SYLLABUS

1. Data about the program of study

1.1 Institution	The Technical University of Cluj-Napoca
1.2 Faculty	Faculty of Automation and Computer Science
1.3 Department	Computer Science
1.4 Field of study	Computer Science and Information Technology
1.5 Cycle of study	Bachelor of Science
1.6 Program of study/Qualification	Computer science/ Engineer
1.7 Form of education	Full time
1.8 Subject code	41.

2. Data about the subject

2.1 Subject name	*Software design*				
2.2 Course responsible/lecturer	Prof.dr.eng. Mihaela Dinsoreanu – mihaela.dinsoreanu@cs.utcluj.ro				
2.3 Teachers in charge of seminars/ laboratory/ project	Prof.dr.eng. Mihaela Dinsoreanu – mihaela.dinsoreanu@cs.utcluj.ro				
2.4 Year of study	III	2.5 Semester	2	2.6 Type of assessment (E - exam, C - colloquium, V - verification)	E
2.7 Subject category	*DF – fundamentală, DD – în domeniu, DS – de specialitate, DC – complementară*				DS
	DI – Impusă, DOp – opțională, DFac – facultativă				DI

3. Estimated total time

3.1 Number of hours per week	5	of which:	Course	2	Seminars		Laboratory	2	Project	1
3.2 Number of hours per semester	70	of which:	Course	28	Seminars		Laboratory	28	Project	14

3.3 Individual study:	
(a) Manual, lecture material and notes, bibliography	10
(b) Supplementary study in the library, online and in the field	5
(c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays	6
(d) Tutoring	4
(e) Exams and tests	5
(f) Other activities:	

3.4 Total hours of individual study (suma (3.3(a)…3.3(f)))	30	
3.5 Total hours per semester (3.2+3.4)	100	
3.6 Number of credit points	4	

4. Pre-requisites (where appropriate)

4.1 Curriculum	Programming Techniques, Software Engineering
4.2 Competence	

5. Requirements (where appropriate)

5.1. For the course	Video projector (compulsory), internet connected computer, Moodle, Webex
5.2. For the applications	16 internet connected computers, Specific software, GitHub, Skype

6. Specific competence

6.1 Professional competences	**C3** - Problem solving using specific Computer Science and Computer Engineering tools **C3.1** Identifying classes of problems and solving methods that are specific to computing systems **C3.2** Using interdisciplinary knowledge, solution patterns and tools, making experiments and interpreting their results **C3.3** Applying solution patterns using specific engineering tools and methods **C3.4** Evaluating, comparatively and experimentally, the available alternative

	solutions for performance optimization **C3.5** Developing and implementing software solutions for specific problems
6.2 Cross competences	N/A

7. Discipline objective (as results from the *key competences gained*)

7.1 General objective	Understand and model requirements, analyse and design appropriate solutions
7.2 Specific objectives	• Identify the most relevant functional and non-functional requirements of a software system and to document them • Design and motivate software architecture for (large scale) software systems • Recognize and apply major software architectural styles, design patterns, and frameworks • Describe a software architecture using various documentation approaches and architectural description languages • Generate architectural alternatives for a problem and select among them

8. Contents

8.1 Lectures	Hours	Teaching methods	Notes
Introduction. SOLID class design principles	2		
GRASP class design principles and package design principles	2		
Architectural styles (Layers, Event-driven, MVC)	2		
Domain-driven design	2		
Service-oriented design	2		
Midterm/Live coding session	2	Face-to-Face lecture, Powerpoint slides, Live streaming	
Enterprise app architectures (Resource Access)	2		
Enterprise app architectures (Presentation)	2		
Enterprise app architectures (Concurrency)	2		
Applying Creational Design Patterns	2		
Applying Structural Design Patterns	2		
Applying Behavioral Design Patterns	2		
Software Design Quality metrics	2		
Final review	2		

Bibliography

1. Juval Lowy, Righting software, O'Reilly, 2020
2. Mark Richards, Software Architecture Patterns, O'Reilly, 2015
3. Vaughn Vernon, Domain Driven Design Distilled, Addison Wesley, 2016
4. Ian Gorton, Essential Software Architecture, Springer, second ed. 2011.
5. Taylor, R., Medvidovic, N., Dashofy, E., Software Architecture: Foundations, Theory, and Practice, 2010, Wiley.
6. Len Bass, Paul Clements, Rick Kazman, Software Architecture in Practice, 3rd edition, 2013.
7. Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sornmerlad, and Michael Stal. 2001. Pattern-oriented system architecture, volume 1: A system of patterns. Hoboken, NJ: John Wiley & Sons. [POSA book]
8. Fowler Martin, Patterns of Enterprise Application Architecture, Addison-Wesley Professional, 2002.
9. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns. AddisonWesley, 1995.
10. Craig Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd Edition), Prentice Hall, 2004, ISBN: 0131489062

Course materials published at moodle.cs.utcluj.ro

8.2 Applications – Seminars/Laboratory/Project	Hours	Teaching methods	Notes
Revision exercises (OOP, UML, testing techniques)	2		
Database connections and operations	2		
Architectural styles exercises	2		
Assignment 1 presentation and discussion	2		
Assignment 1 progress and discussion	2	Face-to-Face tutoring, additional materials	
Domain-driven design exercises	2		
Service-oriented design exercises	2		
Assignment 2 presentation and discussion	2		
Assignment 2 progress and discussion	2		

Design patterns exercises	2		
Design patterns exercises	2		
Assignment 3 presentation and discussion	2		
Assignment 3 progress and discussion	2		
Assignments catch-up session	2		

Bibliography
Course materials published at moodle.cs.utcluj.ro
Java tutorial - docs.oracle.com/javase/tutorial/
C# tutorial – msdn.microsoft.com

Se vor preciza, după caz: tematica seminariilor, lucrările de laborator, tematica și etapele proiectului.

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

ACM Curriculum compliant course

10. Evaluation

Activity type	Assessment criteria	Assessment methods	Weight in the final grade
Course	Ability to understand requirements, analyse alternative solutions and design an appropriate solution	Online exam Moodle, Skype	50%
Seminar			
Laboratory	Analyse requirements and alternative solutions, design an appropriate solution and implement it in either java or C#.	Assignments, project deliverables online Github, Skype	50%
Project			
Minimum standard of performance: Grade calculus: 25% lab + 25% project + 50% final exam Conditions for participating in the final exam: Lab ≥ 5, Project ≥ 5 Conditions for promotion: final exam ≥ 5			

Date of filling in:	Titulari	Titlu Prenume NUME	Semnătura
	Course	Prof.dr.eng. Mihaela Dinsoreanu	
	Applications	Prof.dr.eng. Mihaela Dinsoreanu	

Date of approval in the department	Head of department Prof.dr.ing. Rodica Potolea
Date of approval in the Faculty Council	Dean Prof.dr.ing. Liviu Miclea