PROGRAMA DE LA ASIGNATURA

Curso académico: 2013/2014

Identificación y características de la asignatura					
Código	501275			Créditos ECTS	6
Denominación	Desarrollo de Programas				
Denominacón (inglés)	Program Development				
Titulaciones	Grado en Ingeniería Informática en Ingeniería de Computadores Grado en Ingeniería Informática en Ingeniería del Software				
Centro	Escuela Politécnica				
Semestre	3	Carácter	Obligatorio		
Módulo	Común a la rama de Informática				
Materia	Programación				

Profesor/es			
Nombre	Despacho	Correo-e	Página web
José María Conejero Manzano	20	chemacm@unex.es	http://epcc.unex.es
María Encarnación Sosa Sánchez	10	esosa@unex.es	http://epcc.unex.es

Área de conocimiento	Lenguajes y Sistemas Informáticos
Departamento	Ingeniería de Sistemas Informáticos y Telemáticos
Profesor coordinador (si hay más de uno)	María Encarnación Sosa Sánchez

Competencias

Básicas

(Competencias básicas establecidas para Grado en el Anexo I 3.2 del RD 861/2010. Se recogen por defecto)

CB1: Que los estudiantes hayan demostrado poseer y comprender conocimientos en un área de estudio que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.

CB2: Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.

CB3: Que los estudiantes tengan la capacidad de reunir e interpretar datos relevantes (normalmente dentro de su área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.

CB4: Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.

CB5: Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.

Competencias generales del módulo Común a la rama de Informática

Según los planes de estudio aprobados, esta asignatura cubrirá las siguientes competencias específicas y sus resultados de aprendizaje:

CI07: Conocimiento, diseño y utilización de forma eficiente de los tipos y estructuras de datos más adecuados a la resolución de un problema.
CI08: Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.

Resultados de aprendizaje de estas competencias:

- Puede utilizar de manera eficaz un entorno de programación que incluya herramientas de edición, compilación, depuración y documentación de programas.
- Justifica la utilización de distintos paradigmas de programación y plataformas de desarrollo de software en un determinado contexto.
- Busca, analiza, sintetiza y critica nueva información para aprender nuevos lenguajes, algoritmos, técnicas, paradigmas y metodologías de programación aplicables a distintas áreas, teniendo como objetivo la actualización continua de los conocimientos y competencias.
- Analiza, planifica, diseña y desarrolla soluciones algorítmicas y programas robustos y correctos a problemas planteados, argumentando las decisiones tomadas, evaluando el resultado final y documentando el código y el proceso.

Competencias transversales del módulo Común a la rama de Informática

Según los planes de estudio aprobados, esta asignatura cubrirá las siguientes competencias transversales y sus resultados de aprendizaje:

CT03: Capacidad para resolver problemas.
CT07: Capacidad de análisis y síntesis.

Resultados de aprendizaje de estas competencias:

- Reconoce la estructura de un problema, datos de entrada, incógnitas, magnitudes, condiciones iniciales, así como los pasos para su resolución.
- Extrae del problema las soluciones triviales, reconoce la multiplicidad de soluciones, etc...
- Sabe elegir con fundamento los métodos y medios más adecuados para resolver un problema.
- Desarrollar la capacidad de observación, generalización, abstracción, razonamiento lógico, deductivo e inductivo, y síntesis.
- Identifica relaciones básicas, desagrega los fenómenos en sus partes componentes. Establece relaciones causales sencillas, o identifica las ventajas y desventajas de las decisiones. Establece prioridades en las tareas según su orden de importancia.

Objetivos de aprendizaje de la asignatura

Para desarrollar convenientemente las competencias asignadas a esta asignatura y poder alcanzar los resultados de aprendizaje propuestos, se establecen los siguientes objetivos de aprendizaje concretos, clasificados, según la taxonomía de Bloom, en los niveles de conocimiento, comprensión, aplicación y análisis.

Conocimiento:

Obj. 1. Conocer los fundamentos de la Programación Orientada a Objetos **(CI08, CT07)**.

Obj. 2. Conocer las tareas del ciclo de vida de desarrollo software siguiendo una metodología orientada a objetos que posteriormente llevarán a cabo en entornos profesionales **(CI08, CT03)**.

Obj. 3. Conocer las características principales de los entornos de desarrollo de programas **(CI08, CT07)**.

Obj. 4. Conocer los conceptos necesarios sobre estructuras de datos **(CI07)**.

Obj. 5. Conocer el concepto de genericidad en lenguajes de programación **(CI07)**.

Obj. 6. Conocer diferentes técnicas de manejo de errores en lenguajes de programación **(CI08)**.

Obj. 7. Conocer distintos patrones de diseño para solucionar problemas recurrentes **(CI08, CT03)**.

Obj. 8. Conocer técnicas de verificación y validación de programas **(CI08, CT07)**.

Comprensión:

Obj. 9. Comprender la necesidad y ventajas de la aplicación de las características básicas de la Programación Orientada a Objetos **(CI08, CT07)**.

Obj. 10. Desarrollar la capacidad de abstracción del alumnado para identificar las estructuras de datos más adecuadas para solucionar un problema **(CI07)**.

Obj. 11. Comprender código y soluciones de diseño que haya sido escrito previamente por otros desarrolladores **(CT07)**.

Obj. 12. Analizar librerías externas que faciliten la resolución de problemas concretos **(CT03, CT07)**.

Obj. 13. Comprender el uso del mecanismo de control de errores basado en excepciones **(CI08)**.

Obj. 14. Comprender la necesidad de asegurar la calidad del software desarrollado **(CI08, CT07)**.

Aplicación:

Obj. 15. Realizar el análisis y el diseño detallado de un proyecto software **(CI08)**.

Obj. 16. Permitir a los alumnos resolver supuestos que incluyan la implementación de programas de tamaño medio-grande **(CI08, CT03)**.

Obj. 17. Desarrollar codificaciones correctas y eficientes, utilizando el paradigma orientado a objetos **(CI08, CT03)**.

Obj. 18. Plasmar de forma escrita los pasos realizados en el proceso de desarrollo software, de manera que tanto el usuario de la aplicación, como otros desarrolladores, sean capaces de entender la solución propuesta **(CI08, CT07)**.

Obj. 19. Realizar pruebas que verifiquen la validez funcional, la integridad de los datos y el rendimiento de las aplicaciones informáticas **(CI08, CT07)**.

Obj. 20. Utilizar librerías externas que faciliten la resolución de problemas **(CI07,CT03)**.

Obj. 21. Utilizar con fluidez estructuras de datos para desarrollar algoritmos adecuados a la resolución de problemas concretos **(CI07, CT03)**.

Análisis:

Obj. 22. Analizar las ventajas e inconvenientes de diferentes lenguajes de programación orientados a objetos **(CI08, CT07)**.

Obj. 23. Conocer y utilizar las herramientas enseñadas para valorar las soluciones aportadas al diseño de un programa **(CT07)**.

Obj. 24. Verificar que las soluciones aportadas cumplen los objetivos propuestos de manera eficiente **(CI08, CT07)**.

	Competencias			
Objetivos de Aprendizaje	**CI07**	**CI08**	**CT03**	**CT07**
Conocimiento				
Obj. 1		x		x
Obj. 2		x	x	
Obj. 3		x		x
Obj. 4	x			
Obj. 5	x			
Obj. 6		x		
Obj. 7		x	x	
Obj. 8		x		x
Comprensión				
Obj. 9		x		x
Obj. 10	x			
Obj. 11	x			
Obj. 12			x	x
Obj. 13		x		
Obj. 14		x		x
Aplicación				
Obj. 15		x		
Obj. 16		x	x	
Obj. 17		x	x	
Obj. 18		x		x
Obj. 19		x		x
Obj. 20	x		x	
Obj. 21	x		x	
Análisis				
Obj. 22		x		x
Obj. 23	x			x
Obj. 24		x		x

Temas y contenidos

Breve descripción del contenido

Análisis y Diseño Orientado a Objetos. Técnicas del paradigma de Programación Orientadas a Objetos. Aplicación de diferentes algoritmos y estructuras de datos en casos prácticos. Control de errores y verificación de programas. Patrones de diseño.

Temario de la asignatura

Tema 1: Introducción al lenguaje de programación Java y recordatorio de Entornos de Desarrollo de Programas.

1. Lenguajes de programación compilados vs. interpretados. Compilador Java. Classpath. Creación y destrucción de objetos en Java. Constructores.
2. Entorno de Desarrollo Integrado. Herramientas integradas.

Tema 2: Análisis y Diseño Orientado a Objetos.

1. Desarrollo de software orientado a objetos.
2. Identificación de clases, atributos y operaciones: análisis gramatical.
3. Identificación de colaboración y responsabilidad: CRCs.
4. Contrato de operaciones.
5. Diseño estructural del sistema con UML.
6. Cohesión y acoplamiento.
7. Documentación de programas.
   a. Documentación interna.

b. Documentación externa.
Tema 3: Técnicas de Reutilización de código en Programación Orientada a Objetos. 1. Herencia. a. Concepto y objetivos. b. Tipos de herencia. c. Redefinición de métodos. 2. Polimorfismo a. Concepto y objetivos. b. Ligadura estática vs. ligadura dinámica. c. Usos correctos e incorrectos. d. Lenguajes de POO: soporte de polimorfismo. 3. Genericidad. a. Concepto y objetivos. b. Lenguajes de POO: soporte de genericidad. c. Polimorfismo vs. Genericidad.
Tema 4: Tratamiento de errores. Manejo de Excepciones. 1. Introducción. 2. Objetivos. 3. Conceptos. 4. Lenguajes de POO con soporte de excepciones.
Tema 5: Colecciones de objetos. 1. Introducción y objetivos. 2. Estructuras de datos lineales (listas, pilas y colas). 3. Otras estructuras de datos: grafo y ABB. 4. Ejemplos de frameworks de colecciones de objetos. 5. Contenedores. 6. Iteradores. 7. Algoritmos.
Tema 6: Pruebas de software. 1. Introducción. 2. Objetivos. 3. Tipos de pruebas. 4. Introducción a las pruebas unitarias. 5. Soporte automatizado para las pruebas unitarias: frameworks. 6. Introducción al desarrollo software dirigido por pruebas.
Tema 7: Patrones de Diseño. 1. Introducción. 2. Objetivos. 3. Catálogos de patrones de diseño. 4. Desarrollo software dirigido por patrones. 5. Ejemplos de aplicación: Singleton y Template Method. 6. Antipatrones. Ejemplos.
Tema 8: Interfaces gráficas de usuario. 1. Introducción. 2. Inversión de control. 3. Programación orientada a eventos. 4. Librerías y herramientas de desarrollo de Interfaces gráficas de usuario.

Actividades formativas					
Horas de trabajo del alumno por tema		**Presencial**		**Actividad de seguimiento**	**No presencial**
Tema	**Tota**	**GG**	**SL**	**TP**	**EP**

	I				
1	9	3	2		4
2	18	4	1		13
3	31	5	5	1	20
4	20,5	2,5	3		15
5	20,5	4	4		12,5
6	20	4	4		12
7	22	4	4	1	13
8	8	2	2		4
Evaluación del conjunto	1	1	0	0	0
Total		29,5	25	2	93,5

GG: Grupo Grande (100 estudiantes).
SL: Seminario/Laboratorio (prácticas clínicas hospitalarias = 7 estudiantes; prácticas laboratorio o campo = 15; prácticas sala ordenador o laboratorio de idiomas = 30, clases problemas o seminarios o casos prácticos = 40).
TP: Tutorías Programadas (seguimiento docente, tipo tutorías ECTS).
EP: Estudio personal, trabajos individuales o en grupo, y lectura de bibliografía.

Actividades formativas que se plantearán
Para alcanzar los objetivos de aprendizaje de la asignatura se plantearán las siguientes actividades formativas:

Presenciales en grupo grande
- clase de explicación de conceptos
- clase de ejercicios y problemas
- resolución de ejercicios y problemas
- desarrollo de problemas en común
- presentación de problemas resueltos
- resolución de ejercicios de test

Presenciales en el laboratorio
- ejercicios guiados
- ejemplos de software desarrollado
- implementación de ejercicios de programación
- detección de errores de programas
- uso de librerías externas
- realización del proyecto de programación
- uso de estructuras de datos
- ejecución de pruebas de código
- uso del aula virtual

No presenciales
- estudio de temas
- trabajo con el entorno de trabajo utilizado
- búsqueda de información (libros, Internet, etc.)
- reuniones de grupos
- realización de proyecto de programación
- realización de documentación externa e interna del proyecto de programación
- uso de foros de la asignatura
- resolución de problemas planteados en sesiones teóricas y prácticas

Sistemas de evaluación

Esta asignatura se imparte mediante sesiones presenciales de dos tipos diferenciados: sesiones de teoría y sesiones de práctica.

- En las **sesiones de teoría**, el alumno dispone (con la suficiente antelación) de un documento con apuntes previos sobre el tema que el profesor explicará en la sesión de teoría. Por tanto, la metodología seguida para las sesiones de teoría requiere una lectura y trabajo previo por parte del alumno (la dedicación aconsejada es de aproximadamente media hora) de los apuntes dejados por el profesor para cada uno de los temas vistos en la sesión de teoría. Una vez resueltas las dudas que los alumnos puedan tener sobre el tema de la sesión, el profesor explicará el tema teórico con ejemplos prácticos, de modo que los alumnos puedan asimilar de una manera más sencilla los conceptos que se presentan en el tema de teoría. Se implementarán de una forma práctica durante la sesión ejemplos y ejercicios sobre los conceptos explicados.
- En cuanto a las **sesiones prácticas**, el alumno parte de un guión previo en el que tiene detallados los conceptos que se van a explicar en la sesión, una serie de ejercicios previos aconsejados, así como una serie de ejercicios que deberá realizar durante la sesión. En este guión, se detallarán también los ejercicios propuestos no presenciales que el alumno debe realizar para afianzar los conceptos vistos en la sesión presencial.
  De este modo, los alumnos disponen desde la primera sesión tanto de teoría como de prácticas de una planificación completa de los temas y trabajos prácticos que van a explicarse en cada una de las sesiones durante todo el curso.

Se hará uso constante del espacio de apoyo a la asignatura en el campus virtual. Este espacio recogerá la publicación semanal de los contenidos teóricos y prácticos, así como otra información útil para los alumnos (material bibliográfico aconsejado, manuales de consulta en Internet, ejercicios propuestos, ejercicios resueltos, etc.).

Basándonos en esta metodología, la evaluación de la asignatura se realizará basándose en los siguientes instrumentos de evaluación:

- Proyecto de programación
- Evaluación de conocimientos mínimos adquiridos mediante el proyecto de programación
- Actividades de autoaprendizaje

Estos instrumentos de evaluación incluyen a otros instrumentos de evaluación más simples que nos permitirán la evaluación de las competencias adquiridas en la asignatura.

- **Proyecto de programación**

La evaluación del proyecto de programación es la parte más importante de la asignatura y permitirá evaluar la mayoría de competencias técnicas y transversales desarrolladas por el alumno.
A lo largo del semestre, los alumnos desarrollarán un proyecto de programación en grupo guiado y dividido en partes. Durante el desarrollo del proyecto los grupos tendrán un seguimiento por parte del profesor. Durante las sesiones prácticas y teóricas se impartirán todos los conocimientos y conceptos relacionados con la asignatura. La comprensión de estos conceptos permitirá a los alumnos su posible utilización en el proyecto de programación. Además, durante algunas sesiones teóricas y prácticas se realizará un seguimiento más detallado del trabajo que va desarrollando cada grupo.
El proyecto deberá entregarse debidamente resuelto y documentado en las fechas indicadas. Tanto el desarrollo del proyecto como la documentación del

mismo deben ajustarse a los criterios especificados por el profesorado de la asignatura.
El proyecto se realizará normalmente en grupos de 2 alumnos:

- El trabajo en grupo está pensado para llevarse a cabo durante la evaluación continua de la asignatura, por este motivo sólo se permitirá la formación de grupos al comienzo del semestre.
- Los componentes de cada grupo deberán informar de su composición a los profesores de la asignatura durante las primeras semanas del semestre (se especificará a través del aula virtual).

Las características básicas que se evaluarán dentro de este apartado son las siguientes:

- Utilización correcta de principios básicos de POO: encapsulación, herencia y polimorfismo
- Análisis y Diseño Orientado a Objetos adecuado al problema: estructura correcta de la solución
- Documentación externa del proyecto (adecuada a los criterios especificados por el profesorado, contenido, faltas de ortografía, etc.)
- Documentación interna del proyecto
- Utilización correcta de estructuras de datos y desarrollo de algoritmos
- Genericidad
- Sobrecarga
- Excepciones
- Estilo de programación adecuado
- Utilización correcta de patrones de diseño
- Uso de bibliotecas de funciones estándar
- Habilidad en el uso de herramientas de programación
- Utilización de herramienta de control de versiones

- **Evaluación de conocimientos mínimos adquiridos mediante el proyecto de programación**

Para asegurar la adquisición de los conocimientos y habilidades mínimos de las competencias técnicas se realizarán pruebas escritas o en ordenador que consistirán en la resolución de preguntas tipo test, preguntas cortas, resolución de ejercicios, resolución de problemas prácticos relacionados con el proyecto, etc.

- **Actividades de autoaprendizaje**

Dentro de estas actividades se recogen todas aquellas actividades de carácter eminentemente cooperativo. En concreto, en este curso desarrollaremos las siguientes actividades:

- participación activa en los foros de la asignatura dentro del Aula Virtual de la UEx
- la propuesta de ejercicios extra por parte del alumno
- creación de materiales adicionales por parte del alumno (glosarios)
- participación en debates

Las actividades se irán proponiendo a lo largo del semestre por parte del profesorado.

<u>Relación entre instrumentos de evaluación y objetivos de aprendizaje</u>
En la siguiente tabla se detallan los objetivos de aprendizaje de la asignatura que se cubren con los instrumentos de evaluación propuestos.

	Instrumentos de evaluación		
Objetivos de	Proyecto	Conocimientos teóricos	Actividades de autoaprendizaje
conocimiento			
Ob 1		x	
Ob 2	x	x	
Ob 3	x	x	
Ob 4		x	
Ob 5		x	
Ob 6		x	
Ob 7		x	
Ob 8		x	
comprensión			
Ob 9	x	x	
Ob 10	x	x	
Ob 11	x	x	
Ob 12		x	
Ob 13	x	x	
Ob 14	x	x	
aplicación			
Ob 15	x		
Ob 16	x		
Ob 17	x		
Ob 18	x		
Ob 19	x		
Ob 20	x		
Ob 21	x		x
análisis			
Ob 22	x		x
Ob 23	x		x
Ob 24	x	x	x

Criterios de evaluación

Cada estudiante podrá ser calificado en la asignatura atendiendo a dos modalidades diferentes: Evaluación continua y Evaluación final.

En ambos casos, para aprobar la asignatura el alumno debe:

- Desarrollar el proyecto de programación en grupo sobre un supuesto práctico que el profesorado propondrá al comienzo de la asignatura.
- Superar la prueba de conocimientos mínimos adquiridos durante el proyecto.

Evaluación continua.

Para superar la asignatura por evaluación continua, el estudiante deberá realizar todas las entregas del proyecto de programación (se permitirá que no

se entregue una de ellas). Además, deberá superar la prueba de conocimientos mínimos del proyecto que se realizará durante la semana de la última entrega. Si un estudiante no supera esta prueba o no cumple los requisitos de entregas mínimas para superar la asignatura por evaluación continua, tendrá otra oportunidad para superar la asignatura por evaluación final. Asimismo, aquellos estudiantes que deseen subir nota en su proyecto y hayan superado la asignatura por evaluación continua, podrán entregar el proyecto mejorado en la convocatoria oficial de la asignatura.

Evaluación final.

Aquellos estudiantes que no hayan cumplido los requisitos para superar la asignatura por evaluación continua deberán entregar el proyecto de programación completo en la convocatoria oficial de la asignatura. En dicha convocatoria deberán superar, además, la prueba de conocimientos mínimos del proyecto de programación.

- La evaluación de la asignatura (en cualquiera de las modalidades) se realizará atendiendo a **3 bloques de calificación** diferentes.
- La puntuación de cada bloque se calculará sobre 10.
- La calificación en un bloque (sub-bloque o prueba de conocimientos mínimos) superado (cumplidos los requisitos mínimos) se guardará durante todas las convocatorias de ese curso, siempre que el estudiante tenga derecho a examen en la convocatoria que supera el bloque.

Bloque 1: proyecto de programación

- Este bloque es recuperable.
- Es obligatorio superar este bloque con una nota mínima de 5.
- Su calificación es un 45% de la calificación final de la asignatura.
- Es responsabilidad del alumno o grupo de alumnos la custodia y protección de su proyecto.
- Se utilizará un software específico de detección de copias en programas.
- **Evaluación de los conocimientos mínimos adquiridos mediante el proyecto de programación:**

  Para superar el proyecto de programación, además de obtener una calificación mínima de 5, el estudiante deberá superar esta prueba de conocimientos mínimos que será evaluada como Apto / No apto.

Bloque 2: evaluación de componentes del grupo

- Este bloque es recuperable.
- No hay una nota mínima para este bloque.
- Su calificación es un 35% de la calificación final de la asignatura.
- La calificación de este bloque se realizará mediante “bolsa de notas” sobre el proyecto de programación (bloque 1), es decir, recibirá una calificación que será multiplicada por el número de componentes del grupo (normalmente 2) y los componentes del grupo decidirán la calificación que les corresponde a cada uno de ellos.

  Ejemplo: si un proyecto obtiene una calificación de “7” al grupo se le asignará una calificación de “14” y los miembros del grupo podrán “repartirse” dicha calificación de diferentes formas: “7+7” ó “8+6” ó “9+5” ó “10+4”.

Bloque 3: actividades de autoaprendizaje

- Este bloque es no recuperable.
- No es obligatorio superar este bloque con una nota mínima.

- La nota de este bloque se obtiene sumando las puntuaciones obtenidas en las actividades propuestas realizadas.
- Cada actividad propuesta recibirá una puntuación entre 1 y 10 hasta completar un máximo de 30 puntos.
- La calificación de este bloque será un 20% de la calificación total en la asignatura.

Cálculo de la calificación final del alumno
Si se cumplen los requisitos mínimos para el bloque 1 y 3, la nota se calculará según la siguiente fórmula:

Nota final = Bloque1 * 0,45 + Bloque2 * 0,35 + Bloque3 * 0,20

Si no se cumplen los requisitos mínimos para el bloque 1, la calificación en esa convocatoria será "Suspenso – 3".

La copia o el plagio demostrados en cualquier actividad supone una nota final de SUSPENSO (0) en la convocatoria y una nota de 0 en los bloques no recuperables para todos los implicados, además de las actuaciones legales indicadas según la normativa vigente.

Sistema de revisión y comentario de exámenes
El alumno podrá comentar y revisar los resultados de las actividades recuperables en las fechas previstas de acuerdo a la normativa vigente, para los exámenes de convocatorias oficiales.
Para el resto de actividades no recuperables, la revisión se realizará en horario de clases o en el horario de tutorías de los profesores.

Bibliografía y otros recursos

Bibliografía básica:

- Bertrand Meyer, *Construcción de Software Orientada a Objetos. 2ª Edición*, Ed. Prentice Hall
- Roberto Rodríguez, Encarna Sosa, Álvaro Prieto, *Programación Orientada a Objetos*. Editado por Librería Álvaro. http://www.libreriaalvaro.com/libropoo.html Licencia Creative Commons.
- David Barnes, *Programación orientada a objetos con Java, 3ª ed.: una introducción práctica usando BlueJ.* Ed. Prentice Hall
- Roberto Hernández, Juan Carlos Lázaro, Raquel Dormido, Salvador Ros. *Estructuras de Datos y Algoritmos*. ED. Prentice Hall.
- Bruce Eckel, *Piensa en Java, 4ª Edición.* Ed. Pearson

Bibliografía adicional:

- Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides, *Design Patterns, Elements of Reusable Object-Oriented Software. Ed. Addison-Wesley*
- Francisco J. Ceballos, *JAVA 2, Interfaces gráficas y aplicaciones para internet, 2ª Ed. Ed. Ra-Ma.*
- Andrew Hunt, David Thomas, *The Pragmatic Programmer: from journeyman to master.*
- Brett D. McLaughlin, Gary Pollice, Dave West, Head First Object-Oriented Analysis and Design, Ed. O'Reilly Media
- Ed Burnette, *Eclipse IDE pocket guide.* Ed. O'Reilly Media

Otros recursos

Medios materiales utilizados:

- Teoría: aula, pizarra, ordenadores portátiles de los alumnos y cañón de vídeo.
- Práctica: laboratorio de ordenadores (1 ordenador por alumno) con todas las herramientas software de la asignatura correctamente instaladas, pizarra, cañón de vídeo y aula virtual.

Materiales y recursos utilizados:

Todo el material y recursos utilizados en la asignatura están disponibles en el aula virtual de la misma:

- Transparencias para cada tema de teoría.
- Guiones de las sesiones de laboratorio.
- Planificación del curso.

Los recursos propios del aula virtual que se utilizarán en la asignatura son los siguientes:

- Sistemas de participación:
  - Foros de comunicación.
  - Tablón de anuncios y novedades.
  - Foros de debates.
  - Foros de ejercicios no presenciales.
- Información adicional:
  - Glosario de términos y palabras clave.
  - Conjunto de referencias web relacionadas con los contenidos de la asignatura.
  - Tutoriales y vídeos explicativos.
- Autoevaluación:
  - Test de conocimientos previos de la asignatura.
  - Test de autoevaluación de contenidos.
  - Problemas de autoevaluación.
- Tareas virtuales para la entrega de problemas.

Además, en la biblioteca del centro existen ejemplares de los libros aconsejados en la bibliografía. Los manuales y enlaces digitales podrán ser consultados y/o descargados durante las sesiones prácticas, en las cuales se dispone de acceso a internet.

Recursos virtuales

Se utilizará de una forma constante el espacio de apoyo a asignaturas presenciales del campus virtual de la UEx como apoyo a la docencia de la asignatura, tanto para el seguimiento de las sesiones como para la realización y seguimiento de cualquier clase de actividad o ejercicio propuesto durante todo el curso. Las entregas de actividades, ejercicios, controles periódicos, proyecto de programación y modificaciones al proyecto se realizarán también utilizando dicha plataforma virtual. Se utilizarán foros informativos para comentar, fomentar el debate y discutir sobre todos los aspectos relacionados con la asignatura; así como para anunciar posibles novedades sobre la asignatura.

Horario de tutorías

Tutorías Programadas: el horario de tutorías programadas se publicará al comienzo del semestre. Aproximadamente se impartirá una hora por alumno durante la cuarta semana del semestre y otra hora durante la décima semana del semestre.

Tutorías de libre acceso: dado que el centro requiere al profesorado su horario de tutorías durante el mes de comienzo del semestre, el horario de los profesores será publicado en esas fechas (en este caso, septiembre).
Además, la comunicación entre profesor-estudiante y estudiante-estudiante será continua a lo largo de todo el curso mediante los diferentes canales de comunicación electrónicos utilizados en la asignatura: aula virtual, correo electrónico y redes sociales (@dpuex).

Recomendaciones

Para cursar adecuadamente esta asignatura se recomienda:

- Haber superado las asignaturas de Introducción a la programación y Estructuras de Datos y de la Información de primer curso del mismo grado.
- Cursar en paralelo o haber cursado la asignatura del mismo semestre Análisis y Diseño de Algoritmos.
- Consultar y utilizar la bibliografía o los recursos adicionales recomendados en la asignatura.
- Seguir la asignatura según la planificación establecida por el profesorado de la misma, poniendo especial atención a:
    - Resolución de problemas propuestos.
    - Realizar un trabajo constante durante todo el semestre, realizando el trabajo propuesto en los guiones previos de las sesiones prácticas para poder aprovechar al máximo estas sesiones, y avanzando el proyecto final de programación.
    - Asistencia regular a clase.
    - Acceso regular al aula virtual de la asignatura y participación activa en las actividades propuestas en el mismo.
    - Utilizar las tutorías del profesorado para resolver dudas.
    - Pensar detenidamente con quién se va a formar el grupo; ya que, el desempeño de cada uno de los miembros del grupo repercute sobre el resto.
- Llevar ordenador portátil a las clases de teoría.

Horas de estudio recomendadas
El número mínimo de horas que un estudiante medio debe dedicar a la asignatura para superarla se estima alrededor de 4,5 horas a la semana de trabajo personal fuera de las sesiones teóricas y prácticas programadas.