Universitat Autònoma de Barcelona

2019/2020

Programación Avanzada

Código: 104338
Créditos ECTS: 6

Titulación	Tipo	Curso	Semestre
2503758 Ingeniería de Datos	FB	1	2

Contacto

Nombre: Ernest Valveny Llobet
Correo electrónico: Ernest.Valveny@uab.cat

Uso de idiomas

Lengua vehicular mayoritaria: catalán (cat)
Algún grupo íntegramente en inglés: No
Algún grupo íntegramente en catalán: Sí
Algún grupo íntegramente en español: No

Prerequisitos

La asignatura no tiene ningún prerrequisito oficial. De todas formas, se asume que el estudiante ha cursado la asignatura previa de Fundamentos de programación y, por tanto, está familiarizado con las estructuras básicas de la programación.

Objetivos y contextualización

Esta asignatura debe verse como la continuación lógica de la asignatura de Fundamentos de Programación. El objetivo básico es ampliar las nociones básicas de programación introducidas en Fundamentos de Programación y completarlas con los principios de la programación orientada a objetos (POO).

La programación orientada a objetos es una de las metodologías más extendidas de diseño y programación. En la programación orientada a objetos el software se organiza en clases que contienen métodos (procedimientos) y atributos (datos). Las instancias u objetos de estas clases se envían mensajes unas a otras (hacen "llamadas" a métodos de otros objetos) consiguiendo así la funcionalidad deseada. Aparte del concepto de clase, otros tres elementos fundamentales de la POO son la herencia, la composición y la separación de las partes pública y privada de las clases.

Durante el curso explicaremos todos estos conceptos de la POO y veremos cómo los podemos utilizar para construir soluciones que sean fácilmente extensibles en el futuro, ya que una ley básica del software es que los cambios son inevitables. Por ello explicaremos también un conjunto de principios de diseño o heurísticas que nos dicen cómo deben utilizarse los elementos de la POO. Y sobre ellos, todavía se construyen los patrones de diseño, que son soluciones a nivel de diseño OO a problemas recurrentes, que aparecen a menudo al resolver problemas aparentemente independientes.

De esta forma, los objetivos formativos de la asignatura son los siguientes:

- Entender y aplicar correctamente los principios básicos de la programación orientada a objetos: concepto de clase, objeto y encapsulamiento de datos.
- Entender y aplicar a problemas reales los principios de herencia y composición de clases.
- Conocer y aplicar los principios y patrones de diseño a la resolución de problemas de programación.
- Adquirir la capacidad de diseño y programación de algoritmos para la resolución de problemas complejos,

- Utilizar las estructuras de programación y de representación de la información más adecuadas para la implementación de un algoritmo.
- Entender el ciclo de vida del software para la resolución de problemas de programación: análisis del problema, diseño, implementación y prueba.
- Desarrollar los programas siguiendo unas normas de estilo tendentes a conseguir programas de calidad.

Competencias

- Buscar, seleccionar y gestionar de manera responsable la información y el conocimiento.
- Demostrar sensibilidad hacia los temas éticos, sociales y medioambientales.
- Diseñar soluciones algorítmicas eficientes para problemas computacionales, implementarlas en forma de desarrollo de software robustos, estructurados y fáciles de mantener, y verificar su validez.
- Evaluar de manera crítica el trabajo realizado.
- Planificar y gestionar el tiempo y los recursos disponibles.
- 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.

Resultados de aprendizaje

1. Aplicar estrategias de depuración, prueba y corrección de programas.
2. Aplicar los principios básicos de la estructura y la programación de los computadores.
3. Buscar, seleccionar y gestionar de manera responsable la información y el conocimiento.
4. Demostrar sensibilidad hacia los temas éticos, sociales y medioambientales.
5. Desarrollar programas con un buen estilo de programación y bien documentados.
6. Evaluar de manera crítica el trabajo realizado.
7. Implementar en un lenguaje de programación problemas algorítmicos de dificultad media.
8. Planificar y gestionar el tiempo y los recursos disponibles.
9. 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.

Contenido

1. Repaso de estructuras básicas programación
2. Conceptos de orientación a objeto: clases y objetos, encapsulamiento de datos, abstracción, herencia, composición
3. Principios generales de diseño orientado a objeto
4. Patrones de diseño
5. Estilo de codificación. Control y detección de errores. Pruebas

Metodología

La metodología docente de la asignatura parte del principio que dice que "programar es la única forma de aprender a programar" y, por tanto, estará centrada principalmente en el trabajo práctico del estudiante. Las sesiones presenciales de clase se organizarán para introducir los contenidos teóricos de la asignatura, desde una perspectiva muy práctica a partir de ejemplos y de ejercicios y problemas de programación que se deberán resolver en clase directamente con el ordenador. Por otra parte, se realizará un proyecto de programación que se tendrá que ir desarrollando de forma principalmente autónoma durante todo el curso (con seguimiento y control por parte del profesor en sesiones puntuales) y que supondrá integrar de forma práctica casi todos los conceptos y herramientas de programación introducidos en las sesiones presenciales en la resolución de un problema real complejo. Además, se propondrá un conjunto de ejercicios que deben ser

resueltos de forma individual a lo largo del curso (algunos de los cuales se resolverán y discutirán en las sesiones presenciales) que deben servir para comprender, integrar y aplicar los conceptos desarrollados en las sesiones presenciales. En las actividades del curso (sesiones presenciales, problemas y prácticas) se utilizará el lenguaje de programación Python.

A nivel presencial, las sesiones de clase se organizarán en cuatro horas semanales y se harán en un aula con ordenadores para facilitar el trabajo práctico del estudiante. Se alienta que el alumno lleve su propio portátil a clase si dispone de uno. En las sesiones presenciales se irán introduciendo los conceptos que se detallan en el temario de la asignatura. En algunos casos, se podrán poner a disposición del estudiante vídeos explicativos u otro material complementario que el estudiantedeberá visualizar antes de la sesión de clase. Las sesiones de clase tendrán un enfoque bastante práctico con ejemplos y ejercicios que se plantearán a los alumnos para facilitar la comprensión y aprendizaje de los conceptos explicados. Estos ejercicios se realizarán y discutirán durante la sesión y servirán para ir introduciendo los contenidos de la asignatura y ver su aplicación práctica.

El estudiante deberá completar las clases presenciales con el trabajo personal autónomo en la realización de los ejercicios que se vayan proponiendo y que deben servir para acabar de entender los contenidos de la asignatura. Hay que tener presente que el temario de la asignatura tiene una continuidad lógica a lo largo del curso, por lo que para poder seguir correctamente una clase necesario haber asimilado lo explicado en las sesiones anteriores. Algunos de estos ejercicios se deberán entregar de forma individual como parte de la evaluación de la asignatura.

Además, los estudiantes deberán hacer en grupos de 2 un proyecto de programación que se desarrollará de forma autónoma durante todo el curso fuera de las sesiones presenciales. El proyecto de programación permitirá abordar un problema de programación de cierta complejidad que integre la mayoría de los conceptos explicados durante el curso. Durante el curso, se dedicarán algunas sesiones presenciales el control, seguimiento y evaluación del trabajo realizado por el alumno en el proyecto de programación.

Actividades

Título	Horas	ECTS	Resultados de aprendizaje
Tipo: Dirigidas			
Sesiones presenciales	50	2	2, 6, 3, 4, 7, 9
Tipo: Supervisadas			
Implementación del proyecto de programación	50	2	2, 1, 5, 7, 8
Tipo: Autónomas			
Estudio individual	14	0,56	3, 8
Resolución de problemas	32	1,28	2, 1, 5, 7, 8

Evaluación

La evaluación de la asignatura tendrá en cuenta tres tipos de actividades de evaluación: entrega de problemas, evaluación individual y proyecto de programación.

La nota final de la asignatura se obtiene combinando la evaluación de estas 3 actividades de la siguiente manera:

Nota Final = (0.3 * Evaluación Problemas) + (0.4 * Proyecto) + (0.3 * Evaluación Individual)

Entrega de problemas: en este apartado se incluye la entrega de los ejercicios que se propongan a lo largo del curso.

- Se deberá alcanzar una nota mínima de 5 en esta actividad para poder aprobar la asignatura.
- Los ejercicios que se entreguen fuera de plazo o que tengan una evaluación de suspenso se podrán recuperar y volver a entregar en cualquier momento del curso antes de la fecha del examen final de la asignatura, con una reducción sobre la nota del 20 %.

Evaluación individual: en este apartado se incluye el resultado de las pruebas individuales que se harán a lo largo del curso. Habrá dos pruebas parciales que se realizarán durante el período lectivo del curso y una prueba final durante el periodo oficial de exámenes. Esta prueba final será de recuperación y sólo tendrán que hacer los estudiantes que no hayan superado alguno de los dos parciales. Si se ha superado uno de los dos parciales, pero el otro no, en esta prueba sólo se ha de recuperar la parte de la asignatura correspondiente al parcial que no se haya superado.

- Se deberá alcanzar una nota mínima de 4,5 en cada uno de los dos parciales para poder aprobar la asignatura.
- La nota final será la media de los dos parciales:

Evaluación Individual = (0.5 * Parcial1) + (0.5 * Parcial 2)

- Deberá obtenerse una nota mínima de 5 en la nota de la evaluación individual para poder aprobar la asignatura.

Proyecto: incluye todo el trabajo del proyecto de programación. Incluye la evaluación de las dos entregas del proyecto (una entrega parcial a mitad de curso y la entrega final) y la evaluación del seguimiento del proyecto que se hará a lo largo del curso.

- Se deberá alcanzar una nota mínima de 5 en la entrega final del proyecto para poder aprobar el proyecto.
- Se deberá alcanzar una nota mínima de 5 en la nota global del proyecto para poder aprobar la asignatura.
- La nota de la entrega final del proyecto se podrá recuperar si la nota del proyecto es> = 3 y la nota de la evaluación individual es> = 5.

No evaluable: Un alumno se considerará no evaluable (NA) si no hace como mínimo el 50% de las entregas de ejercicios y no hace ninguna de las pruebas de evaluación: parcial 1, parcial 2, prueba final de recuperación, entrega final de la práctica.

Suspendidos: Si el cálculo de la nota final es igual o superior a 5 pero no se llega al mínimo exigido en alguna de las actividades de evaluación, la nota final será suspendido y se pondrá un 4.5 en la nota del expediente del alumno.

Matrículas de honor: Otorgar una calificación de matrícula de honor es decisión del profesorado responsable de la asignatura. La normativa de la UAB indica que las MH sólo se podrán conceder a estudiantes que hayan obtenido una calificación final igual o superior a 9.00. Se puede otorgar hasta un 5% de MH del total de estudiantes matriculados.

Convalidaciones: Para los alumnos repetidores se convalidará la nota del proyecto del año anterior si se cumplen estas condiciones:

- La nota final del proyecto del curso anterior es mayor o igual a 7

- La nota de la evaluación individual del curso anterior es mayor o igual a 3

Revisiones de notas: Para cada actividad de evaluación, se indicará un lugar, fecha y hora de revisión en la que el estudiante podrá revisar la actividad con el profesor. En este contexto, se podrán hacer reclamaciones sobre la nota de la actividad, que serán evaluadas por el profesorado responsable de la asignatura. Si el estudiante no se presenta en esta revisión, no se podrá revisar posteriormente esta actividad.

Nota importante: copias y plagios

Sin perjuicio de otras medidas disciplinarias que se estimen oportunas, y de acuerdo con la normativa académica vigente, las irregularidades cometidas por un estudiante que puedan conducir a una variación de la calificación se calificarán con un cero (0). Las actividades de evaluación calificadas de esta forma y por este procedimiento no serán recuperables. Si es necesario superar cualquiera de estas actividades de evaluación para aprobar la asignatura, esta asignatura quedará suspendida directamente, sin oportunidad de recuperarla en el mismo curso. Estas irregularidades incluyen, entre otros:

- la copia total o parcial de una práctica, informe, o cualquier otra actividad de evaluación;
- dejar copiar;
- presentar un trabajo de grupo no hecho íntegramente por los miembros del grupo;
- presentar como propios materiales elaborados por un tercero, aunque sean traducciones o adaptaciones, y en general trabajos con elementos no originales y exclusivos del estudiante;
- tener dispositivos de comunicación (como teléfonos móviles, smart watches, etc.) accesibles durante las pruebas de evaluación teórico-prácticas individuales (exámenes);
- hablar con compañeros durante las pruebas de evaluación teórico-prácticas individuales (exámenes);
- copiar o intentar copiar de otros alumnos durante las pruebas de evaluación teórico-prácticas (exámenes);
- usar o intentar usar escritos relacionados con la materia durante la realización de las pruebas de evaluación teórico-prácticas (exámenes), cuando éstos no hayan sido explícitamente permitidos.

En estos casos, la nota numérica del expediente será el valor menor entre 3.0 y la media ponderada de las notas (y por tanto no será posible el aprobado por compensación).

En la evaluación de las entregas de problemas y prácticas se utilizarán herramientas de detección de copia del código del programa.

Actividades de evaluación

Título	Peso	Horas	ECTS	Resultados de aprendizaje
Entrega de problemas	30%	0	0	2, 1, 6, 5, 7
Evaluación individual	30%	4	0,16	2, 9
Proyecto de programación	40%	0	0	1, 3, 4, 5, 7, 8

Bibliografía

- Python: master the art of design patterns. D. Phillips, C. Giridhar, S. Kasampalis. Packt Publishing, 2016.
- Mastering object-oriented Python. Steven F. Lott. Packt publishing, 2014.
- Head first object oriented analysis and design. B.D. McLaughlin, G. Pollice, D. West. O'Reilly, 2007.
- Design patterns explained simply. Alexander Shvets. https://sourcemaking.com/design-patterns-ebook.
- Head first design patterns. E. Freeman, E. Freeman, K. Sierra, B. Bates. O'Reilly, 2004.
- Clean code: a handbook of agile software craftmanship. R.C. Martin. Prentice Hall, 2008. O la versió en castellà, Código limpio, d'Anaya
  Multimedia 2012.