PROGRAMACIÓN ORIENTADA A OBJETOS
GUÍA DOCENTE CURSO 2014-15

Titulación:	Grado en Matemáticas					**701G**	
Asignatura:	Programación orientada a objetos					**828**	
Materia:	Informática						
Módulo:	Informática						
Carácter:	Obligatoria	**Curso:**	2	**Duración:**	Semestral		
Créditos ECTS:	6,00	**Horas presenciales:**	60,00	**Horas estimadas de trabajo autónomo:**	90,00		

Idiomas en que se imparte la asignatura:	Español
Idiomas del material de lectura o audiovisual:	Inglés, Español

DEPARTAMENTOS RESPONSABLES DE LA DOCENCIA

MATEMÁTICAS Y COMPUTACIÓN					**R111**
Dirección:	C/ Luis de Ulloa, s/n			**Código postal:**	26004
Localidad:	Logroño			**Provincia:**	La Rioja
Teléfono:	941299452	**Fax:**	941299460	**Correo electrónico:**	

PROFESORADO PREVISTO

Profesor:	Aransay Azofra, Jesús María			**Responsable de la asignatura**	
Teléfono:	941299438	**Correo electrónico:**	jesus-maria.aransay@unirioja.es		
Despacho:	235	**Edificio:**	EDIFICIO VIVES	**Tutorías:**	Consultar

DESCRIPCIÓN DE LOS 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 DE CONOCIMIENTOS Y COMPETENCIAS PARA PODER CURSAR CON ÉXITO LA ASIGNATURA

Recomendados para poder superar la asignatura.

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

Asignaturas que proporcionan los conocimientos y competencias:

- Metodología de la programación
- Tecnología de la programación

CONTEXTO

La asignatura corresponde al módulo “Informática” propio de la titulación, al primer semestre del segundo curso.
Dentro de dicho módulo se sitúa temporalmente detrás de las asignaturas “Metodología de la Programación” y “Tecnología de la programación”, ambas de primer curso, y además la asignatura supone cierta familiaridad de los alumnos con algunas de las competencias adquiridas en dichas asignaturas, como por ejemplo la capacidad para encontrar soluciones algorítmicas de problemas matemáticos y de aplicación sabiendo comparar distintas alternativas, según criterios de adecuación, complejidad y coste o el hecho de saber programar algoritmos de modo correcto y eficaz, eligiendo convenientemente lenguajes y plataformas de programación. También se suponen algunos de los resultados de aprendizaje de las asignaturas antes citadas; será importante para poder cursar la asignatura con aprovechamiento el conocer la sintaxis de algún lenguaje de programación imperativa (preferentemente que soporte un posterior enfoque orientado a objetos), o el haber desarrollado programas de tamaño medio y haber aprendido a usar los mecanismos de construcción de estructuras de datos para la representación y manejo de información.
Partiendo de esas competencias y asumiendo los resultados de aprendizaje reseñados, la asignatura debe capacitar a los alumnos para cursar la asignatura “Especificación y Desarrollo de Sistemas Software” (2º curso, 2º semestre), para lo cual se

dotará a los alumnos de las competencias para desarrollar programas de tamaño medio en diversos lenguajes de Programación Orientada a Objetos (POO), abstraer y representar estructuras de datos en lenguajes de POO o analizar y abordar problemas aplicando el paradigma de POO.

COMPETENCIAS

Competencias generales

CG6: Relacionar el conocimiento especializado de Matemáticas con el conocimiento general en el que se inserta y con las herramientas que utiliza cuando se aplica en diversas opciones profesionales, especialmente en el marco de las TIC.
CG7: Saber abstraer las propiedades estructurales de objetos de la realidad observada y de otros ámbitos, distinguiéndolas de aquellas puramente ocasionales, comprobando la aplicabilidad de las Matemáticas.
CG8: Capacitar para el aprendizaje autónomo de nuevos conocimientos y técnicas.

Competencias específicas

CE4: Encontrar soluciones algorítmicas de problemas matemáticos y de aplicación (de ámbito académico, técnico, financiero o social), sabiendo comparar distintas alternativas, según criterios de adecuación, complejidad y coste.
CE5: Saber programar algoritmos de modo correcto y eficaz, eligiendo convenientemente lenguajes y plataformas de programación.
CE6: Utilizar herramientas de búsqueda de recursos en Matemáticas, Informática y aplicaciones.

RESULTADOS DEL APRENDIZAJE

Abordar la solución de problemas aplicando el paradigmas 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

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

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

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

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

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

Tipo:	Título
Básica	Thinking in Java; Bruce Eckel (disponible también en versión en castellano) Absys
Básica	Thinking in C++; Bruce Eckel, Chuck Allison (disponible también en castellano) Absys
Básica	Objects first with Java : a practical introduction using BlueJ Absys
Básica	Java Generics and Collections; Maurice Naftalin and Philip Wadler Absys
Básica	Effective Java; Joshua Bloch Absys
Básica	The unified modelling language reference manual; James Rumbaugh, Ivar Jacobson, Grady Booch (disponible también en castellano) Absys
Complementaria	Design patterns: elements of reusable object-oriented software; Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides Absys
Complementaria	Objects, abstraction, data structures and design using Java; Elliot B. Koffman, Paul A.T. Wolfgang Absys
Complementaria	Practical object-oriented development with UML and Java; Richard C. Lee, William M. Tepfenhart Absys
Complementaria	Java in a nutshell; David Flanagan Absys
Complementaria	The C++ programming language; Bjarne Stroustrup Absys
Complementaria	The Java Language Specification; James Gosling, Bill Joy, Guy Steele, Gilad Bracha Absys
Complementaria	Java Examples in a Nutshell; David Flanagan Absys

Recursos en Internet
Los apuntes necesarios para el seguimiento de la asignatura estarán disponibles en el Aula Virtual de la misma http://campusvirtual.unirioja.es
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.
http://java.sun.com/reference/api

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

Información adicional sobre la asignatura como el horario de tutorías se podrán encontrar en la siguiente web
http://www.unirioja.es/cu/jearansa

METODOLOGÍA

Modalidades organizativas
Clases teóricas
Clases prácticas
Estudio y trabajo autónomo individual

Métodos de enseñanza
Método expositivo - Lección magistral
Resolución de ejercicios y problemas

ORGANIZACIÓN

Actividades presenciales	Tamaño de grupo	Horas
Clases prácticas de laboratorio o aula informática	Informática	28,00
Clases teóricas	Grande	32,00
Total de horas presenciales		60,00

Trabajo autónomo del estudiante	Horas
Estudio autónomo individual o en grupo	30,00
Preparación de las prácticas y elaboración de cuaderno de prácticas	30,00
Resolución individual de ejercicios, cuestiones u otros trabajos, actividades en biblioteca o similar	30,00
Total de horas de trabajo autónomo	90,00

Total de horas	150,00

EVALUACIÓN

Sistemas de evaluación	Recuperable	No Recup.
Pruebas escritas	70%	
Informes y memorias de prácticas		30%
Total	100%	

Comentarios
En el caso de estudiantes a tiempo parcial (reconocidos como tales por la Universidad) el profesor responsable de la asignatura podrá sustituir las actividades de evaluación no recuperable por otras a especificar en cada caso.
La evaluación continua (30%) se realizará mediante los sistemas de evaluación "Informes y memorias de prácticas"

Criterios críticos para superar la asignatura
Será imprescindible superar (obtener más de un 50% de la calificación) ambas partes de la evaluación (Pruebas escritas, e Informes y memorias de prácticas) para aprobar la asignatura