UNIVERSIDAD AUTÓNOMA DE MADRID

17832 - SOFTWARE ANALYSIS AND DESIGN

Information of the subject

Code - Course title: 17832 - SOFTWARE ANALYSIS AND DESIGN

Degree: 473 - Graduado/a en Ingeniería Informática
474 - Graduado/a en Ingeniería Informática y Matemáticas
722 - Graduado/a en Ingeniería Informática
734 - Graduado/a en Ingeniería Informática y Matemáticas (2019)

Faculty: 350 - Escuela Politécnica Superior

Academic year: 2020/21

1. Course details

1.1. Content area

Ingeniería del software

1.2. Course nature

Compulsory

1.3. Course level

Grado (EQF/MECU 6)

1.4. Year of study

2

1.5. Semester

Second semester

1.6. ECTS Credit allotment

6.0

1.7. Language of instruction

English

1.9. Recommendations

This course assumes knowledge of structured programming by the student, which may have been acquired by studying the subjects of the Programming and Data Structures module of the study plan. It is highly advisable to have successfully passed the subjects Programming I, Programming II and Programming Project in order to take advantage of the Software Analysis and Design (ADS) subject. The ADS subject is further divided into two subjects: Software Analysis and Design and Software

Analysis and Design Project, both taught in the 2nd semester of the second course. Since both provide complementary training and related concepts, it is recommended to study them at the same time.

To ensure the assimilation of the contents and the acquisition of skills, it is recommended to read critically the texts of the bibliography, use the electronic material for this subject available on the Moodle platform and actively search for complementary material on the internet. It is recommended to have a good command of English to follow the course. Likewise, personal initiative and perseverance are required for the design and implementation of applications, as well as predisposition and empathy for collaborative group work.

1.10. Minimum attendance requirement

Two itineraries are proposed, one with compulsory class attendance and the other without it, students must choose one or the other within the first two weeks and meet the different assessment requirements that each of the models entails, published in this Teaching guide. If a student wishes to move from continuous to final itinerary, after the first two weeks, they can do so.

ITINERARY WITH MANDATORY CLASS ASSISTANCE

Attendance is compulsory at least 85%.

ITINERARY WITHOUT MANDATORY CLASS ATTENDANCE

Attendance is highly recommended but not mandatory.

1.11. Subject coordinator/s

Juan de Lara Jaramillo

https://autoservicio.uam.es/paginas-blancas/

1.12. Competences and learning outcomes

1.12.1. Competences

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.
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.

1.12.2. Learning outcomes

Analyze, Design, Implement and Test programs using Object Orientated technologies.

Know and apply good practices of Software Engineering to application development.

1.12.3. Course objectives

The specific objectives to be achieved with this course are:

LESSON 1.- Software life-cycle	
1.	Understand the usefulness of the different phases of a project: developing software is not just programming.
2.	Compare the different methodologies and models of software development, and be able to select the most appropriate for each type of project.

LESSON 2.- Object Orientation	
1.	Know the fundamental concepts of object-oriented design.
2.	Compare the procedural approach with the object-oriented one, and understand the advantages and limitations of each one.
3.	Apply design notations (UML) to reflect the structure and behavior of an object-oriented software system.
LESSON 3.- Design and Implementation in Java	
1.	Compare the Java language with C, understand how it works, its advantages and limitations.
2.	Design an application at a basic level: classes and objects, and its implementation in Java.
3.	Design an inheritance hierarchy as well as master and apply the interface concept in object-oriented design.
4.	Design the access control and modular organization of an application.
5.	Learn how to use Interfaces
6.	Design and implementation of error handling and exception handling.
7.	Learn to use the different Library Collections and Generic parameters.
8.	Understand how Internal Classes and Reflection work.
9.	Using Lambda expressions and functional interfaces
LESSON 4.- Design patterns	

1.	Understand the critical role of design patterns and reuse in software construction.
2.	Use design patterns in application development.

1.13. Course contents

Synthetic Program

UNIT 1. Software Life Cycle

UNIT 2. Object Orientation

UNIT 3. Design and Implementation in Java

UNIT 4. Design Patterns

Detailed Program

1. Software Life Cycle
    1. Phases in software development
    2. Life Cycle Models
    3. Methodologies
2. Object Orientation
    1. Object Oriented Design Concepts
    2. Objects and Classes
    3. Encapsulation
    4. Inheritance and Polymorphism
    5. Design notations.
3. Design and Implementation in Java
    1. Introduction to Java
    2. Classes and Objects in Java
    3. Class hierarchies.
    4. Subclasses and Inheritance
    5. Polymorphism.
    6. Abstract methods.
    7. Anonymous classes.
    8. Access control, modularity.
    9. Interfaces.
    10. I/O. Error handling. Exceptions.
    11. Collections and Genericity.
    12. Internal Classes and Reflection.
    13. Lambda expressions and functional interfaces.
4. Design patterns
    1. Introduction.
    2. Creation Patterns.
    3. Structural Patterns.
    4. Behavioural patterns.

1.14. Course bibliography

Note: This course does not follow any specific book. The recommended reading is listed in order of affinity to the program content.

Unit 1:

Software engineering a practitioner's approach, 7th ed. Roger Pressman, McGraw Hill Higher Education, 2010. INF / C6110B / PRE. Also available in Spanish. Chapters 1 and 2.

Software engineering, 9th ed. Ian Sommerville, Addison Wesley, 2011. INF / C6110B / SOM. Also available in Spanish.

Chapters 1 and 2.

Unit 2:

Classical and Object Oriented Software Engineering, Sixth Edition. Stephen Schach. McGraw-Hill. INF / C6110B / SCH.

Construction of object-oriented software. Betrand Meyer. Prentice Hall. INF / C6110J / MEY.

The Unified Modeling Language Reference Manual. Rumbaugh, James. Pearson Addison Wesley. 2007. INF / C6140-U / RUM.

Unit 3:

Core Java Vol. I Fundamentals, Horstmann, Cay S. Prentice Hall, 2015. INF / C6140-J / HOR Vol. 1. Also available in Spanish.

Core Java Vol. II Advanced features, Horstmann, Cay S. Prentice Hall, 2015. INF / C6140-J / HOR Vol. 2. Also available in Spanish.

Functional programming in Java harnessing the power of Java 8 Lambda expressions, Subramaniam, Venkat. Available electronically in the EPS library.

Unit 4:

Design patterns elements of reusable object-oriented software. Gamma, E., Helm, R., Johnson, R., Vlissides, J. INF / C6110J / PAT. Addison-Wesley, 2003.

Design patterns applied to Java. Stelting, Stephen. INF / C6140-J / STE. Pearson, Prentice Hall. 2003.

UML and patterns an introduction to object-oriented analysis and design and the unified process (2nd edition). Craig Larman. Prentice Hall, 2002. INF / C6140-U / LAR. Also available in electronic format.

Note: Students are not recommended to purchase any book until they have compared its content to the program and previously reviewed it in the library.

Electronic working material: the electronic working documents (recommendations on preparing documentation, making design diagrams, recommendations on code readability, course exercises, etc.) are published in the ADS section on the Moodle platform (http: / /moodle.uam.es)

2. Teaching-and-learning methodologies and student workload

2.1. Contact hours

	#horas
Contact hours (minimum 33%)	75
Independent study time	75

2.2. List of training activities

Activity	# hours
Lectures in the classroom	42
Seminars	
Practical sessions	
Clinical sessions	
Computer lab	26
Laboratory	
Work placement	
Supervised study	
Tutorials	5
Assessment activities	2
Other	

3. Evaluation procedures and weight of components in the final grade

3.1. Regular assessment

- Both parts, theory and practice are scored on 10 points.
- The final grade for the course is obtained from the theory and practical notes by means of the equation:

Grade: 0.3 * Lab + 0.7 * Theory

- To pass the course it is mandatory to obtain a grade greater than or equal to 5 points, both in the theory part and in the lab. Otherwise, the final grade will be

Grade: (0.3 * Min (5, Lab) + 0.7 * Min (5, Theory))

The grades corresponding to the Theory part is the one that results from the maximum of:

- The final test grade (60%) and the intermediate tests / activities / exercises (40%).
- The final test grade.

The grade of the intermediate tests will be calculated using the weighted average of each test, with a weight between 30% and 70%. Said weighting will be published together with the schedule of the tests.

To be evaluated by the continuous modality, it is necessary to have an average of 4.0 or more in the intermediate tests, and to have taken all of them.

The grade corresponding to the Theory part for the itinerary without compulsory attendance corresponds only to the final test (different from that of the trajectory with compulsory attendance since it covers exercises related to intermediate activities).

Written tests may include both theoretical questions and exercises such as diagramming, and the design and writing of programs.

The note corresponding to the lab part is the one that results from carrying out the practices programmed in the course.

To pass the lab part, the student must attend at least 85% of the lab hours. Otherwise, the student must take a practical exam consisting of a more complex assignments than those performed in the lab.

To pass the lab part, it is essential to deliver each of the assignments and obtain at least a grade of 3.5 points in each one of them. In the event that this is not met, because some assignement is not submitted, because the minimum grade is not met, or because of failing the lab exam in the case of non-continuous evaluation, the lab mark will be the result of Min (4, average weighted practices).

The qualification of the practical part will take into account the quality of the designs made and the level of the results obtained. The validity of the results obtained in each of the sections that have been established for their implementation in the practice scripts will also be assessed.

The grades of both theory and practice are kept only for the extraordinary call of the same academic year.

The minimum number of tests to which the student must take to receive a numerical grade is 1. Below this number the student will receive the grade "Not evaluated". Even if the student does not take the final exam, as long as they have taken this minimum number of tests, they will receive a numerical grade.

WARNING: Any discovered copy that has been made throughout the course, in any of the theory tests, or in the labs, will be strictly punished. The UAM regulations establish that when a teacher observes behaviours in a student that are incompatible with probity and ethics, regardless of their possible repercussion on the test score, they may request the Rector to initiate the corresponding information file for the student in question.

3.1.1. List of evaluation activities

Evaluatory activity	%
Final exam	60%
Continuous assessment	40%

3.2. Resit

- Both parts, theory and practice are scored over 10 points.
- The final grade for the course is obtained from the theory and practical notes by means of the equation:

Grade: 0.3 * Lab + 0.7 * Theory

To pass the course it is mandatory to obtain a grade greater than or equal to 5 points, both in the theory part and in the lab. Otherwise, the final grade will be

Grade: (0.3 * Min (5, Lab) + 0.7 * Min (5, Theory))

Written tests may include both theoretical questions and exercises such as creating design diagrams, and the design and coding of programs.

The minimum number of tests to which the student must take to receive a numerical grade is 1. Below this number the student will receive the grade "Not evaluated". Even if the student does not take the final exam, as long as they have taken this minimum number of tests, they will receive a numerical grade.

WARNING: Any discovered copy that has been made throughout the course, in any of the theory tests, or in the labs, will be strictly punished. The UAM regulations establish that when a teacher observes behaviours in a student that are incompatible with probity and ethics, regardless of their possible repercussion on the test score, they may request the Rector to initiate the corresponding information file for the student in question.

3.2.1. List of evaluation activities

Exam 100%

4. Proposed workplan

Week	Contents	Hours (in the classroom)	Non-contact hours (Autonomous student work)
1	- Presentation of the course, description of the syllabus, regulations and evaluation methods. - Unit 1. Software Life Cycle. Topics 1.1, 1.2 and 1.3	3	5 Reading of the theory and practice regulations. Study of the material from Unit 1.
2	- Unit 2 Object Orientation. Topics 2.1, 2.2 and 2.3 - Lab 1: Introduction to Java.	5	3 Study of the material in Unit 2. Work on the proposed exercises. Work of Lab 1.
3	- Unit 2 Object Orientation. Topics 2.4 and 2.5 - Lab 2: Introduction to object-oriented design.	4	2 Study of the material in Unit 2. Work on the proposed exercises. Delivery of Lab 1. Work on Lab 2.
3	Test # 1: Analysis and concepts of object orientation. Tutoring on Units 1 and 2.	1 1	
4	- Unit 3. Design and Implementation in Java. Topic 3.1. - Lab 2: Introduction to object-oriented design.	5	3 Study of the material in Unit 3. Work on exercises. Work on Lab 2.
5	- Unit 3. Design and Implementation in Java. Topic 3.2. - Lab 3: Classes and Java Objects.	5	3 Study of the material in Unit 3. Work on exercises. Delivery of Lab 2. Work on Lab 3.
6	-Unit 3. Design and Implementation in Java. Topic 3.3. - Lab 3: Classes and Java Objects.	5	3 Study of the material in Unit 3. Work on exercises. Work on lab 3.
7	-Unit 3. Design and Implementation in Java. Topic 3.4. -Lab 3: Classes and Java Objects.	5	3 Study of the material in Unit 3.

			Work on exercises. Work on Lab 3.
8	-Unit 3. Design and Implementation in Java. Topic 3.5 -Lab 4: Inheritance, interfaces and exceptions.	4	2 Study of the material in Unit 3. Work on exercises. Delivery of Lab 3. Work on Lab 4.
9	-Unit 3. Design and Implementation in Java. Topic 3.6. -Lab 4: Inheritance, interfaces and exceptions	5	3 Study of the material in Unit 3. Work on exercises. Work on Lab 4.
9	Test #2: Java Tutorials.	1 1	
10	-Unit 3. Design and Implementation in Java. Topic 3.7. -Lab 4: Inheritance, interfaces and exceptions.	5	3 Study of the material in Unit 3. Work on exercises. Work on Lab 4.
11	-Unit 3. Design and Implementation in Java. Topic 3.8. -Lab 5: Genericity, lambda expressions and design patterns.	4	2 Study of the material in Unit 3. Work on exercises. Delivery of Lab 4. Work on Lab 5.
11	Tutorial	1	
12	-Unit 3. Design and Implementation in Java. Topic 3.9. -Practice 5: Genericity, lambda expressions and design patterns.	5	3 Study of the material in Unit 3. Work on exercises. Work on Lab 5.
13	-Unit 4. Design Patterns -Practice 5: Genericity, lambda expressions and design patterns.	5	3 Study of the material in Unit 4. Work on exercises. Work and delivery of Lab 5.
14	- Tutorial on lessons 3 and 4.	1	
14	-Unit 4. Design Patterns - Review and preparation for the exam	5	2 Study of the material in Unit 4. Work on exercises.
	Final exam	3	16h