UNIVERSIDADE DA CORUÑA

GUÍA DOCENTE

Programación general de la asignatura

Curso académico

2008/2009

Asignatura		
Programación Orientada a Objetos		
Curso	Ciclo	Profesor responsable del programa
2º / 3º	1er ciclo	Eduardo Mosqueira Rey
Titulación		
Ingeniería Informática Ingeniería Técnica en Informática de Gestión Ingeniería Técnica en Informática de Sistemas		
Centro		
Facultad de Informática		

Curso académico	2008/2009

DATOS DESCRIPTIVOS	
Código titulación	**Titulación**
614-111 614-211 614-311	Ingeniería Informática Ingeniería Técnica en Informática de Gestión Ingeniería Técnica en Informática de Sistemas

Plan de estudios		
Fecha de homologación	Fecha publicación BOE	Curso de implantación
28-09-1994	**23-11-1994**	**1994/1995**

Asignatura		
Código asignatura	Nombre	
636	Programación Orientada a Objetos	
Curso	Ciclo	Idiomas en que se imparte
2º / 3º	1er ciclo	Castellano
Carácter	Duración	convocatoria
Optativa	Cuatrimestral	Ordinaria (Febrero), Septiembre y Fin de Carrera-Febrero

Créditos asignatura		
Tipo	**LRU**	**ECTS**
Teóricos	4	
Prácticos	2	
Totales	6	4

Departamento	
Código	Nombre
103	Computación

Área	
Código	Nombre
075	Ciencias de la Computación e Inteligencia Artificial

Centro / Facultad / Escuela			
Código	Nombre		
614	Facultad de Informática		
Campus	Calle	Nº	Código postal
Elviña		s/n	15071
Teléfono	Fax	E-mail	
981.167.000	981.167.160		

Descriptores de la asignatura
Encapsulación, Jerarquía, Herencia, Métodos

Profesorado y tutorías			
Profesor 1			
Nombre	Despacho	Extensión	Email[1]
Eduardo Mosqueira Rey	**3.17**	**1343**	**eduardo@udc.es**
Tutorías[2]			
1º Cuatrimeste		**2º Cuatrimeste**	
Días semana	Hora	Días semana	Hora
Martes	10:30 – 11:30	Lunes	10:30 – 12:30
Miércoles	10:30 – 11:30	Martes	10:30 – 12:30
Miércoles	12:30 – 13:30	Jueves	10:30 – 12:30
Jueves	12:30 – 13:30		
Viernes	10:30 – 12:30		

[1] La dirección de correo no deberá utilizarse para preguntar dudas sobre la asignatura, para eso están las tutorías virtuales que pueden realizarse a través de la facultad virtual: http://fv.udc.es.

[2] Las horas de tutorías pueden variar a lo largo del curso, la versión actualizada de las mismas estará siempre disponible en la página web de la Faculta de Informática: http://www.fic.udc.es

Profesorado y tutorías			
Profesor 2			
Nombre	Despacho	Extensión	Email
...	**...**	**...**	**...**
Tutorías[2]			
1º Cuatrimeste		**2º Cuatrimeste**	
Días semana	Hora	Días semana	Hora

PROGRAMA GENERAL DE LA ASIGNATURA

Prerrequisitos

Se recomienda haber cursado con anterioridad las asignaturas de primero "Programación" y "Estructura de datos y de la Información" así como otras asignaturas que puedan tocar en mayor o menor medida la orientación a objetos, el lenguaje Java o el modelado UML (p. ej. "Metodología de la Programación", "Principios de Análisis Informático", etc.).

Contexto

La programación orientada a objetos (POO) es, hoy en día, el paradigma de programación dominante en el desarrollo de sistemas informáticos. La POO surge como un paso más en la evolución de la programación imperativa añadiendo nuevas propiedades (como herencia, polimorfismo, etc.) a los tipos abstractos de datos. El carácter optativo de la asignatura puede llevar a engaño ya que la relativa antigüedad de los planes de estudio (1994) y de las troncalidades (1991) no recogen correctamente la situación actual en el ámbito de la programación (Java sólo tiene poco más de 10 años de antigüedad).

La filosofía de aprendizaje marcada en el plan de estudios se configura como "Imperativo primero", es decir, primero se explican los lenguajes imperativos para dar paso más adelante a la programación orientada a objetos. De esta forma el aprendizaje de la programación se hace más gradual, aunque el paso a los conceptos de la orientación a objetos obliga a un cambio de filosofía en la forma de programar cuya adaptación puede resultar compleja al alumno.

Las competencias académicas que se pretenden desarrollar son importantes para cursar otras materias ligadas directa o indirectamente con la programación. Entre las más directamente relacionadas podemos destacar: (II, ITIG e ITIS) Proyecto fin de Carrera, (II) Análisis de Sistemas Informáticos, Diseño de Sistemas Informáticos, Integración de Sistemas, Análisis y Diseño Orientado a Objetos, e (ITIG) Principios de Análisis Informático.

Esta materia también resulta muy interesante a la hora de configurar habilidades en el contexto del ámbito profesional ya que el paradigma de la orientación a objetos es el dominante dentro de los lenguajes de programación más utilizados profesionalmente (Java, C#, C++, VisualBasic, Delphi, etc.).

Objetivos

- Analizar cómo la evolución de los tipos abstractos de datos nos lleva al desarrollo del modelo de orientación a objetos y al cambio de una metodología top-down por una metodología bottom-up.
- Familiarizarse con la filosofía de la orientación a objetos y con los conceptos básicos de clases, objetos, propiedades y métodos.
- Analizar las principales propiedades de la orientación a objetos incluyendo aquellas que son compartidas por los tipos abstractos de datos (como abstracción, encapsulación, modularidad), como aquellas propias de la orientación a objetos (herencia, polimorfismo, ligadura dinámica, etc.).
- Conocer las características básicas del lenguaje de modelado visual UML y aprender a establecer una relación bidireccional existente entre un diseño detallado UML y el código final resultante.
- Conocer los principios de diseño que marcan lo que se considera un "buen" desarrollo orientado a objetos (como el principio abierto-cerrado, el principio de substitución de Liskov, etc.) y analizar su aplicación en supuestos prácticos.
- Familiarizarse con el concepto de patrones de diseño y conocer los principales patrones que se utilizan en el diseño de programas orientados a objetos.
- Desarrollar, a partir de la definición de un problema, un diseño detallado en UML de los aspectos estáticos y dinámicos de los objetos del dominio y plasmar este diseño en un código de un lenguaje de programación orientado a objetos como Java.

Competencias

- Capacidades conceptuales (saber teórico). El alumno ha de ser capaz de:
  - Conocer las diferencias entre el paradigma de programación imperativa tradicional y el paradigma de la orientación a objetos.
  - Comprender los conceptos básicos en los que se basa la orientación a objetos: clases, objetos, mensajes, etc.
  - Analizar, comprender y relacionar las propiedades básicas propias de la orientación a objetos: herencia, polimorfismo, ligadura dinámica, etc.
  - Conocer los aspectos básicos del diseño UML, sobre todo lo que involucra el diseño estático (diagrama de clases) y dinámico (diagrama de secuencia) de un programa orientado a objetos.
  - Conocer, comprender e identificar los principios y patrones de diseño básicos para lograr un programa orientado a objetos flexible y reusable.
- Capacidades procedimentales (saber práctico y metodológico). El alumno ha de ser capaz de:
  - Construir clases básicas que representen objetos del dominio y que encapsulen el estado y el comportamiento de dichos objetos.
  - Desarrollar y organizar clases sacando provecho de las propiedades propias de la orientación a objetos para desarrollar código flexible y reusable.
  - Diseñar un programa orientado a objetos utilizando el estándar UML y desarrollar posteriormente el código resultante de dicho diseño.
  - Diseñar un programa utilizando principios y patrones de diseño garantizando así que las propiedades propias de la orientación a objetos se utilizan adecuadamente.
  - Usar la programación orientada a objetos para resolver problemas reales.
  - Manejar con solvencia un entorno integrado de desarrollo (IDE) en la programación
- Capacidades actitudinales (saber social). El alumno ha de ser capaz de:
  - Asumir la responsabilidad del trabajo y las acciones propias.
  - Valorar y respetar el trabajo propio y el ajeno.
  - Colaborar con los compañeros para llevar a cabo una práctica en grupo.
  - Valorar la importancia del análisis y el diseño en el desarrollo de programas orientados a objetos.
  - Preferir las buenas prácticas de programación orientada a objetos sobre los antipatrones (malas prácticas) propias de los principiantes o de los malos programadores.

Contribución al desarrollo de habilidades y destrezas

- Desenvolverse con soltura en el manejo de herramientas de desarrollo de programas orientados a objetos.
- Adquirir la experiencia necesaria para obtener rápidamente la respuesta a ciertos problemas sin necesidad de profundizar en los principios básicos de la disciplina
- Desarrollar el pensamiento creativo para resolver problemas
- Desarrollar la capacidad de adaptarse al entorno de trabajo y hacer frente a los cambios en las técnicas y conocimientos de la disciplina.

Contenidos (temario teórico y práctico)		
Título	**Cronograma[a]**	
	Inicio	**Fin**
Temario teoría		
Presentación		
T1: Introducción		
T2: Elementos básicos de la orientación a objetos		
T3: Propiedades básicas de la orientación a objetos		
T4: Modelado visual de objetos: UML		
T5: Principios de diseño		
T6: Patrones de diseño		
Temario prácticas (PPx = clases presenciales)		
CP1: Introducción Java y NetBeans		
CP2: Introducción a JUnit		
P1: Boletín 1: Ejercicios básicos de Java		
P2: Boletín 2: Ejercicios sobre los elementos básicos de la orientación a objetos		
P3: Boletín 3: Ejercicios sobre los elementos avanzados de la orientación a objetos		
CP3: Introducción a Robocode		
P4: Práctica de diseño: Practica final de diseño usando Robocode		

[a] El cronograma de la asignatura puede consultarse en la Facultad Virtual en el fichero "Calendario POO"

Metodología

En las clases presenciales de teoría, el profesor realizará una breve descripción de los contenidos temáticos y de los objetivos básicos perseguidos, con el fin de dotar al alumno de una visión global de la materia. Además se tratará de establecer interrelaciones con otros conceptos previamente adquiridos, de forma que se pueda establecer una línea temporal, y expondrá la bibliografía recomendada. Seguidamente pasará a desarrollar los contenidos teóricos utilizando como método la clase magistral.

Dentro de las clases presenciales también se incluirán clases o ejercicios de resolución de problemas que serán resueltos "in situ" con la colaboración activa de los alumnos. Estas clases o ejercicios se plantean como una forma de afianzar los conceptos teóricos explicados por el profesor y tratan de fomentar la participación de los alumnos, el diálogo abierto y la valoración de soluciones.

Como actividades no presenciales para las clases de teoría se planteará la realización de boletines de ejercicios (dentro del horario de prácticas) que el alumno deberá resolver de forma individual y, posteriormente, comentar o corregir con el profesor durante las horas de tutorías. También, al finalizar cada tema, se proporcionará al alumno un test de autoevaluación para que pueda comprobar el progreso de su aprendizaje.

Las clases de prácticas consistirán principalmente en el desarrollo de una práctica final de la asignatura en la que se intentará ejercitar todos los contenidos expuestos en las clases teóricas. El enunciado de la práctica, que se proporcionará con la suficiente antelación, detallará el problema y las especificaciones que el alumno deberá respetar estrictamente. Posteriormente, la labor del profesor consistirá en la resolución de errores, solución de dudas, corrección de malos hábitos, etc. durante las horas de tutoría.

Distribución ECTS

- 4 Nº créditos ECTS x 27 = 108 horas curso.

Actividad académica	Tipo de actividades	A	F (1)	B	C	D
		Horas presenciales	Factor estimado de horas no presenciales	Horas no presenciales	Horas totales (A + B)	Créditos ECTS (C ÷ 27)
Clases teóricas de aula	Clases magistrales	25	1	25	**50**	1,85
	Clases de ejercicios	2	1	2	**4**	0,15
Clases prácticas	Boletín 3			12	**12**	0,44
	Práctica de diseño	2		12	**14**	0,52
Trabajo tutelado	Boletines 1 y 2	4		12	**16**	0,59
Seminarios, conferencias, congresos, talleres y/o simposios						
Tutorías	Tutorías Clases teóricas de aula	1			**1**	0,04
	Tutorías clases prácticas	1			**1**	0,04
Estudio de preparación de exámenes	Clases teóricas de aula	4			**4**	0,15
	Clases prácticas	4			**4**	0,15
Realización de exámenes	Examen clases teóricas	2			**2**	0,07
	Examen prácticas					
Revisión de exámenes	Examen clases teóricas	0,1			**0,1**	0,004
	Examen prácticas	0,1			**0,1**	0,004
Total		**45,2**		**63**	**108,2**	(2) **4,01**

(1) [F] Si no dispone de una medida directa, puede utilizar el **"Factor estimado de horas no presenciales"** o "**factor de estimación de trabajo personal"**. Define el número de horas de trabajo personal o independiente del alumnado (organización de apuntes, estudio, documentación, preparación de seminarios, etc.) por cada hora presencial que realiza respecto a cada una de las actividades.

- De acuerdo al informe técnico "El crédito Europeo y el Sistema Educativo Español" se propone para la estimación de horas de trabajo personal del alumnado aplicar la siguiente equivalencia:
  - **1 hora presencial de teoría** → 1.5 - 2 h de trabajo independiente del alumnado o trabajo personal.
  - **1 hora presencial de prácticas** → 0,75 - 1 h. de trabajo independiente del alumnado o trabajo personal.

(2) Si no lo ha hecho previamente, compruebe si el resultado es razonable según la tabla de simulación que acompañamos para cada titulación. Recuerde que se trata de un ejercicio teórico sin otro sentido o validez.

Recursos
Bibliografía básica: [1]
• Eckel, B.. Piensa en Java. Prentice-Hall, Madrid. 4ª Edición. 2007. Signatura FIC: D32 (Jav) ECK. • Sierra, K., Bates, B. Head First Java, O'Reilly, Sebastopol, CA, 2nd Edition, 2005. • Cohoon, J., Davidson, J., Programación en Java 5.0, McGraw-Hill, Madrid, 2006. • Booch, G., Rumbaugh, J. y Jacobson. El lenguaje unificado de modelado, 2ª Ed.. Addison-Wesley, Madrid. 2006. Signatura FIC: D22 BOO. • Gamma, E. et al. Design patterns: elements of reusable object oriented software. Reading, MA. Addison-Wesley. 1995. FIC: D211 DES - D123 PAT (español).
Bibliografía complementaria:
• McLaughlin, B., Flanagan, D., Java 1.5 Tiger: A Developer's Notebook, O'Reilly, Sebastopol, CA, 2004. • Martin, R.C. UML para programadores Java, Pearson, Madrid, 2004. • Rumbaugh, J., Jacobson, I., Booch, G., El lenguaje unificado de modelado: Manual de referencia, Addison-Wesley Iberoamericana, Madrid, 2000. • Budd, T.. An introduction to object-oriented programming. Pearson. 3Edición. 2002. Signatura FIC: D15 BUD. • Knoernschild, K.. Java desing: objects, UML and process. Addison-Wesley, Boston, MA. 2002. Signatura FIC: D32 (Jav) KNO.
Recursos web:
El principal recurso web de la asignatura es la página de la misma existente en la facultad virtual de la UDC (http://fv.udc.es). En dicha página se encontrará toda la información necesaria para el desarrollo de la asignatura (transparencias, boletines de ejercicios, prácticas, tests, preguntas frecuentes, links en Internet, etc.). El acceso a dicha página está limitado a los alumnos matriculados en la asignatura. Para información general sobre Java se recomienda acudir a la página web oficial del lenguaje (http://java.sun.com) en donde podremos acceder a noticias, documentación, descarga del herramientas, etc. Para estar al día sobre las novedades del lenguaje se recomienda acceder a páginas de noticias como por ejemplo JavaHispano (http://javahispano.org).
Otros materiales de apoyo:
La biblioteca de la Facultad dispone de varias revistas científicas que están a disposición del alumno, y que revisten interés para la profundización en algún aspecto particular del temario, o para disponer de actualizaciones (estados del arte) de alguna de las técnicas y metodologías expuestas en clase. Algunas de las más utilizadas en la asignatura de POO son: • Communications of the ACM • Journal of Object Oriented Programming (ya no se publica pero hay números antiguos) • IEEE Transactions on Knowledge and Data Engineering • Data and Knowledge Enginnering • ACM transactions on programming languages and systems • Lecture Notes in Computer Science (aquellos volúmenes que hacen referencia a congresos sobre objetos como ECOOP – European Conference on Object Oriented Programming).

[1] En la Facultad Virtual tienes disponible un documento con bibliografía comentada y su disponibilidad detallada en la Facultad de Informática

Evaluación

Consideraciones generales:

La evaluación del alumno consistirá en una evaluación formativa y una evaluación sumativa. La evaluación formativa se realizará a través de los boletines de ejercicios y la práctica final de la asignatura. La evaluación sumativa consistirá en un examen escrito al término del cuatrimestre.

Los boletines de ejercicios están pensados para realizar un trabajo individual que refuerce los conceptos vistos en teoría y se corresponden con los temas de teoría T1, T2 y T3; y con los temas de prácticas. La práctica final de la asignatura está pensada para fomentar el trabajo en grupo y consolidar los conocimientos utilizados en los boletines de ejercicios así como las explicaciones de teoría pertenecientes a los temas T4, T5 y T6.

Por último, el examen escrito al término del cuatrimestre servirá para demostrar que el alumno ha adquirido los conocimientos necesarios sobre la programación orientada a objetos y se ha entrenado lo suficiente como para poseer las habilidades precisas para resolver supuestos prácticos sobre la materia.

Aspectos y criterios de evaluación:

Examen escrito:
- Peso: 40% (convocatoria ordinaria)
- Carácter: Obligatorio (nota mínima 4 sobre 10)
- Validez: La nota del examen escrito se guarda para las convocatorias de Septiembre y Diciembre celebradas dentro del mismo año, pero sólo en el caso de que la nota sea igual o superior a 5.
- Criterios de evaluación: El examen escrito representará el examen final de la asignatura. Generalmente será un examen tipo test (debido al elevado número de alumnos en la materia). En convocatorias extraordinarias podría plantearse la posibilidad de realizar exámenes de respuesta libre, lo que sería puesto en conocimiento de los alumnos.

Práctica de diseño:
- Peso: 30%
- Carácter: Trabajo Obligatorio realizado en grupo (se exige nota mínima 4 sobre 10)
- Validez: La nota obtenida en la práctica final de una convocatoria tiene validez (se guarda) para todas las convocatorias celebradas dentro del mismo año (Septiembre y Diciembre) sin que el alumno tenga que hacer nada. Incluso se guardan las notas inferiores a 5 ya que la práctica es la misma en todas las convocatorias y la información ya está disponible al profesor en los directorios del CECAFI. Si se quiere mejorar la nota es necesario volver a entregar la memoria con las mejoras.
- Criterios de evaluación: La nota tendrá en cuenta no sólo el código desarrollado sino también la calidad de la memoria entregada, la calidad del análisis y diseño realizado, la calidad de los modelos UML desarrollados, etc. Más detalles en las instrucciones de la propia práctica.

Boletines de ejercicios:
- Peso: 30%
- Carácter: Trabajo voluntario e individual
- Validez: La nota obtenida, en caso de que sea igual o superior a 5, se guarda para las siguientes convocatorias celebradas dentro del mismo año (Septiembre y Diciembre).
- Criterios de evaluación: En la convocatoria ordinaria la evaluación de los ejercicios será siguiendo los criterios estadísticos ECTS (quién haga más ejercicios correctos más nota tendrá). Para evitar copias se penalizará con un cero en un boletín en el caso de encontrar un único ejercicio copiado en el mismo (tanto el original como la copia). Mas detalles en las instrucciones de los boletines.
- Convocatorias extraordinarias: Para las convocatorias extraordinarias de Septiembre y Diciembre **se planteará una práctica alternativa**, aprovechando parte del trabajo realizado. La evaluación dependerá del análisis del código de dicho programa.

Notas importantes:
- Si no se alcanza la nota mínima exigida en alguna de las partes la nota final del alumno no podrá ser superior a 4,5.
- Las notas de los boletines de ejercicios o de la práctica final se publicarán después del examen final debido a la imposibilidad de corregirlas con anterioridad.
- Las entregas de los boletines de ejercicios son de forma escalada, los alumnos de diciembre tienen la posibilidad de entregar todos los boletines de ejercicios al final del cuatrimestre.

Orientaciones para el estudio:

El estudio de la asignatura no debe plantearse como una actividad de estudio memorístico de los temas presentados en clase, sino que deberá tener una orientación más práctica. Por ejemplo, se debe saber qué es la ligadura dinámica, pero también se debe saber poner un ejemplo en Java de la misma o reconocer su aparición en un código Java.

Por otro lado hay que tener en cuenta que una asignatura como la programación orientada a objetos incluye un gran número de nuevos términos o conceptos (polimorfismo, genericidad, patrones, etc.) que no sólo hay que saber aplicar en la práctica sino que también hay que conocer en qué consisten, cuáles son sus ventajas e inconvenientes, cuándo es recomendable utilizarlos. Todos estos aspectos deben ser objeto de estudio por parte del alumno, si bien se intentará evitar en el examen final preguntas que prioricen el aprendizaje memorístico.

En cuanto a los boletines de ejercicios y a la práctica final se recomienda encarecidamente al alumno que no haga uso de actividades académicas deshonestas como pueden ser la copia o préstamo de ejercicios o prácticas ya que pueden llevar al suspenso automático de la asignatura (tanto del que ha copiado como del que ha permitido que le copien).

Pautas para la mejora y la recuperación:

Las recomendaciones para mejorar o recuperar un examen suspenso son las siguientes:

- Ir a la revisión de exámenes y conocer en qué se ha fallado
- Ir a tutorías y preguntar las dudas que se tengan al profesor, no ir con dudas significativas al examen. Sois alumnos por lo que la única pregunta *estúpida* es aquella que no se hace.
- Intentar mejorar la nota de prácticas mejorando aquellos aspectos en los que se ha tenido menor puntuación
- Realizar más ejercicios de los boletines intentando relacionar la resolución de los ejercicios con los aspectos teóricos vistos en clase.

HORARIOS DE CLASE

Referencias	Ejemplo
Grupo: Teoría, Práctica, Laboratorio, Clínica **Nº de grupo:** 01, 02…. **Intervalo de letras:** (A – L) (M – Z) **Aula:** Aula 02, Aula Informática, laboratorio 03,…	Hora: 09-11 / Lunes: Teoría 01 (A – K) Aula 02

Profesor/a 1					
Nombre				Lengua	
Eduardo Mosqueira Rey				**Castellano**	
Horarios 1º Cuatrimestre					
Hora	Lunes	Martes	Miércoles	Jueves	Viernes
08:30 – 09:30					
09:30 – 10:30					
10:30 – 11:30	Práctica 01 Laboratorio 1.2			Práctica 04 Laboratorio 1.2	
11:30 – 12:30	Práctica 01 Laboratorio 1.2	Teoría Aula 3.0	Teoría Aula 3.0	Práctica 04 Laboratorio 1.2	
12:30 – 13:30	Práctica 03 Laboratorio 1.2	Práctica 02 Laboratorio 1.2			
13:30 – 14:30	Práctica 03 Laboratorio 1.2	Práctica 02 Laboratorio 1.2			

HORARIOS DE EXAMEN

Convocatorias y horarios[1]

Convocatoria	Día	Hora	Centro	Aula
Extraordinaria -Diciembre	02/12/2008	11:30	Facultad de Informática	Exámenes
Ordinaria	29/01/2009	16:00	Facultad de Informática	Exámenes y A3.4
Extraordinaria - Septiembre	02/09/2009	11:30	Facultad de Informática	Exámenes y A3.4

[1] El profesor no se hace responsable de los posibles cambios que el decanato pueda decidir sobre estas fechas. Para una información actualizada consultar la web de la facultad: http://www.fic.udc.es