algunos ejemplos conocidos
- 2.3 Relaciones de especialización/generalización
- 2.4 Definición de la relación de herencia entre clases
- 2.5 Ventajas del uso de relaciones de herencia: reutilización de código y polimorfismo de tipos de datos
- 2.6 Redefinición de métodos en clases heredadas
- 2.7 Modificador de uso "protegido": posibilidades de uso
- 2.8 Representación de relaciones de herencia en diagramas UML
- 2.9 Programación en Java y C++ de relaciones de herencia

Tema 3. Definición y uso de métodos polimorfos

- 3.1 Definición de polimorfismo y ventajas de uso
- 3.2 Obtención de polimorfismo en C++: utilización de memoria dinámica y métodos virtual
- 3.3 Polimorfismo en Java
- 3.4 Utilización de métodos polimorfos sobre ejemplos ya construidos

Tema 4. Clases abstractas e interfaces

- 4.1 Definición de métodos abstractos en POO. Algunos ejemplos de uso
- 4.2 Relación entre polimorfismo y métodos abstractos
- 4.3 Definición y ventajas de uso de clases completamente abstractas o interfaces
- 4.4 Representación en UML de métodos abstractos, clases abstractas e interfaces
- 4.5 Implementación en C++ de métodos abstractos y clases abstractas
- 4.6 Implementación en Java de métodos abstractos e interfaces

Tema 5. Excepciones en Java

- 5.1 Definición de excepciones en programación
- 5.2 Tipos de excepciones/errores y cómo tratarlos
- 5.3 Trabajando con excepciones: declaración, construcción, lanzamiento y gestión de excepciones
- 5.4 Programación de excepciones en Java. Utilización de excepciones de la librería y definición de excepciones propias

Bibliografía

Apuntes de la asignatura, disponibles a través de <http://www.unirioja.es/cu/jearansa>

Bruce Eckel, "Thinking in Java" (también disponible en castellano y en la página web del autor en formato HTML). Libro sobre Java que abarca tanto el lenguaje Java como las principales características de POO. Contiene ejercicios.

Bruce Eckel, Chuck Allison, "Thinking in C++" (también disponible en castellano y en la página web del autor en formato HTML). Libro sobre C++ que abarca la implementación de sistemas de información en POO usando C++.

James Rumbaugh, Ivar Jacobson, Grady Booch, "The unified modelling language reference manual" (también disponible en castellano). Libro sobre el lenguaje de especificación UML.

David Flanagan, "Java in a nutshell". Libro sobre Java con ejemplos de uso sobre las librerías propias del lenguaje.

Elliot B. Koffman, Paul A.T. Wolfgang, "Objects, Abstraction, Data Structures and design using Java". Libro sobre implementación y diseño de estructuras de datos en el lenguaje de programación Java, contiene interesantes ejemplos desarrollados de programación y uso de UML.

James Gosling, Bill Joy, Guy Steele, Gilad Bracha, "The Java Language Specification". Manual de referencia de Java escrito por alguno de los diseñadores y desarrolladores del lenguaje que contiene la sintaxis completa del lenguaje así como numerosos fragmentos de código que pueden resultar útiles.

Bjarne Stroustrup, "The C++ programming language". Manual de referencia sobre C++, aunque no está centrado en el uso de POO. Útil para resolver cualquier tipo de duda sobre el lenguaje.

Richard C. Lee, William M. Tepfenhart, "Practical object-oriented development with UML and Java". Libro sobre Java y desarrollo orientado a objetos. Especialmente claro en la explicación de las relaciones entre clases y su implementación.

<http://java.sun.com/reference/api/> Página de la API de Java (en sus distintas versiones). Interesante tanto para consultar las características de los elementos del lenguaje, como para encontrar ejemplos de uso de muchas de las características de la POO que se aprenden en el curso. En la página también es posible encontrar tutoriales y FAQ's útiles.

<http://www.cplusplus.com/> Página sobre C++ con documentación diversa sobre el lenguaje. Especialmente útil resulta su referencia sobre la librería de C++.

Metodología

Modalidades organizativas:	Métodos de enseñanza:
MO1: clases teóricas	ME1: lección magistral
MO3: clases prácticas	ME3: resolución de ejercicios y problemas
MO5: tutorías	ME4: utilización de recursos informáticos
MO6: estudio y trabajo autónomo del alumno	

Organización

Actividades presenciales:	Horas
- Clases teóricas	32
- Clases prácticas de laboratorio o aula informática	28
Total horas presenciales	60

Actividades no presenciales (trabajo autónomo):	Horas estimadas
- Estudio autónomo individual o en grupo	30
- Resolución individual de ejercicios, cuestiones u otros trabajos, actividades en biblioteca o similar	30
- Preparación de las prácticas y elaboración de cuaderno de prácticas	30
Total horas estimadas de trabajo autónomo	90
Total horas estimadas	150

Evaluación

Sistemas de evaluación: Común para todas las titulaciones donde se imparta la asignatura	% sobre total	Recuperable/ No Recuperable
SE1: Pruebas escritas	80	Recuperable
SE4: Informes/Memorias de prácticas	20	No Recupera.

Comentario:

Para los estudiantes a tiempo parcial (reconocidos como tales por la Universidad), las actividades de evaluación no recuperable podrán ser sustituidas por otras, a especificar en cada caso. Esta posibilidad se habilitará siempre y cuando la causa que le impida la realización de la actividad de evaluación programada sea la que ha llevado al reconocimiento de la dedicación a tiempo parcial.

Criterios críticos para superar la asignatura:

La asistencia, el aprovechamiento y la entrega de las prácticas serán obligatorios para superar la asignatura. Para superar la asignatura se deberá obtener al menos un 40% de la puntuación en la prueba escrita.