GUÍA DOCENTE DE ANALISIS Y DISEÑO DE SOFTWARE

La presente guía docente corresponde a la asignatura Análisis y Diseño de Software (ADS), aprobada para el curso lectivo 2016-2017 en Junta de Centro y publicada en su versión definitiva en la página web de la Escuela Politécnica Superior. El presente documento es una guía, y por tanto es posible que su desarrollo durante el curso sufra alguna variación.

ASIGNATURA

ANÁLISIS Y DISEÑO DE SOFTWARE (ADS)

1.1. Código

17832 del Grado en Ingeniería Informática

1.2. Materia

Análisis y Diseño de Software

1.3. Tipo

Formación obligatoria

1.4. Nivel

Grado

1.5. Curso

2º

1.6. Semestre

2º

1.7. Número de créditos

6 créditos ECTS

1.8. Requisitos previos

Esta asignatura asume conocimientos de programación estructurada por parte del estudiante, que deben haber sido adquiridos cursando las materias del módulo de *Programación y Estructuras de Datos* del plan de estudios. Es muy aconsejable haber superado con éxito las asignaturas *Programación I*, *Programación II* y *Proyecto de Programación* para un aprovechamiento de la asignatura ADS. La materia Análisis y Diseño de Software se desglosa en dos asignaturas: Análisis y Diseño de Software y Proyecto de Análisis y Diseño de Software, ambas impartidas en el 2º cuatrimestre

del segundo curso. Ya que ambas proporcionan formación complementaria y conceptos relacionados, **se recomienda cursarlas a la vez**.

Se recomienda para garantizar la asimilación de los contenidos y la adquisición de habilidades la lectura crítica de los textos de la bibliografía, el uso del material electrónico de esta asignatura disponible en la plataforma Moodle (http://moodle.uam.es) y la búsqueda activa de material complementario en la red. Es recomendable disponer de un dominio de inglés que permita al alumno leer la bibliografía de consulta. Asimismo, se requiere iniciativa personal y constancia para el diseño e implementación de aplicaciones, así como predisposición y empatía para el trabajo colaborativo en grupo.

1.9. Requisitos mínimos de asistencia a las sesiones presenciales

Se plantean dos itinerarios, uno con asistencia obligatoria a clase y otro sin ella, los estudiantes deberán optar por uno u otro **dentro de las primeras dos semanas** y cumplir con los distintos requisitos de evaluación que conlleva cada uno de los modelos, publicados en la presente guía docente (ver apartado 4). Si un estudiante desea pasar de itinerario continuo a final, después de las dos primeras semanas, podrá hacerlo.

ITINERARIO CON ASISTENCIA OBLIGATORIA A CLASE
La asistencia es obligatoria al menos en un 85%.

ITINERARIO SIN ASISTENCIA OBLIGATORIA A CLASE
La asistencia es muy recomendable aunque no obligatoria.

1.10. Datos del equipo docente

Nota: se debe añadir @uam.es a todas las direcciones de correo electrónico.

Profesores de teoría:

Dr. Miguel Ángel Mora Rincón (coordinador)
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Despacho - Módulo: B-345 Edificio B – 3ª Planta
Teléfono: +34 91 497 2214
Correo electrónico: miguel.mora
Página web: http://www.ii.uam.es/~mora/
Horario de atención al alumnado: Petición de cita previa por correo electrónico.

1.11. Objetivos del curso

El estudiante aprenderá a analizar, diseñar, implementar y probar software usando el paradigma de Orientación a Objetos. El estudiante será capaz de emplear métodos, técnicas y herramientas ingenieriles para la construcción de aplicaciones robustas y mantenibles. Se hará especial énfasis en la fase de implementación, usando el lenguaje de programación Java.

Las **competencias** que se pretenden adquirir con esta asignatura son:

Comunes:
C1. Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
C2. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
C3. Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software.
C4. Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.
C5. Conocimiento, administración y mantenimiento sistemas, servicios y aplicaciones informáticas.
C8. Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
C16. Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.
C17. Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.

De tecnología específica
IS1. Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la Ingeniería del Software.

Los objetivos que se pretenden alcanzar con esta asignatura son:

OBJETIVOS GENERALES	
G1	Analizar, Diseñar, Implementar y Probar programas usando tecnologías de Orientación a Objetos.
G2	Conocer y aplicar buenas prácticas de Ingeniería del Software en el desarrollo de aplicaciones.

OBJETIVOS ESPECIFICOS POR TEMA

TEMA 1.- Ciclo de Vida del Software	
1.1.	Comprender la utilidad de las distintas fases de un proyecto: desarrollar software no es sólo programar.
1.2.	Comparar las distintas metodologías y modelos de desarrollo de software, y ser capaz de seleccionar los más adecuados a cada tipo de proyecto.
TEMA 2.- Orientación a Objetos	
2.1.	Conocer los conceptos fundamentales del diseño orientado a objetos.
2.2.	Comparar el enfoque procedimental con el orientado a objetos, y entender las ventajas y limitaciones de cada uno.
2.3.	Aplicar notaciones de diseño (UML) para reflejar la estructura y comportamiento de un sistema software orientado a objetos.
TEMA 3.- Diseño e Implementación en Java	
3.1.	Comparar el lenguaje Java con C, entender su modo de funcionamiento, sus ventajas y limitaciones.
3.2.	Diseñar una aplicación a nivel básico: clases y objetos, y su implementación en Java.
3.3.	Diseñar el control de acceso y la organización modular de una aplicación.
3.4.	Diseñar la jerarquía de herencia, así como dominar y aplicar el concepto de interfaz en el diseño orientado a objetos.
3.5.	Aplicación de las librerías de Java: colecciones.
3.6.	Diseño e implementación de la gestión de errores, y el manejo de excepciones.
3.7.	Colecciones y Genericidad.
3.8.	Expresiones Lambda e interfaces funcionales.
TEMA 4.- Patrones de Diseño	
4.1.	Comprender el papel fundamental de los patrones de diseño y la reutilización en la construcción de software.
4.2.	Utilizar patrones de diseño en el desarrollo de aplicaciones.

1.12. Contenidos del programa

Programa Sintético

UNIDAD 1. Ciclo de Vida del Software
UNIDAD 2. Orientación a Objetos
UNIDAD 3. Diseño e Implementación en Java
UNIDAD 4. Patrones de Diseño

Programa Detallado

1. **Cliclo de Vida del Software**
   1.1. Fases en el desarrollo del Software
   1.2. Modelos de Ciclo de Vida
   1.3. Metodologías
2. **Orientación a Objetos**
   2.1. Conceptos de Diseño Orientado a Objetos
   2.2. Objetos y Clases
   2.3. Encapsulamiento

2.4. Herencia y Polimorfismo
2.5. Notaciones de Diseño.

3. Diseño e Implementación en Java

3.1. Introducción a Java
3.2. Clases y Objetos en Java
3.3. Jerarquías de clases.
3.3.1. Subclases y Herencia
3.3.2. Polimorfismo.
3.3.3. Métodos abstractos.
3.3.4. Clases anónimas.
3.4. Control de acceso, modularidad.
3.5. Interfaces.
3.6. Entrada/Salida. Tratamiento de errores. Excepciones.
3.7. Colecciones y Genericidad.
3.8. Expresiones Lambda e interfaces funcionales.

4. Patrones de Diseño

4.1. Introducción.
4.2. Patrones de Creación.
4.3. Patrones Estructurales.
4.4. Patrones de Comportamiento.

1.13. Referencias de consulta

Bibliografía:

Nota: Esta asignatura no sigue ningún libro en concreto. La lectura recomendada se lista por orden de afinidad al contenido del programa.

Unidad 1:

1. Software engineering a practitioner's approach, 7ªed. Roger Pressman, McGraw Hill Higher Education, 2010. INF/C6110B/PRE. También disponible en castellano. Capítulos 1 y 2.
2. Software engineering, 9ª ed. Ian Sommerville, Addison Wesley, 2011. INF/C6110B/SOM. También disponible en castellano. Capítulos 1 y 2.

Unidad 2:

3. Ingeniería de software clásica y orientada a objetos, Sexta Edición. Stephen Schach. McGraw-Hill. INF/C6110B/SCH.
4. Construcción de software orientado a objetos. Betrand Meyer. Prentice Hall. INF/C6110J/MEY.
5. El lenguaje unificado de modelado manual de referencia. Rumbaugh, James. Pearson Addison Wesley. 2007. INF/C6140-U/RUM.

Unidad 3:

6. Core Java Vol. I Fundamentals, Horstmann, Cay S. Prentice Hall, 2015. INF/C6140-J/HOR Vol. 1. También disponible en castellano.

7. Core Java Vol. II Advanced features, Horstmann, Cay S. Prentice Hall, 2015. INF/C6140-J/HOR Vol. 2. También disponible en castellano.
8. Functional programming in Java harnessing the power of Java 8 Lambda expressions, Subramaniam, Venkat. Disponible electrónicamente en la biblioteca de la EPS.

Unidad 4:

9. Patrones de diseño elementos de software orientado a objetos reutilizable. Gamma, E., Helm, R., Johnson, R., Vlissides, J. INF/C6110J/PAT. Addison-Wesley, 2003.
10. Patrones de diseño aplicados a Java. Stelting, Stephen. INF/C6140-J/STE. Pearson, Prentice Hall. 2003.
11. UML y patrones una introducción al análisis y diseño orientado a objetos y al proceso unificado (2ª edición). Craig Larman. Prentice Hall, 2002. INF/C6140-U/LAR. También disponible en formato electrónico.

Nota: no se recomienda a los estudiantes comprar ningún libro hasta no haber comparado su contenido con el programa y revisado previamente en la biblioteca.

Material electrónico de trabajo: los documentos electrónicos de trabajo (recomendaciones sobre la elaboración de documentación, realización de diagramas, recomendaciones sobre legibilidad en el código, ejercicios del curso, etc.) se publican en la sección de ADS en plataforma Moodle (http://moodle.uam.es)

2. Métodos docentes

La metodología utilizada en el desarrollo de la actividad docente incluye los siguientes tipos de actividades:

***Clases de teoría:**

Actividad del profesor

Clases expositivas simultaneadas con el modelado de sistemas, la realización de programas y la propuesta de ejercicios. Se utilizará la pizarra, combinada con la realización de modelos y diagramas, así como programas Java en formato electrónico cuya ejecución se visualizará en la pantalla de la clase.

Actividad del estudiante:

Actividad presencial: Toma de apuntes, participación activa en clase respondiendo a las cuestiones planteadas. Resolución de los ejercicios propuestos, realización de diagramas, diseño de aplicaciones sencillas y escritura de pequeños programas durante el desarrollo de las clases.

Actividad no presencial: lectura del material bibliográfico y de apoyo, estudio de la materia, realización de diagramas y desarrollo de programas de manera autónoma para entender los conceptos y resolución de los cuestionarios planteados en la plataforma Moodle.

***Clases de problemas/ejercicios en aula:**

Actividad del profesor

Primera parte expositiva, una segunda parte de supervisión y asesoramiento en la resolución de los problemas por parte del alumno y una parte final de análisis del resultado y generalización a otros tipos de problemas. Se utilizará la pizarra y el proyector del aula para visualizar los diseños o programas propuestos.

Actividad del estudiante:

Actividad presencial: Participación activa en la resolución de los ejercicios, realización de diagramas diseño y escritura de los programas y en el análisis de la ejecución.

Actividad no presencial: Realización de ejercicios, modelos y programas, planteados en clase o a través de la plataforma Moodle. Estudio, generalización y planteamiento de modificaciones que permitan la optimización de los programas.

***Tutorías en aula:**

Actividad del profesor:

Tutorización a toda la clase o en grupos de alumnos reducidos (8-10) con el objetivo de resolver dudas comunes plantadas por los alumnos a nivel individual o en grupo, surgidas a partir de cuestiones/ejercicios/modelos/programas señalados en clase para tal fin y orientarlos en la realización de los mismos.

Actividad del estudiante:

Actividad presencial: Planteamiento de dudas individuales o en grupo y enfoque de posibles soluciones a las tareas planteadas.

Actividad no presencial: Estudio de las tareas marcadas y debate de las soluciones planteadas en el seno del grupo.

***Prácticas:**

Actividad del profesor:

Asignar una práctica/proyecto a cada grupo de trabajo y explicar la práctica asignada a cada grupo de trabajo al comienzo de la sesión de prácticas. Supervisar el trabajo de los grupos de trabajo en el laboratorio. Suministrar el guión de prácticas a completar en el laboratorio.

Se utilizan el método expositivo tanto en tutorías como en el laboratorio con cada grupo de trabajo. Los medios utilizados son los entornos de desarrollo y los ordenadores del propio laboratorio para el modelado, la ejecución y análisis de los programas realizados.

Actividad del estudiante:

Actividad presencial: Planteamiento inicial, previo al desarrollo de la práctica, sobre información contenida en el enunciado. Debate en el seno del grupo sobre el planteamiento de la solución óptima. Al finalizar la práctica se entrega un breve informe con la solución obtenida. En algunas prácticas, se requerirá la ejecución con el profesor presente, quien hará las preguntas oportunas a cada miembro del grupo para calificar de forma individual la práctica.

Actividad no presencial: Profundizar en el enunciado de la práctica, diseño y programación adicionales. Redacción del informe de la práctica.

3. Tiempo de trabajo del estudiante

		Nº de horas	Porcentaje
Presencial	Clases teóricas	42 h (28%)	75 h (50%)
	Clases prácticas	26 h (17,3%)	
	Tutorías programadas a lo largo del semestre	5 h (3,3%)	
	Realización de pruebas escritas parciales	2 h (1,4%)	
No presencial	Estudio semanal regulado	18 h (12%)	75 h (50%)
	Realización de actividades prácticas	19 h (12,7%)	
	Preparación de los exámenes	38 h (25,3%)	
Carga total de horas de trabajo: 25 horas x 6 ECTS		**150 h**	

4. Métodos de evaluación y porcentaje en la calificación final

- Ambas partes, teoría y prácticas se puntúan sobre 10 puntos.
- La nota final de la asignatura se obtiene de las notas de teoría y prácticas por medio de la ecuación:

Calificación: 0.4*Prácticas + 0.6*Teoría

- Para aprobar la asignatura es obligatorio obtener una nota mayor o igual a 5 puntos, tanto en la parte de teoría como en las prácticas. En caso contrario, la nota final en actas será

Calificación: (0,4*Mín(5,Prácticas) + 0,6*Mín(5,Teoría))

La nota correspondiente a la parte de Teoría es la que resulta del **máximo** de:

- ✓ La calificación de la prueba final (60%) y la calificación de las pruebas/actividades/ejercicios intermedios (40%). Las pruebas intermedias tendrán una ponderación de 30% la primera, y 70% la segunda.
- ✓ La calificación de la prueba final.

Para ser evaluado por la modalidad continua, es necesario tener una media de 4.0 o más en las pruebas parciales, y haberse presentado a todas ellas.

La nota correspondiente a la parte de Teoría para el itinerario sin asistencia obligatoria corresponde únicamente a la prueba final (distinta a la de la trayectoria con asistencia obligatoria ya que abarca ejercicios relacionados con las actividades intermedias).

Las pruebas escritas, podrán incluir tanto cuestiones teóricas y ejercicios como la realización de diagramas, y el diseño y escritura de programas.

➢ La nota correspondiente a la parte de prácticas es la que resulta de realizar las prácticas programadas en el curso.

- ✓ Para aprobar la parte práctica el estudiante deberá asistir, al menos, al 85% de las prácticas. En caso contrario deberá realizar un examen de prácticas consistente en una práctica de mayor complejidad a las realizadas en el laboratorio.
- ✓ Para aprobar la parte práctica, es imprescindible entregar cada una de las prácticas y obtener al menos una nota de 3,5 puntos en cada una. En caso de que esto no se cumpla, por faltar alguna entrega, no llegar a la nota mínima en cada una, o no hacer el examen de prácticas en el caso de evaluación no continua, la nota de prácticas será el Min(4, media ponderada de las prácticas).
- ✓ La calificación de la parte práctica tendrá en cuenta la calidad de los diseños realizados y el nivel de los resultados obtenidos. También se valorará la validez de los resultados obtenidos en cada uno de los apartados que se hayan establecido para su realización en los guiones de las prácticas.

➢ Las notas tanto de teoría como de prácticas se conservan sólo para la convocatoria extraordinaria del mismo curso académico.

➢ El número mínimo de pruebas a las que el estudiante se ha de presentar para recibir una calificación numérica es 1. Por debajo de este número el estudiante recibirá la calificación "No evaluado". Aunque el estudiante no se presente a la prueba final, siempre que se haya presentado a este mínimo de pruebas, recibirá una calificación numérica.

ATENCIÓN: Cualquier copia descubierta que se haya realizado a lo largo del curso, tanto en cualquiera de las actividades de teoría desarrolladas, como en cualquiera de los apartados de las prácticas, serán penalizadas con rigurosidad. La normativa de la UAM establece que cuando un profesor observe conductas o actos en un estudiante incompatibles con la probidad y la ética, con independencia de su posible repercusión en la calificación de la prueba, podrá solicitar del Rector la incoación del correspondiente expediente informativo al estudiante en cuestión.

5. Cronograma

Semana	Contenido	Horas presenciales	Horas no presenciales (Trabajo autónomo del estudiante)
1	- Presentación de la asignatura, descripción del programa, normativa y los métodos de evaluación. - **Unidad 1. Ciclo de Vida del Software. Temas 1.1, 1.2 y 1.3**	3	5 Lectura de las normativas de teoría y prácticas. Estudio del material de la Unidad 1.
2	- **Unidad 2 Orientación a objetos. Temas 2.1, 2.2 y 2.3**	5	3 Estudio del material de la Unidad 2.

Semana	Contenido	Horas presenciales	Horas no presenciales (Trabajo autónomo del estudiante)
	- Práctica 0: Introducción a Java.		Realización de los ejercicios propuestos. Realización de Práctica 0.
3	**- Unidad 2 Orientación a Objetos. Temas 2.4 y 2.5** - Práctica 1: Introducción al diseño orientado a objetos	4	2 Estudio del material de la Unidad 2. Realización de los ejercicios propuestos. Entrega de la Práctica 0. Realización de Práctica 1.
3	**Prueba #1: Análisis y conceptos de orientación a objetos.** **Tutoría sobre las Unidades 1 y 2.**	1 1	
4	**- Unidad 3. Diseño e Implementación en Java. Tema 3.1.** - Práctica 2: Java Básico.	5	3 Estudio del material de la Unidad 3. Realización de los ejercicios. Entrega de la práctica 1. Realización de Práctica 2.
5	**- Unidad 3. Diseño e Implementación en Java. Tema 3.2.** -Práctica 2: Java Básico.	5	3 Estudio del material de la Unidad 3. Realización de los ejercicios. Realización de Práctica 2.
6	**-Unidad 3. Diseño e Implementación en Java. Tema 3.3.** -Práctica 3: Clases y Objetos Java.	5	3 Estudio del material de la Unidad 3. Realización de los ejercicios. Entrega Práctica 2. Realización de Práctica 3.
7	**-Unidad 3. Diseño e Implementación en Java. Tema 3.3.** -Práctica 3: Clases y Objetos Java.	5	3 Estudio del material de la Unidad 3. Realización de los ejercicios. Realización de Práctica 3.
8	**-Unidad 3. Diseño e Implementación en Java. Tema 3.4 y 3.5** -Práctica 3: Clases y Objetos Java.	4	2 Estudio del material de la Unidad 3. Realización de los ejercicios. Realización Práctica 3.
9	**-Unidad 3. Diseño e Implementación en Java. Tema 3.5 y 3.6.** -Práctica 4: Java avanzado.	5	3 Estudio del material de la Unidad 3. Realización de los ejercicios. Entrega Práctica 3. Realización Práctica 4.
9	**Prueba #2: Java** **Tutoría.**	1 1	
10	**-Unidad 3. Diseño e Implementación en Java. Tema 3.6.**	5	3 Estudio del material de la

Semana	Contenido	Horas presenciales	Horas no presenciales (Trabajo autónomo del estudiante)
	-Practica 4: Java avanzado.		Unidad 3. Realización de los ejercicios. Realización de Práctica 4.
11	**-Unidad 3. Diseño e Implementación en Java. Tema 3.6 y 3.7** -Practica 5: Colecciones y expresiones lambda.	4	2 Estudio del material de la Unidad 3. Realización de los ejercicios. Entrega Práctica 4. Realización de Práctica 5.
11	**Tutoría**	1	
12	**-Unidad 3. Diseño e Implementación en Java. Tema 3.8.** -Practica 5: Colecciones y expresiones lambda.	5	3 Estudio del material de la Unidad 3. Realización de los ejercicios. Realización de Práctica 5.
13	**-Unidad 4. Patrones de Diseño** -Practica 5: Colecciones y expresiones lambda.	5	3 Estudio del material de la Unidad 4. Realización de los ejercicios. Realización y entrega de Práctica 5.
14	- Tutorías sobre los temas 3 y 4.	1	
14	**-Unidad 4. Patrones de Diseño** **- Repaso y preparación del examen**	5	2 Estudio del material de la Unidad 4. Realización de los ejercicios.
	Examen Final	3	16h