SYLLABUS

Design Patterns

University year 2025 - 2026

1. Information regarding the programme

1.1. Higher education institution	Babeş-Bolyai University of Cluj-Napoca
1.2. Faculty	Faculty of Mathematics and Computer Science
1.3. Department	Department of Computer Science
1.4. Field of study	Computer Science
1.5. Study cycle	Bachelor
1.6. Study programme/Qualification	Computer Science
1.7. Form of education	Full time

2. Information regarding the discipline

2.1. Name of the discipline	Design Patterns					Discipline code	**MLE8115**
2.2. Course coordinator					Assoc. Prof. Molnar Arthur-Jozsef		
2.3. Seminar coordinator					Assoc. Prof. Molnar Arthur-Jozsef		
2.4. Year of study	**3**	2.5. Semester	**6**	2.6. Type of evaluation	**C**	2.7. Discipline regime	**Elective**

3. Total estimated time (hours/semester of didactic activities)

3.1. Hours per week	**3**	of which: 3.2 course	**2**	3.3 seminar/laboratory/project	**1**
3.4. Total hours in the curriculum	**36**	of which: 3.5 course	**24**	3.6 seminar/laboratory/project	**12**
Time allotment for individual study (ID) and self-study activities (SA)					**hours**
Learning using manual, course support, bibliography, course notes (SA)					20
Additional documentation (in libraries, on electronic platforms, field documentation)					20
Preparation for seminars/labs, homework, papers, portfolios and essays					30
Tutorship					14
Evaluations					2
Other activities:					3
3.7. Total individual study hours	**89**				
3.8. Total hours per semester	**125**				
3.9. Number of ECTS credits	**5**				

4. Prerequisites (if necessary)

4.1. curriculum	Fundamentals of Programming, Object-Oriented Programming
4.2. competencies	Good programming skills in a programming language that supports the object-oriented paradigm (preferably one of Python, C++, Java or C#)

5. Conditions (if necessary)

5.1. for the course	Classroom with video-projector and Internet access.
5.2. for the seminar /lab activities	Classroom with video-projector and Internet access.

6.1. Specific competencies acquired [1]

Professional/essential competencies	• Advanced programming skills in high-level programming languages • Development and maintenance of software systems
Transversal competencies	• Application of organized and efficient work rules, of responsible attitudes towards the didactic-scientific field, to bring creative value to own potential, with respect for professional ethics principles and norms • Use of efficient methods and techniques to learn, inform, research and develop the abilities to bring value to knowledge, to adapt at the requirements of a dynamical society and to communicate efficiently in the Romanian language and in an international language

6.2. Learning outcomes

Knowledge	• The graduate has the necessary knowledge for using computers, developing software programs and applications, information processing. • The graduate is able to apply architectural styles, design patterns and best practices in the field to design software applications of high complexity. • The graduate has the ability to understand and use design patterns for application development.
Skills	• The graduate has the necessary skills for computer program design and software systems analysis. • The graduate has the ability to apply general rules to specific problems and produce relevant solutions.
Responsibility and autonomy:	• The graduate has the ability to understand and communicate information effectively. • The graduate is able to combine diverse information to formulate solutions and generate ideas for developing new products and applications. • The graduate has the ability to choose and use programming paradigms (procedural, object-oriented, functional) to develop software applications appropriate for the specific domain of the application being developed. • The graduate is able to present and explain methods, algorithms, paradigms and techniques used in various branches of computer science.

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	Enhance student understanding of software design concepts and patterns through a pragmatic, empirical approach.
7.2 Specific objective of the discipline	• Give students the ability to explore various object-oriented programming languages. • Provide students with an environment in which they can explore the usage and usefulness of software design concepts in various business scenarios. • Induce a realistic and industry driven view of software design concepts such as design patterns and their inherent benefits. • Provide students with insights into ways of working towards achieving high quality software.

[1] One can choose either competences or learning outcomes, or both. If only one option is chosen, the row related to the other option will be deleted, and the kept one will be numbered 6.

8. Content

8.1 Course	Teaching methods
Object oriented programming (OOP) principles Recap presentation that covers the OOP principles of encapsulation, inheritance, polymorphism, cohesion, coupling, aggregation, composition.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration
SOLID principles Their role as important principles behind high quality software: single responsibility principles, open-closed principle, Liskov substitution, interface segregation and dependency inversion.	
Creational design patterns (Abstract) Factory, Builder, Prototype, Singleton	
Structural design patterns Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy	
Behavioural design patterns Chain of responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, Template, Visitor.	
Antipatterns Common responses to recurring problems that are usually ineffective and risk being highly counterproductive. A selection of antipatterns with code examples illustrating some of the most encountered issues in source code (e.g., golden hammer, God class, spaghetti code)	
Dark Patterns in user experience design Deceptive design patterns applied to the user experience/interface aiming to trick users into doing things they otherwise would not. A selection of dark patterns with examples from real applications (e.g., drip pricing, roach motel, misdirection).	
Model-View-* design patterns A selection of higher-level design patterns (Model View Controller, Model View ViewModel, Model View Presenter)	
Enterprise and architecture design patterns A selection of high-level design patterns encountered in enterprise applications requiring high availability and performance (e.g., message routing, pipes and filters, peer to peer, microservices).	

Bibliography

1. M. Fowler – ***Patterns of Enterprise Application Architecture***, Addison Wesley, 2003
2. E. Freeman, E. Freeman, B. Bates – ***Head-First Design Patterns***, Oreilly, 2004
3. E. Gamma, R. Helm, R.Johnson, J. Vlissides – ***Design Patterns: Elements of Reusable Object-Oriented Software***, Addison Wesley, 1995
4. William J. Brown, Raphael C. Malveau, Hays W. "Skip" McCormick, Thomas J. Mowbray - ***AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis***, Wiley, 1998.
5. Harry Brignull - ***Deceptive Patterns, Exposing the Tricks Tech Companies Use to Control You*** (free edition on https://www.deceptive.design/book/contents/get-started), Testimonium Ltd., 2023.

8.2 Seminar / laboratory	Teaching methods
OOP concepts recap. Introduction to laboratory activities and grading	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration
SOLID principles. Creational design patterns	
Structural design patterns. Checking progress of laboratory project	
Behavioural design patterns. Checking progress of laboratory project	
Antipatterns. Dark patterns. Architectural and enterprise patterns	
Laboratory project turn-in	

Bibliography

1. M. Fowler – ***Patterns of Enterprise Application Architecture***, Addison Wesley, 2003
2. E. Freeman, E. Freeman, B. Bates – ***Head-First Design Patterns***, Oreilly, 2004
3. E. Gamma, R. Helm, R.Johnson, J. Vlissides – ***Design Patterns: Elements of Reusable Object-Oriented Software***, Addison Wesley, 1995
4. William J. Brown, Raphael C. Malveau, Hays W. "Skip" McCormick, Thomas J. Mowbray - ***AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis***, Wiley, 1998.
5. Harry Brignull - ***Deceptive Patterns, Exposing the Tricks Tech Companies Use to Control You*** (free edition on https://www.deceptive.design/book/contents/get-started), Testimonium Ltd., 2023.

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

- The course respects the IEEE and ACM Curricula Recommendations for Computer Science studies.
- The course exists in the study program of all major universities in Romania and abroad.
- The content of the course is considered by software companies as important for average programming skills.

10. Evaluation

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	Presentation during the semester	Grading based on presentation quality, thoroughness and suitability of examples selected.	**25%**
	Examination during the final week.		**50%**
10.5 Seminar/laboratory	Laboratory project		**25%**
10.6 Minimum standard of performance			
• Students must observe the standards of academic integrity. • A minimum passing grade is defined by attaining at least 50% (5/10) points in the final grade.			

11. Labels ODD (Sustainable Development Goals)[2]

Not applicable.

Date:
28.04.2025

Signature of course coordinator

Assoc. Prof. Molnar Arthur-Jozsef

Signature of seminar coordinator

Assoc. Prof. Molnar Arthur-Jozsef

Date of approval:
...

Signature of the head of department

Assoc.prof.phd. Adrian STERCA

[2] Keep only the labels that, according to the *Procedure for applying ODD labels in the academic process*, suit the discipline and delete the others, including the general one for *Sustainable Development* – if not applicable. If no label describes the discipline, delete them all and write „*Not applicable.*".