PROGRAMACIÓN ORIENTADA A OBJETOS
GUÍA DOCENTE CURSO 2019-20

Titulación:	Grado en Ingeniería Informática				**801G**
Asignatura:	Programación orientada a objetos				**828**
Materia:	Programación				
Módulo:	Programación				
Modalidad de enseñanza de la titulación:	Presencial		**Carácter:**	Obligatoria	
Curso:	2	**Créditos ECTS:** 6,00	**Duración:**	Semestral (Primer Semestre)	
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/ Madre de Dios, 53			**Código postal:**	26006
Localidad:	Logroño		**Provincia:**	La Rioja	
Teléfono:	941299452	**Fax:** 941299460	**Correo electrónico:**	dpto.dmc@unirioja.es	

PROFESORADO PREVISTO

Profesor:	Aransay Azofra, Jesús María			**Responsable de la asignatura**	
Teléfono:	941299438	**Correo electrónico:**	jesus-maria.aransay@unirioja.es		
Despacho:	3245	**Edificio:**	CENTRO CIENTÍFICO TECNOLÓGICO	**Tutorías:**	Consultar

Profesor:	Heras Vicente, Jónatan				
Teléfono:	941299445	**Correo electrónico:**	jonathan.heras@unirioja.es		
Despacho:	3210	**Edificio:**	CENTRO CIENTÍFICO TECNOLÓGICO	**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 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 estudiantes. Sin embargo, en la asignatura se asume cierta familiaridad de los estudiantes 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 estudiantes 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 estudiantes.
Dentro del módulo “Programación”, la asignatura tiene que ser capaz de dotar a los estudiantes 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 estudiantes 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.
CG8-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.
CG11-Capacidad para definir, evaluar y seleccionar plataformas hardware y software para el desarrollo y la ejecución de sistemas, servicios y aplicaciones informáticas.
CG12-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.

Competencias específicas

CE7-Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
CE12-Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.
CE13-Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema.
CE14-Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
CE28-Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.

RESULTADOS DEL APRENDIZAJE

Abordar la solución de problemas aplicando el paradigma de la programación orientada a objetos.
- Comprender el procesamiento de información con estado y sin estado.
- 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 interface
4.7 Java Generics
4.8 Java Collections
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
Básica	Java y C++ : paso a paso 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	Java in a nutshell; David Flanagan Absys
Complementaria	Practical object-oriented development with UML and Java; Richard C. Lee, William M. Tepfenhart 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
Complementaria	Effective C++

Recursos en Internet
Los apuntes necesarios para el seguimiento de la asignatura estarán disponibles en el Aula Virtual de la misma https://unirioja.blackboard.com
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.
https://docs.oracle.com/en/java/javase/11/

Página sobre C++ con documentación diversa sobre el lenguaje. Especialmente útil resulta su referencia sobre la librería de C++.
https://en.cppreference.com

Información adicional sobre la asignatura como el horario de tutorías se podrán encontrar en la siguiente web
https://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
Para los estudiantes a tiempo parcial (reconocidos como tales por la Universidad), los apartados de evaluación no recuperable podrán ser sustituidos por otros, a especificar en cada caso.
La evaluación continua (30%) se realizará mediante los sistemas de evaluación "Informes y memorias de prácticas".
Para superar la parte de evaluación continua se considerará la asistencia con participación y el aprovechamiento de las clases, tanto teóricas como prácticas de aula informática.
La evaluación final se corresponderá con las actividades de evaluación recuperables (70%).

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.