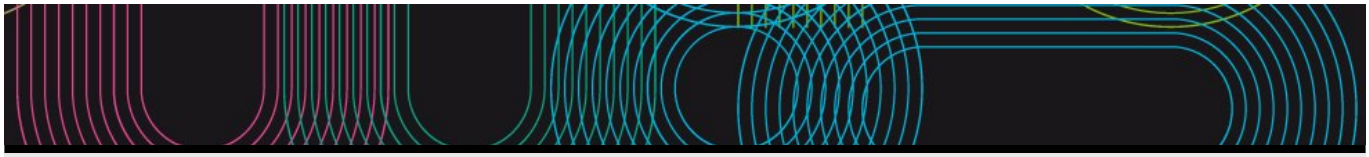

DATOS IDENTIFICATIVOS				
Programación II				
Asignatura	Programación II			
Código	V05G300V01302			
Titulacion	Grado en Ingeniería de Tecnologías de Telecomunicación			
Descriptores	Creditos ECTS	Seleccione	Curso	Cuatrimestre
	6	OB	2	1c
Lengua Impartición	Castellano			
Departamento	Ingeniería telemática			
Coordinador/a	Fernández Masaguer, Francisco			
Profesorado	Blanco Fernández, Yolanda Fernández Masaguer, Francisco Sousa Vieira, Estrella			
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 también se hace una introducción a la Ingeniería del Software. En este sentido no se ocupa de todas las fases generalmente reconocidas en los procesos de desarrollo software, que van desde la captura y descripción de requisitos hasta el despliegue de los sistemas, sino que se tratan principalmente las etapas de análisis, diseño, implementación y depuración. En primer lugar se presentará la Ingeniería del Software como disciplina imprescindible para el desarrollo de grandes aplicaciones informáticas, mostrando los principales retos a los que se enfrenta y los conceptos básicos que se utilizarán. A continuación se analizarán los elementos del paradigma de la programación orientado a objetos (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: El paradigma Orientado a Objetos - 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. Introducción a la Ingeniería del Software - Conceptos básicos de la Ingeniería del Software. Reseña histórica - Introducción y concepto de Ciclo de Vida. Estándar ISO/IEC 12207 - Introducción a las metodologías de desarrollo de software. Clasificación - Introducción a los procesos de desarrollo de software orientado a objetos. Métrica v3 y el Proceso Unificado. - Fases principales en el desarrollo orientado a objetos: análisis, diseño, implementación y pruebas. - Introducción al lenguaje de modelado UML: estructura e interacción.

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	La atención individualizada se articulará con el seguimiento del trabajo de cada alumno, supervisando las soluciones que propone para cada problema planteado en las prácticas de laboratorio, y el seguimiento del proyecto software que debe implementar.
Proyectos	La atención individualizada se articulará con el seguimiento del trabajo de cada alumno, supervisando las soluciones que propone para cada problema planteado en las prácticas de laboratorio, y el seguimiento del proyecto software que debe implementar.
Resolución de problemas y/o ejercicios de forma autónoma	La atención individualizada se articulará con el seguimiento del trabajo de cada alumno, supervisando las soluciones que propone para cada problema planteado en las prácticas de laboratorio, y el seguimiento del proyecto software que debe implementar.
Estudio de casos/análisis de situaciones	La atención individualizada se articulará con el seguimiento del trabajo de cada alumno, supervisando las soluciones que propone para cada problema planteado en las prácticas de laboratorio, y el seguimiento del proyecto software que debe implementar.

Evaluación

	Descripción	Calificación	Resultados de Formación y Aprendizaje	
Proyectos	Los alumnos, organizados en grupos de dos, entregarán el proyecto software propuesto como máximo el día 5 de Enero. 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. En caso de que un alumno no acredite adecuadamente la autoría, la evaluación del proyecto se hará mediante un examen práctico de programación individual en el laboratorio docente, en la fecha aprobada por la Junta de Escuela a tal fin. Si el alumno no se presenta a este examen práctico perderá el 30% de la nota de la asignatura. Para los alumnos que acrediten adecuadamente la autoría, la evaluación del proyecto tendrá en cuenta tanto la correcta funcionalidad como la calidad del código y el uso de las técnicas de la programación orientada a objetos.	30	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 de un proyecto software.	10		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. No se permite la 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.	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 del examen teorico no se guardra en ningun caso.

 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 (10%). Se corresponde con la 2ª prueba descrita en el apartado evaluación.

2. Implementación (30%). 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 (o 1/3 sobre el total del examen práctico en su caso).

- 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 del examen teorico no se guardra en ningun caso.

 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 día 5 de Enero.

- 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, la evaluación de la parte práctica se hará por medio de un examen práctico.

 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 (o 1/3 sobre el total del examen práctico en su caso).

- 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 teorico no se guardra 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 ha obtenido una nota igual o superior a 0.6 sobre 1.

- Los alumnos que no entreguen el proyecto en la primera convocatoria serán evaluados mediante un examen de programación individual que tendrá lugar en el laboratorio en la fecha fijada por la Junta de Escuela a tal fin. La evaluación de este examen supondrá un 50% de la nota final.

- 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, no teniendo que presentarse al examen práctico en la segunda convocatoria. 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. Podrán optar entre realizar el examen práctico o 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 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. Podrán optar entre realizar el examen práctico o 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.

 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 sin tener el cuenta la nota de iniciación en Java y diseño UML si optaron por la EC en la primera convocatoria (o 1/3 sobre el total del examen práctico en su caso).

- 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

Se propone la siguiente bibliografía organizada en dos grandes grupos: manuales básicos y referencias adicionales.

Manuales básicos:

[1] "Absolute Java". W. Savitch, 4ª edición. 2010, Pearson.

[2] "Introduction to Java programming". Y. D. Liang, 8ª edición. 2010, Pearson.

[3] "Java: How to program". P. Deitel, H. Deitel, 9ª edición. 2011, Pearson.

Referencias adicionales:

[1] "Programación orientada a objetos con Java: Una introducción práctica usando BlueJ". D. J. Barnes, M. Kölling, 3ª edición. 2007, Pearson.

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

[3] "Data Structures & Algorithms in Java". M. T. Goodrich, R. Tamassia, 5ª edición. 2010, Willey.

[4] "Java Tools". A. Eberhart, S. Fischer. 2002, Wiley.

[5] "Java in a Nutshell". D. Flanagan, 5ª edición. 2005, O'Reilly.

[6] "Thinking in Java". B. Eckel, 4ª edición. 2006, Prentice-Hall.

[7] "Learning Java". P. Niemeyer, D. Leuck, 4ª edición. 2013, O'Reilly.

[8] "How to Think Like a Computer Scientist. JavaTM Version", 4ª versión. Online: http://www.greenteapress.com/thinkapjava/

[9] "Java notes". F. Swartz. Online: http://www.leepoint.net/notes-java/index.html

[10] "Java SE. Oracle". Online: http://www.oracle.com/technetwork/java/javase/downloads/index.html

[11] "Java 2 Platform Standard Edition 5.0. API Specification". Online: http://download.oracle.com/javase/1.5.0/docs/api/

[12] "The Java Tutorials". Oracle. Online: http://download.oracle.com/javase/tutorial/

[13] "Ingeniería del Software orientada a objetos con UML, Java e Internet". A. Weitzenfeld. 2005, Thomson.

[14] "Open-oriented Analysis and Design with Applications". G. Booch, R. Maksimchuk, M. Engel, B. Young, J. Conallen, K. Houston, 3ª edición. 2007, Addison-Wesley.

[15] "The Unified Modeling Language User Guide". G. Booch, J. Rumbaugh, I. Jacobson, 2ª edición. 2005. Addison-Wesley.

[16] "UML Distilled: A Brief Guide to the Standard Object Modeling Language". M. Fowler, 3ª edición. 2003, Addison-Wesley.

[17] "Fundamentals of object-oriented design in UML". M. Page-Jones. 2002, Addison-Wesley.

Recomendaciones

Asignaturas que se recomienda haber cursado previamente

Programación I/V05G300V01205