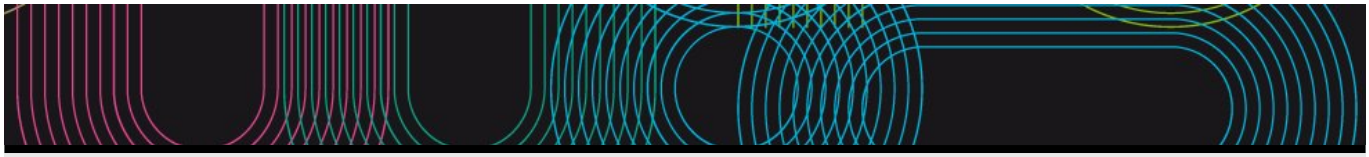

DATOS IDENTIFICATIVOS

Programación II

Asignatura	Programación II			
Código	V05G300V01302			
Titulacion	Grado en Ingeniería de Tecnologías de Telecomunicación			
Descriptores	Creditos ECTS	Seleccione	Curso	Cuatrimestre
	6	OB	2	1c
Lengua Impartición	Castellano			
Departamento	Ingeniería telemática			
Coordinador/a	Fernández Masaguer, Francisco			
Profesorado	Blanco Fernández, Yolanda Fernández Masaguer, Francisco			
Correo-e	francisco.fernandez@det.uvigo.es			
Web	http://www.faitic.es			
Descripción general	El objetivo general de la asignatura es proporcionar al alumno los fundamentos teóricos y las competencias prácticas que le permitan analizar, diseñar, desarrollar y depurar aplicaciones informáticas siguiendo el paradigma orientado a objetos. Esta es una asignatura eminentemente práctica y en este sentido está orientada al trabajo de los alumnos en la realización de uno o varios proyectos. Para facilitar el desarrollo de los proyectos, en la asignatura se realizará primeramente una muy breve introducción a la disciplina de Ingeniería del Software, conectandola con el paradigma de la programacion orientada a objetos (POO) y limitandola solo a las etapas de análisis, diseño, implementación y depuración. A continuación se analizarán en detalle los elementos de la POO, utilizando elementos y diagramas UML que serán utilizados por los alumnos en sus desarrollos. Para alcanzar este objetivo general los contenidos que se verán en la asignatura se pueden resumir en los siguientes ítems: - Conceptos básicos de Ingeniería del Software. - Conceptos básicos de la orientación a objetos: clases y objetos. - Encapsulación. Principio de ocultación. Conceptos de desacoplamiento y cohesión - Herencia, abstracción, polimorfismo y reutilización - Relaciones entre clases: generalización, asociación y dependencia. - Comunicación entre objetos: métodos, eventos, mensajes. - Persistencia. Almacenamiento en ficheros y en bases de datos. - Generación, captura y procesamiento de excepciones. - Lenguaje de modelado UML.			

Competencias

Código	
B6	CG6 Facilidad para el manejo de especificaciones, reglamentos y normas de obligado cumplimiento.
B14	CG14 Capacidad para utilizar herramientes informáticas de búsqueda de recursos bibliográficos o de información.
C50	(CE50/T18) Capacidad de desarrollar, interpretar y depurar programas utilizando los conceptos básicos. de la Programación Orientada a Objetos (POO): clases y objetos, encapsulación, relaciones entre clases y objetos, y herencia.

C51	(CE51/T19) Capacidad de aplicación básica de las fases de análisis, diseño, implementación y depuración de programas en la POO.
C52	(CE52/T20) Capacidad de manejo de herramientas CASE (editores, depuradores).
C53	(CE53/T21) Capacidad de desarrollo de programas atendiendo a los principios básicos de calidad de la ingeniería del software, teniendo en cuenta las principales fuentes existentes en normas, estándares y especificaciones.

Resultados de aprendizaje

Resultados previstos en la materia	Resultados de Formación y Aprendizaje	
Comprender los aspectos básicos de la Programación Orientada a Objetos (POO).	B14	C50
Conocer los principales diagramas UML para la documentación de las fases de análisis y diseño de programas de acuerdo a la POO.	B6 B14	C52 C53
Adquirir habilidades sobre el proceso de análisis, diseño, implementación y depuración de aplicaciones de acuerdo a la POO, teniendo en cuenta los estándares principales y normas de calidad.	B6 B14	C51 C53
Adquirir una madurez básica en técnicas de desarrollo y depuración de programas para permitir el aprendizaje autónomo de nuevas capacidades y lenguajes de programación.	B6	C51 C52 C53

Contenidos

Tema	
1. Introducción al paradigma orientado a objetos	a. Breve introducción a la asignatura y su organización b. Nacimiento del paradigma c. Bases: clases y objetos d. Conceptos de encapsulación, herencia (generalización), y polimorfismo e. Breve introducción a UML.
2. Encapsulación	a. Clases, interfaces y paquetes b. Métodos y variables miembro. Visibilidad. Resolución de ámbito. c. Método constructor d. Paso de parámetros: punteros y referencias e. Punteros a objetos
3. Herencia	a. Clases derivadas y tipos de herencia b. Clases abstractas c. Herencia múltiple d. Clase object
4. Diseño orientado a objetos	a. Fundamentos de diseño. b. Conceptos básicos de la Ingeniería del Software. c. Utilización de diagramas UML
5. Polimorfismo	a. Sobrecarga y sobreescritura b. Clases abstractas e interfaces c. Clases genéricas
6. Gestión de excepciones	a. Fundamentos de excepciones b. Manipulación de excepciones en Java

Planificación

	Horas en clase	Horas fuera de clase	Horas totales
Sesión magistral	28	42	70
Resolución de problemas y/o ejercicios	9	9	18
Resolución de problemas y/o ejercicios de forma autónoma	4	10	14
Estudio de casos/análisis de situaciones	1	1	2
Proyectos	9	31	40
Estudio de casos/análisis de situaciones	0	1	1
Resolución de problemas y/o ejercicios	3	0	3
Pruebas prácticas, de ejecución de tareas reales y/o simuladas.	2	0	2

*Los datos que aparecen en la tabla de planificación son de carácter orientativo, considerando la heterogeneidad de alumnado

Metodologías

Descripción

Sesión magistral	Clases que combinarán la exposición de los conceptos a tratar en la asignatura con la realización de pequeños ejercicios. Éstos podrán ser resueltos por el docente o por los propios alumnos individualmente y/o en grupo. El objetivo es fomentar el debate en clase y reforzar la adquisición de destrezas. Esta metodología está orientada a la adquisición por el alumno de las competencias CE50, CE51 y CE53.
Resolución de problemas y/o ejercicios	En el laboratorio, el profesor planteará pequeños retos que serán resueltos colectivamente para que se puedan debatir los conceptos subyacentes, las diferentes opciones de resolución y que los alumnos adquieran las destrezas objetivo de la asignatura. Esta metodología está orientada a la adquisición por el alumno de las competencias CE50, CE51 y CE53.
Resolución de problemas y/o ejercicios de forma autónoma	Los alumnos resolverán de forma autónoma los problemas que el profesor les plantee en el laboratorio. Las soluciones y las dudas que surjan al abordar dichos problemas serán puestas en común para consensuar la mejor forma de resolución. Esta metodología está orientada a la adquisición por el alumno de las competencias CE50, CE51, CE53, CG6 y CG14.
Estudio de casos/análisis de situaciones	Puesta en común de los diseños propuestos por los alumnos para solucionar el proyecto que han de llevar a cabo durante la segunda parte del curso. La comparación de las diferentes propuestas servirá para seleccionar las mejores opciones y como realimentación para, si es oportuno, mejorar los diseños realizados. Esta metodología está orientada a la adquisición por el alumno de las competencias CE51 y CE52.
Proyectos	Los alumnos implementarán el sistema software planteado por el profesor. Dispondrán para ello de la segunda parte del curso, combinando trabajo presencial en el laboratorio supervisado por el profesor con trabajo no presencial. Esta metodología está orientada a la adquisición por el alumno de las competencias CE50, CE53, CG6 y CG14.

Atención personalizada

Metodologías	Descripción
Resolución de problemas y/o ejercicios	Revisión y comentarios de ejercicios resueltos. Glosario de errores frecuentes a evitar. Recomendaciones de estilo y organización.
Proyectos	Junto a comentar de forma conjunta diversas recomendaciones y estrategias para la buena realización del proyecto, se revisa con cada grupo el nivel de comprensión del proyecto, dudas particulares que puedan surgir, errores de diseño y codificación y opciones de mejora.
Resolución de problemas y/o ejercicios de forma autónoma	Revisión y comentarios con cada grupo de las diversas prácticas propuestas durante su realización. Resolución de errores de compilación y ejecución. Detección y solución de errores conceptuales.
Estudio de casos/análisis de situaciones	Revisión y crítica general del diseño UML de cada grupo durante su realización.

Evaluación

Descripción	Calificación	Resultados de Formación y Aprendizaje

Proyectos	Los alumnos, organizados en grupos de dos, entregarán el proyecto software propuesto como máximo el primer dia lectivo despues de las vacaciones de Navidad. Éste constará de su diseño final (diagramas UML), el código y la documentación generada explicativa de la implementación. Que el código entregado pueda ser compilado y ejecutado en los equipos de los laboratorios es condición indispensable para superar esta prueba de evaluación. Durante la última semana del curso, los alumnos tendrán una entrevista con el profesor en el horario del laboratorio, dedicada a demostrar la autoría del proyecto y realizar diversas pruebas de funcionalidad. Los dos miembros de cada grupo deberán estar obligatoriamente presentes en dicha entrevista. Las cuestiones planteadas en la misma deberán ser respondidas individualmente para poder constatar el grado de entendimiento e implicación del alumno en el proyecto desarrollado, debiendo cada alumno identificar las partes del proyecto que ha implementado. Las respuestas proporcionadas su usarán para establecer, junto con un conjunto de tests de funcionalidad y del análisis de la calidad del código, la nota individual de cada alumno. En caso de que un alumno no acredite adecuadamente la autoría, no se le dará por válido el proyecto, y se considerará suspenso en la convocatoria correspondiente. Para los alumnos que acrediten adecuadamente la autoría, la evaluación del proyecto tendrá en cuenta tanto las respuestas proporcionadas en la entrevista de autoría, como la correcta funcionalidad, como la calidad del código y el uso de las técnicas de la programación orientada a objetos. La determinación de la correcta funcionalidad se realizará mediante un conjunto de alrededor de 50 tests sobre el software entregado.	33	B6 B14	C50 C53
Estudio de casos/análisis de situaciones	Al final de la 9ª semana del curso académico los alumnos, organizados en grupos de dos, entregarán el diseño UML de un proyecto software. En horas lectivas los integrantes de cada grupo realizarán con el profesor una breve entrevista de la autoría de este diseño, la cual junto con el diseño entregado, se usará para establecer la nota individual de cada alumno.	7		C51 C52
Resolución de problemas y/o ejercicios	Resolución de problemas y/o ejercicios: Examen escrito e individual, realizado en la fecha aprobada por la Junta de Escuela para ello, que constará de la combinación de los siguientes tipos de preguntas: resolución de problemas, cuestiones breves para resolver aplicando los conceptos teóricos explicados en clase, justificar razonadamente si una o varias afirmaciones son verdaderas o falsas, pequeños tests sobre aspectos teóricos y de aplicación. En este examen no está permitido ningún material de apoyo, utilización de apuntes, libros o colecciones de problemas. El número y la combinación de dichas preguntas se fijará para cada examen en particular.	50		C50 C51 C53
Pruebas prácticas, de ejecución de tareas reales y/o simuladas.	Al final de la 7ª semana del curso académico los alumnos, organizados en grupos de dos, entregarán las prácticas de iniciación en Java propuestas en el laboratorio. En horas lectivas los integrantes de cada grupo realizarán con el profesor una breve entrevista de la autoría de las practicas de iniciación, la cual junto con un conjunto de tests de correcto funcionamiento sobre el software entregado, se usará para establecer la nota individual de cada alumno.	10		C50 C51 C52 C53

Otros comentarios sobre la Evaluación

Existen dos modalidades de evaluación de la asignatura: evaluación continua (EC) y evaluación tradicional (ET). Los alumnos deberán elegir una de las dos modalidades teniendo en cuenta las siguientes condiciones:

- La EC incluye las 4 pruebas descritas en el apartado evaluación.

- Tanto si optan por la EC como si optan por la ET los alumnos deberán realizar un proyecto de laboratorio. Para facilitar la elección de EC o ET los alumnos dispondrán en Faitic del proyecto a realizar a partir de la 4ª semana del curso académico.

- En la ET el proyecto se realizará de forma individual.

- Los alumnos que opten por la EC deberán entregar al final de la 9ª semana del curso académico el diseño UML del proyecto a realizar (correspondiente a la 2ª prueba descrita en el apartado de evaluación). Mediante dicha entrega los alumnos se comprometen a seguir la EC y renuncian a la ET. Desde este momento estos alumnos no podrán figurar como "No presentados''.

- Los alumnos que no entreguen el diseño UML del proyecto en la fecha estipulada renuncian a la EC, de modo que serán evaluados mediante la modalidad de ET. No existe la posibilidad de sumarse a la EC en las siguientes pruebas intermedias.

- Las pruebas de EC no son en ningún caso recuperables, no pudiendo repetirse fuera de las fechas estipuladas por los profesores.

- No se guardan calificaciones (de pruebas de EC ni de proyectos prácticos o exámenes) de un curso a otro.

Primera convocatoria. Alumnos que opten por la EC. Serán evaluados como sigue:

 Parte teórica:

- Examen escrito (50%). Examen individual. Se corresponde con la 3ª prueba descrita en el apartado evaluación. La nota de este examen teórico solamente se guardará para la segunda convocatoria si es igual o superior a 4.5 sobre 5.

 Parte práctica:

- Prácticas de iniciación en Java (10%). A realizar en grupos de dos. Se corresponde con la 4ª prueba descrita en el apartado de evaluación.

- Proyecto (40%). A realizar en grupos de dos. Se divide en dos partes:

1. Diseño (7%). Se corresponde con la 2ª prueba descrita en el apartado evaluación.
2. Implementación (33%). Se corresponde con la 1ª prueba descrita en el apartado evaluación.

 Los requisitos para aprobar serán:

- Un mínimo de 1/3 sobre el total en la parte teórica.

- Un mínimo de 1/3 sobre el total en la parte de implementación del proyecto.

- Una nota total (suma de las 4 pruebas) igual o superior a 5.

- Si la nota total es igual o superior a 5 pero no se ha alcanzado la nota mínima en alguna parte, la nota final será 4.5 puntos (suspenso).

Primera convocatoria. Alumnos que opten por la ET. Serán evaluados como sigue:

 Parte teórica:

- Examen escrito (50%). Examen individual. Se corresponde con la 3ª prueba descrita en el apartado evaluación. La nota de este examen teórico solamente se guardará para la segunda convocatoria si es igual o superior a 4.5 sobre 5.

 Parte práctica:

- Realización individual de un proyecto software que supondrá el restante 50% de la nota final. Este proyecto constará del diseño (diagramas UML), el código Java y la documentación generada explicativa de la implementación. La evaluación tendrá en cuenta correcto diseño, correcta funcionalidad, calidad del código y uso de técnicas de POO. Deberá ser entregado como máximo el primer día lectivo tras las vacaciones de Navidad.

- Realización de una entrevista con el profesor dedicada a demostrar la autoría del proyecto. Dicha entrevista tendrá lugar en el laboratorio durante la última semana del curso. Si el alumno no acredita adecuadamente la autoría no superará la convocatoria, y deberá realizar el proyecto correspondiente a la segunda convocatoria.

 Los requisitos para aprobar serán:

- Un mínimo de 1/3 sobre el total en la parte teórica.

- Un mínimo de 1/3 sobre el total en el proyecto.

- Una nota total (suma de las 2 pruebas) igual o superior a 5.

- Si la nota total es igual o superior a 5 pero no se ha alcanzado la nota mínima en alguna parte, la nota final será 4.5 puntos (suspenso).

Segunda convocatoria. Los alumnos serán evaluados como sigue:

 Parte teórica:

- Examen escrito (50%). Examen individual. Se corresponde con la 3ª prueba descrita en el apartado evaluación. La nota del

examen teórico no se guarda en ningun caso.

 Parte práctica:

Dependerá de si el alumno ha entregado o no el proyecto en la primera convocatoria. Para los alumnos que han seguido la EC en la primera convocatoria, se considerará que un alumno ha entregado el proyecto si como mínimo ha entregado un diseño UML en el que haya obtenido una nota igual o superior a 0.4 sobre 0.7.

- Los alumnos que no entreguen el proyecto en la primera convocatoria o que no hayan superado la entrevista de autoría, deberán necesariamente realizar el proyecto ampliado de la segunda convocatoria. En cualquier caso se pierden las notas de las partes de iniciación en Java y diseño UML si optaron por la EC en la primera convocatoria, es decir, serán evaluados sobre 5.

- La parte práctica a realizar por los alumnos que entreguen el proyecto en la primera convocatoria dependerá de la nota obtendida en el proyecto en dicha convocatoria, como sigue:

- *Nota >= 1.5 por EC o Nota >= 2.5 por ET*. Se les mantendrá la nota. Podrán, sin embargo, mejorar la puntuación del proyecto entregando una nueva versión del de la primera convocatoria junto con las nuevas funcionalidades a realizar, que se publicarán en su momento en Faitic. Del mismo modo, deberán entregar un documento que recoja los cambios y actualizaciones realizados en el proyecto sobre la versión entregada en la primera convocatoria.
- *Nota entre 1 y 1.5 por EC o Nota entre 5/3 < 2.5 por ET*. Deberán necesariamente realizar el proyecto ampliado de la segunda convocatoria. No se les mantendrá la nota del proyecto de la primera convocatoria, pero sí la de las partes de prácticas de iniciación en Java y diseño UML, si optaron por la EC en la primera convocatoria.
- *Nota < 1 por EC o Nota < 5/3 por ET*. Deberán necesariamente realizar el proyecto ampliado de la segunda convocatoria. En cualquier caso se pierden las notas de las partes de iniciación en Java y diseño UML si optaron por la EC en la primera convocatoria, es decir, serán evaluados sobre 5.

 Requisitos de aprobado**.** Los requisitos para aprobar en esta convocatoria serán:

- Un mínimo de 1/3 sobre el total, en la parte teórica.

- Un mínimo de 1/3 sobre el total en el proyecto sin tener el cuenta la nota de iniciación en Java y diseño UML si optaron por la EC en la primera convocatoria.

- Una nota total (suma de todas las pruebas) igual o superior a 5.

Si la nota total es igual o superior a 5 pero no se ha alcanzado la nota mínima en alguna parte, la nota final será 4.5 puntos (suspenso).

Fuentes de información

Bibliografía Básica

W. Savitch, **Absolute Java**, 4ª edición, Pearson, 2010

Y. D. Liang, **Introduction to Java programming**, 8ª, Pearson, 2010

P. Deitel, H. Deitel, **Java: How to program**, 9ª, Pearson, 2011

Bibliografía Complementaria

B. Eckel, **Thinking in Java**, 4ª edición, Prentice-Hall, 2006

P. Niemeyer, D. Leuck, **Learning Java**, 4ª edición, O'Reilly., 2013

Oracle, **Java SE. Oracle**,

Oracle, **Java API Specifications**, 2016

G. Booch, J. Rumbaugh, I. Jacobson, **The Unified Modeling Language User Guide**, 2, Addison-Wesley., 2005

S. Zakhour, S. Hommel, J. Royal, I. Rabinovitch, T. Risser, M. Hoeber, **The Java Tutorial. A short course on the basics**, 4ª edición, Prentice-Hall, 2006

A. Eberhart, S. Fischer, **Java Tools**, Wiley, 2002

M. Page-Jones, **Fundamentals of object-oriented design in UML**, Addison-Wesley, 2002

M. Fowler, **UML Distilled: A Brief Guide to the Standard Object Modeling Language**, 3ª edicion, Addison-Wesley., 2003

Jean-Michel DOUDOUX, **Développons en Java 2.10**, 2016

Recomendaciones

Asignaturas que se recomienda haber cursado previamente