1. Identificación

1.1. De la Asignatura

Curso Académico	2018/2019
Titulación	GRADO EN INGENIERIA INFORMÁTICA y PROG CONJUNTA DE ESTUDIOS OFICIALES GRADO MATEMÁTICAS Y GRADO ING. INFORMÁTICA
Nombre de la Asignatura	PROGRAMACIÓN ORIENTADA A OBJETOS
Código	1897
Curso	SEGUNDO y SEGUNDO(IC)
Carácter	OBLIGATORIA
N.º Grupos	4
Créditos ECTS	6
Estimación del volumen de trabajo del alumno	150
Organización Temporal/Temporalidad	1º Cuatrimestre y 1º Cuatrimestre(IC)
Idiomas en que se imparte	ESPAÑOL
Tipo de Enseñanza	Presencial

1.2. Del profesorado: Equipo Docente

Coordinación de la asignatura BEGOÑA MOROS VALLE	Área/Departamento	INFORMÁTICA Y SISTEMAS
	Categoría	PROFESORES TITULARES DE UNIVERSIDAD
	Correo Electrónico / Página web / Tutoría electrónica	bmoros@um.es Tutoría Electrónica: SÍ

Grupo de Docencia: 1, 2, 3 y 9 Coordinación de los grupos:2 y 3	Teléfono, Horario y Lugar de atención al alumnado	Duración	Día	Horario	Lugar
		Primer Cuatrimestre	Lunes	09:30- 11:30	868884608, Facultad de Informática B1.2.047
		Primer Cuatrimestre	Martes	17:45- 18:45	868884608, Facultad de Informática B1.2.047
		Primer Cuatrimestre	Martes	17:45- 18:45	868884608, Facultad de Informática B1.2.047
		Segundo Cuatrimestre	Martes	10:00- 13:00	868884608, Facultad de Informática B1.2.047
MARCOS MENARGUEZ TORTOSA Grupo de Docencia: 1, 2, 3 y 9 Coordinación de los grupos:1 y 9(IC)	Área/Departamento	INFORMÁTICA Y SISTEMAS			
	Categoría	PROFESOR CONTRATADO DOCTOR TIPO A (DEI)			
	Correo Electrónico / Página web / Tutoría electrónica	marcos@um.es Tutoría Electrónica: SÍ			
	Teléfono, Horario y Lugar de atención al alumnado	Duración	Día	Horario	Lugar
		Segundo Cuatrimestre	Lunes	10:00- 13:00	868884234, Facultad de Informática B1.2.043
JUAN JOSE VERA GUIRAO	Área/Departamento	INFORMÁTICA Y SISTEMAS			
	Categoría	ASOCIADO A TIEMPO PARCIAL			

<table>
<tr><td rowspan="2">Grupo de Docencia: 1, 2, 3 y 9</td><td>Correo Electrónico / Página web / Tutoría electrónica</td><td colspan="5">jnjsvera@um.es
Tutoría Electrónica: SÍ</td></tr>
<tr><td>Teléfono, Horario y Lugar de atención al alumnado</td><td>Duración</td><td>Día</td><td>Horario</td><td>Lugar</td><td>Observaciones</td></tr>
<tr><td></td><td></td><td>Primer Cuatrimestre</td><td>Viernes</td><td>19:00- 20:30</td><td>868888528, Facultad de Informática B1.2.029</td><td>Solicitar tutoría por mensaje privado del Aula Virtual.</td></tr>
<tr><td rowspan="6">FRANCISCO JAVIER BERMUDEZ RUIZ
Grupo de Docencia: 1, 2, 3 y 9</td><td>Área/Departamento</td><td colspan="5">INFORMÁTICA Y SISTEMAS</td></tr>
<tr><td>Categoría</td><td colspan="5">PROFESOR COLABORADOR (LICENCIADO)</td></tr>
<tr><td>Correo Electrónico / Página web / Tutoría electrónica</td><td colspan="5">fjavier@um.es
Tutoría Electrónica: SÍ</td></tr>
<tr><td rowspan="5">Teléfono, Horario y Lugar de atención al alumnado</td><td>Duración</td><td>Día</td><td>Horario</td><td colspan="2">Lugar</td></tr>
<tr><td>Primer Cuatrimestre</td><td>Martes</td><td>12:00- 13:30</td><td colspan="2">868888524, Facultad de Informática B1.2.038</td></tr>
<tr><td>Primer Cuatrimestre</td><td>Jueves</td><td>12:00- 13:30</td><td colspan="2">868888524, Facultad de Informática B1.2.038</td></tr>
<tr><td>Segundo Cuatrimestre</td><td>Martes</td><td>11:30- 13:00</td><td colspan="2">868888524, Facultad de Informática B1.2.038</td></tr>
<tr><td>Segundo Cuatrimestre</td><td>Jueves</td><td>11:30- 13:00</td><td colspan="2">868888524, Facultad de Informática B1.2.038</td></tr>
</table>

JESUS SANCHEZ CUADRADO Grupo de Docencia: 1	Área/Departamento	INFORMÁTICA Y SISTEMAS				
	Categoría	INVESTIGADOR "RAMON Y CAJAL"				
	Correo Electrónico / Página web / Tutoría electrónica	jesusc@um.es Tutoría Electrónica: SÍ				
	Teléfono, Horario y Lugar de atención al alumnado	Duración	Día	Horario	Lugar	Observaciones
		Primer Cuatrimestre	Miércoles	10:00- 11:30	868887981, Facultad de Informática B1.2.027	Despacho 2.21
		Primer Cuatrimestre	Jueves	11:00- 12:30	868887981, Facultad de Informática B1.2.027	Despacho 2.21

2. Presentación

La asignatura Programación Orientada a Objetos (POO) completa la formación relativa a los paradigmas de programación estudiados en primero, introduciendo los conceptos básicos del paradigma Orientado a Objetos (OO). Dichos conceptos se presentan desde la programación puesto que consideramos que sólo dominando estos conceptos se pueden aplicar correctamente en el análisis y diseño de programas. Además, asumimos que la mejor manera de dominar los conceptos del paradigma OO es poniéndolos en práctica en pequeños proyectos de programación. No obstante, la enseñanza de la POO no se reduce a la enseñanza de la sintaxis de un lenguaje de programación particular, sino que la materia se organiza alrededor de objetivos conceptuales más estables, los conceptos básicos del modelo OO.

3. Condiciones de acceso a la asignatura

3.1 Incompatibilidades

No consta

3.2 Recomendaciones

Es recomendable que el alumno haya superado las asignaturas previas relacionadas con la programación de primer curso: Introducción a la Programación y Tecnología de la Programación.

4. Competencias

4.1 Competencias Básicas

No disponible

4.2 Competencias de la titulación

· CGII16. Aprendizaje autónomo.
· CGII17. Adaptación a nuevas situaciones.
· CGUM6. Capacidad para trabajar en equipo y para relacionarse con otras personas del mismo o distinto ámbito profesional.
· CGII22. Motivación por la calidad.
· CGII4. Conocimiento de una lengua extranjera.
· CGII7. Resolución de problemas.
· CGII9. Trabajo en equipo.
· CEII5. 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.
· CEII8. Conocimiento de las materias básicas y tecnologías, que capaciten para el aprendizaje y desarrollo de nuevos métodos y tecnologías, así como las que les doten de una gran versatilidad para adaptarse a nuevas situaciones.
· CR1. 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.
· CR6. 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.
· CR7. Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema.
· CR8. Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
· CR16. Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.
· CR17. Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.

4.3 Competencias transversales y de materia

· Competencia 1. CM1. Conocer y utilizar lenguajes orientados a objetos para el desarrollo de sistemas software.
· Competencia 2. CM2. Programar aplicaciones de forma robusta, correcta y eficiente.

· Competencia 3. CM3. Usar las herramientas de un entorno de desarrollo de programación para crear y desarrollar aplicaciones.

5. Contenidos

TEMA 0. Presentación de la asignatura

- Conocer las características de la asignatura (objetivos, contenidos, metodología, bibliografía y evaluación) y su relación con el resto de asignaturas del plan de estudios.
- Conocer la evolución de los lenguajes de programación OO.
- Conocer la tecnología Java y las características del lenguaje Java.

TEMA 1. Introducción al paradigma Orientado a Objetos

- Diferenciar entre paradigma y lenguaje de programación.
- Conocer los elementos que constituyen el modelo de objetos.
- Entender la calidad del software y cómo el paradigma de programación OO contribuye a mejorarla.

TEMA 2. Clases y Objetos

- Definir el concepto de clase y describir los elementos que la componen (atributos y métodos).
- Entender y explicar la diferencia entre clases (estructura estática) y objetos (estructura dinámica).
- Explicar la importancia del Principio de Ocultamiento de la Información para el diseño de arquitecturas flexibles y coherentes y aplicarlo en el diseño de las clases utilizando los niveles de visibilidad que proporcionan los LPOO.
- Entender el concepto de semántica referencia y las implicaciones en los operadores (igualdad y asignación) en los LPOO.
- Entender el paso de mensajes como la forma de comunicación entre los objetos y explicar la diferencia con la llamada a procedimientos.
- Definir métodos en las clases conociendo el funcionamiento del paso de parámetros.
- Justificar el papel de los constructores en los LPOO.
- Entender el modelo de ejecución OO.
- Conocer y aplicar principios de diseño de clases en la resolución de problemas.

TEMA 3. Herencia

- Entender y explicar el concepto de herencia.
- Entender el concepto de polimorfismo presente en los LPOO.
- Entender y explicar el concepto de ligadura dinámica.
- Describir el papel fundamental que juegan las clases abstractas para escribir código genérico.
- Reconocer e implementar las clases abstractas en el diseño de una jerarquía de clases.
- Entender el papel de las interfaces para la definición de tipos.
- Conocer y aplicar la implementación de métodos en las interfaces.
- Conocer y describir como se relaciona el mecanismo de herencia con: el sistema de tipos, la visibilidad de las características de una clase y la creación de objetos.
- Conocer y aplicar principios de diseño de jerarquías de herencia en la resolución de problemas.
- Conocer la herencia múltiple y los problemas asociados..

TEMA 4. Genericidad y colecciones

- Entender y explicar el papel de la genericidad en los lenguajes con comprobación estática de tipos.
- Definir, instanciar y utilizar clases genéricas.
- Entender las implicaciones de la herencia en la definición de tipos genéricos (estructuras de datos polimórficas).
- Entender y justificar el uso de la genericidad restringida y aplicarla en la resolución de problemas.
- Conocer las implicaciones de los tipos genéricos en el sistema de tipos en los LPOO.
- Conocer y manejar el API de colecciones de Java.
- Entender el concepto de patrón de diseño.
- Conocer y saber aplicar alguno de los patrones de diseño: Método Plantilla y Estrategia.

TEMA 5. Corrección y Robustez

- Definir los conceptos de corrección y robustez y conocer los mecanismos que proporcionan los LPOO para darles soporte: Asertos y Excepciones.
- Conocer la utilidad de los asertos como mecanismo de depuración y sus limitaciones.
- Conocer el mecanismo de excepciones y aplicarlo en la implementación de rutinas.
- Conocer cómo influye la herencia en la especificación de asertos y excepciones.

- Describir la técnica del Diseño por Contrato y cómo puede soportarse haciendo uso del mecanismo de excepciones.
- Conocer y aplicar principios de diseño de excepciones en la resolución de problemas.

TEMA 6. Programación funcional y streams

- Relacionar la programación funcional y la programación orientada a objetos a través del uso de las expresiones *lambda*.
- Conocer y utilizar las referencias a métodos y constructores

PRÁCTICAS

Práctica 1. Presentación del lenguaje Java: Relacionada con los contenidos Tema 0 y Tema 1

- Conocer la diferencia entre los entornos Java de desarrollo (JDK, Java Development Kit) y de ejecución (JRE, Java Runtime Environment).
- Conocer la ubicación de los comandos Java para el desarrollo de aplicaciones y cómo ejecutarlos desde la línea de comandos.
- Entender la importancia de las variables de entorno para la correcta ejecución de los comandos Java.

Práctica 2. Definición de clases: Relacionada con los contenidos Tema 2

- Crear, compilar y ejecutar un proyecto en Eclipse.
- Conocer y aplicar correctamente el concepto de clase.
- Aplicar el principio de ocultación de la información en la definición de una clase. Definir correctamente la visibilidad de las declaraciones.
- Definir adecuadamente métodos de acceso y modificación para los atributos de una clase.
- Comprender la utilidad de los constructores como mecanismo de inicialización de los objetos. Asimismo, aplicar reutilización en la definición de constructores.
- Conocer la diferencia entre definiciones de instancia y de clase, y aplicarlas correctamente.
- Comprender la semántica de las referencias en Java.
- Utilizar el concepto de paquete como mecanismo de organización del código.
- Comprender la importancia de la aplicación de una convención de nombrado en la escritura de código. Aplicar la convención de nombres de Java.

Práctica 3. Relación de clientela entre clases: Relacionada con los contenidos Tema 2

- Objetivos formativos de la práctica 2.
- Entender el concepto de propiedad calculada. Valorar cuando una propiedad conviene representarla con un atributo o realizar un cálculo para su obtención.
- Aplicar la sobrecarga en la definición de métodos como facilidad del lenguaje para definir signaturas coherentes de métodos y reutilizar la implementación entre métodos sobrecargados.
- Definición y uso de enumerados.
- Definición y uso de constantes.

- Establecer relaciones entre clases mediante relaciones de clientela.

Práctica 4. Diseño de clases I: Relacionada con los contenidos Tema 2

- Objetivos formativos de las prácticas 2 y 3.
- Implementación de relaciones de clientela multivaluadas con arrays.
- Definición de argumentos variables en la implementación de los métodos de una clase.

Práctica 5. Diseño de clases II: Relacionada con los contenidos Tema 2

- Objetivos formativos de las prácticas 2, 3 y 4.
- Implementación de relaciones de clientela multivaluadas con listas.
- Conocer y utilizar adecuadamente listas en Java.

Práctica 6. Herencia: Relacionada con los contenidos Tema 3

- Utilizar el concepto de herencia como mecanismo de reutilización de código donde se ha de cumplir la relación "es-un" entre las clases.
- Entender la necesidad de la redefinición de métodos en la aplicación de la herencia.
- Comprender el concepto de polimorfismo y ligadura dinámica.

Práctica 7. Clase Object: Relacionada con los contenidos Tema 3

- Implementar correctamente los métodos equals, clone, hashCode y toString.
- Conocer y utilizar adecuadamente conjuntos en Java (java.util.HashSet).

Práctica 8. Clases abstractas e interfaces: Relacionada con los contenidos Tema 3

- Implementar clases abstractas.
- Reconocer la necesidad de factorizar código aplicando el patrón de diseño del Método Plantilla.
- Definir e implementar Interfaces.

Práctica 9. Colecciones: Relacionada con los contenidos Tema 4

- Estudio del API de colecciones de Java.
- Interfaces Comparable y Comparator.

Práctica 10. Excepciones: Relacionada con los contenidos Tema 5

- Diseño por Contrato.
- Excepciones.

Práctica 11. Programación funcional y streams: Relacionada con los contenidos Tema 6

- Definición de expresiones lambda.
- Utilización de referencias a métodos y constructores.
- Implementación de métodos en las interfaces.

Práctica 12. Resolución de proyectos: Relacionada con los contenidos Tema 6,Tema 1,Tema 2,Tema 3,Tema 5 y Tema 4

Aplicación de los principios de diseño OO en el desarrollo de varios ejercicios de programación.

6. Metodología Docente

Actividad Formativa	Metodología	Horas Presenciales	Trabajo Autónomo	Volumen de trabajo
A1. Clases de teoría	Lección magistral	24	12	36
A1. Resolución de ejercicios	Actividades de clase práctica en el aula: actividades prácticas de ejercicios y resolución de problemas.	6	12	18
A5. Seminarios virtuales		0	4	4
A1. Examen final		4.5	20	24.5
A2. Clases de laboratorio	Actividades de clase práctica en el laboratorio: aprendizaje orientado a proyectos. Suponen la realización de tareas por parte de los alumnos, dirigidas y supervisadas por el profesor.	21.5	12	33.5
A5. Ejercicios de programación		0	30	30
A4. Revisión de ejercicios	Tutorias individualizadas	4	0	4
	Total	60	90	150

7. Horario de la asignatura

http://www.um.es/informatica/index.php?pagina=planificacion&subseccion=horarios

http://www.um.es/web/matematicas/contenido/estudios/grados/pes/2018-19#horarios

8. Sistema de Evaluación

Métodos / Instrumentos	Examen teórico-práctico. En este instrumento incluimos desde el tradicional examen escrito o tipo test hasta los exámenes basados en resolución de problemas, pasando por los de tipo mixto que incluyen cuestiones cortas o de desarrollo teórico junto con pequeños problemas. También se incluye aquí la consideración de la participación activa del alumno en clase, la entrega de ejercicios o realización de pequeños trabajos escritos y presentaciones.
Criterios de Valoración	Parte 1: Preguntas cortas (escrito) (45%): • Dominio del lenguaje Java. • Dominio de los conceptos del paradigma OO. • Precisión en las respuestas. • Claridad expositiva. Parte 2: Ejercicio de programación (ordenador) (55%): • Correcta aplicación de los conceptos de la POO y los principios del diseño modular en la resolución de ejercicios. • Atención prestada a los criterios de calidad del software, fundamentalmente: extensibilidad, reutilización, corrección, robustez.
Ponderación	60
Métodos / Instrumentos	Informe técnico. En este instrumento incluimos los resultados de actividades prácticas, o de laboratorio, junto con sus memorias descriptivas. Los resúmenes del estado del arte o memorias de investigación sobre temas concretos. Y la posibilidad de realizar entrevistas personales o presentaciones de los trabajos realizados también entran en esta categoría.
Criterios de Valoración	• Correcta aplicación de los conceptos de la POO y los principios del diseño modular. • Atención prestada a los criterios de calidad del software, fundamentalmente: extensibilidad, reutilización, corrección, robustez, legibilidad y documentación.
Ponderación	40

Fechas de exámenes

http://www.um.es/informatica/index.php?pagina=planificacion&subseccion=examenes
http://www.um.es/web/matematicas/contenido/estudios/grados/pes/2018-19#examenes

9. Resultados del Aprendizaje

· Entender la calidad del software como un compromiso entre distintos factores externos y reconocer un método de construcción de software modular como la técnica para mejorar los factores de calidad de extensibilidad, reutilización y fiabilidad (corrección y robustez).

· Enumerar y explicar los requisitos de los módulos reutilizables y las limitaciones de las estructuras de módulos tradicionales (rutinas/paquetes) y de la técnica de sobrecarga para cumplir dichos requisitos.

· Justificar por qué se utilizan los datos como clave para descomponer los sistemas en módulos en lugar de las funciones (descomposición funcional).

· Definir el concepto central de la tecnología de objetos, la clase, y describir los elementos que la componen (atributos y rutinas), contrastando cómo se especifica en diferentes LPOO. Explicar los principios de diseño de clases y aplicarlos en la resolución de problemas.

· Explicar las relaciones entre clases (estructura estática) y objetos (estructura dinámica).

· Contrastar el modelo de ejecución OO frente al modelo de ejecución tradicional, destacando el papel primordial que juegan las referencias.

· Entender la genericidad como un mecanismo necesario en los LPOO tipados para reconciliar la fiabilidad y la reutilización. Contrastar el soporte a la genericidad en diferentes LPOO.

· Describir la técnica del Diseño por Contrato y aplicarla en la implementación de rutinas en distintos LPOO.

· Describir y saber aplicar las técnicas para abordar los casos excepcionales: esquema a priori, esquema a posteriori, mecanismo de excepciones.

· Explicar los conceptos de herencia, polimorfismo y ligadura dinámica y cómo contribuyen a alcanzar los criterios de reutilización de módulos. Aplicar los principios de diseño para especificar jerarquías de herencia.

· Explicar los criterios para escoger entre una relación de herencia o de clientela entre clases y aplicarlo en la resolución de problemas concretos.

· Describir y ejemplificar el papel fundamental de las clases abstractas para escribir código genérico.

· Explicar la diferencia entre iteradores internos y externos y el modo de implementar cada uno de ellos.

· Describir y aplicar los esquemas para parametrizar una rutina con acciones (esquema basado en composición y basado en herencia).

· Diferenciar entre renombramiento y redefinición de características. Explicar cada una de las adaptaciones y la política que siguen diferentes LPOO. Aplicarlo en la resolución de ejercicios.

· Describir cómo se relaciona el mecanismo de herencia con: el sistema de tipos, la ocultación de información, genericidad, aserciones y excepciones.

· Justificar la necesidad de incluir la herencia múltiple en los lenguajes de programación.

· Explicar el patrón Estrategia para motivar el concepto de Patrón de Diseño.

· Describir los problemas que surgen de la utilización de la herencia múltiple (colisión de nombres y herencia repetida) y explicar las posibles soluciones a dichos problemas en diferentes LPOO.

· Explicar diferentes técnicas para simular la herencia múltiple en lenguajes que sólo disponen de herencia simple.

· Manejar un entorno de programación y un LPOO concreto: editor, compilador y diseñador de interfaces gráficas de usuario. Utilizar dicho lenguaje y entorno para desarrollar un mini-proyecto de programación en el que se apliquen los principios de diseño OO.

· Diseñar pruebas unitarias en paralelo al proyecto de programación.

10. Bibliografía

Bibliografía Básica

Número de Título: 450121. Cay S.Horstmann, Gary Cornell. Java 2. Vol I. Fundamentos. Pearson/ Prentice Hall, 2006.

11. Observaciones y recomendaciones

SOBRE LA NOTA DEL EXAMEN:

- El examen consta de dos partes, una parte de preguntas cortas (escrito) y otra parte práctica (ordenador), con un peso de 45% y 55%, respectivamente.

- Para aprobar el examen es requisito tener una nota igual o superior a 4 en cada parte, en cuyo caso se calculará la nota del examen como la media ponderada por los pesos de cada parte.

- En el caso de que el examen esté suspenso, la nota de cada parte no se guardará para el resto de convocatorias.

SOBRE LA CALIFICACIÓN EN EL ACTA:

- La nota de la asignatura está formada por dos componentes: examen (60%) y ejercicios prácticos (40%).

- Para aprobar la asignatura es requisito aprobar (obtener una nota igual o superior a 5) en el examen y en los ejercicios prácticos.

- La calificación en el acta será "No presentado" si no se ha realizado alguno de los componentes y el otro está aprobado.

- En caso de no aprobar alguno de los componentes, la calificación final será “Suspenso”. La nota se calculará como la media ponderada por los pesos de cada componente. En ningún caso la nota será superior a 4,9

SOBRE NECESIDADES EDUCATIVAS ESPECIALES:

Aquellos estudiantes con discapacidad o necesidades educativas especiales podrán dirigirse al Servicio de Atención a la Diversidad y Voluntariado (ADYV; http://www.um.es/adyv/) para recibir orientación sobre un mejor aprovechamiento de su proceso formativo y, en su caso, la adopción de medidas de equiparación y de mejora para la inclusión, en virtud de la Resolución Rectoral R-358/2016. El tratamiento de la información sobre este alumnado, en cumplimiento con la LOPD, es de estricta confidencialidad.