SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babeş Bolyai University**
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**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)			**Design Patterns**				
2.2 Course coordinator			**Lect. PhD. Arthur Molnar**				
2.3 Seminar coordinator							
2.4. Year of study	**3**	2.5 Semester	**6**	2.6. Type of evaluation	**C**	2.7 Type of discipline	**Opt**
2.8 Code of the discipline		**MLE8115**					

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

3.1 Hours per week	4	Of which: 3.2 course	2	3.3 seminar/laboratory	2
3.4 Total hours in the curriculum	48	Of which: 3.5 course	24	3.6 seminar/laboratory	24
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					20
Additional documentation (in libraries, on electronic platforms, field documentation)					20
Preparation for seminars/labs, homework, papers, portfolios and essays					20
Tutorship					18
Evaluations					10
Other activities:					-

3.7 Total individual study hours	107
3.8 Total hours per semester	155
3.9 Number of ECTS credits	7

4. Prerequisites (if necessary)

4.1. curriculum	• Fundamentals of Programming • Object Oriented Programming
4.2. competencies	• Good programming skills in Java or C#

5. Conditions (if necessary)

5.1. for the course	• Lecture hall with projector
5.2. for the seminar /lab activities	• Computers with installed IDE for Java/C# development

6. Specific competencies acquired

Professional competencies	C 2.1 Identify adequate software systems development methodologies C 1.1 Proper description of programming paradigms and language specific mechanisms, and identification of semantical an syntactical differences C4.3. Identify models and methods adequate to real life problem solving
Transversal competencies	CT1 Apply rules to: organized and efficient work, responsibilities of didactical and scientifically activities and creative capitalization of own potential, while respecting principles and rules for professional ethics CT3 Use efficient methods and techniques for learning, knowledge gaining, and research and develop capabilities for capitalization of knowledge, accommodation to society requirements and communication in English

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

7.1 General objective of the discipline	• Enhance students' understanding of software design concepts through a pragmatic approach • 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
7.2 Specific objective of the discipline	• Give students the ability to explore various object oriented programming languages • Improve the students abilities to tackle business requirements • Enhance the students understanding of business needs and business value • Provide students with insights into the way of working towards achieving high quality software through skilled trainers from the IT industry

8. Content

8.1 Course	Teaching methods	Remarks
1. OOP Principles Recap: Recap presentation that mostly covers main OOP principles such as encapsulation, polymorphism, cohesion, coupling, aggregation, composition	description, explanation, example, case studies, dialogue,	-
2. SOLID principles: base principles of high		-

quality software: Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion	debate	
3. Creational Patterns (Factory, Builder, Prototype, Singleton)		-
4. Structural Patterns (Adapter, Bridge, Composite, Decorator, Facade, Proxy)		-
5. Behavioral Patterns (Chain of Responsibility, Command, Iterator, Mediator, Observer)		-
6. Behavioral Patterns (State, Strategy, Template)		-
7. Recap (Factory, Builder, Singleton, Adapter, Composite, Proxy)		-
8. Architectural Patterns (MVVM, MVP, MVC), JS Module Pattern		-
9. Enterprise Integration Patterns		-
10. Enterprise Integration Patterns: Messaging		-
11. Enterprise Integration Patterns: Message Routing		-
12. Enterprise Integration Patterns: Message Translating		-
13. Antipatterns: common responses to recurring problems that are usually ineffective and risk being highly counterproductive		-
14. Design Patterns: Recap Exercises		-
Bibliography 1. M. Fowler – Patterns of Enterprise Application Architecture, Aison 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		
8.2 Seminar / laboratory	Teaching methods	Remarks
1. Advanced UML elements, requirements analysis	Explanation, dialogue, case studies	-
2. SOLID workshop based on business use cases		-
3. Creational Design Patterns workshop based on business use cases		-
4. Structural Design Patterns workshop based on business use cases		-
5. Behavioural Design Patterns workshop based on business use cases		-
6. Antipatterns workshop based on business use cases		-
7. Final project turn-in		-
Bibliography 1. M. Fowler – Patterns of Enterprise Application Architecture, Aison 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		

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 Curriculla Recommendations for Computer Science studies;
- The course exists in the studying program of all major universities in Romania and abroad;
- The content of the course is considered the software companies as important for advanced programming skills

10. Evaluation

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
	Final project: architecture & design pattern application	Project grading	50%
10.5 Seminar/lab activities	Individual presentations during the semester	Grading based on presentation quality, thoroughness and suitability of examples selected.	50%
10.6 Minimum performance standards			
➢ Students must observe the standards of academic integrity. ➢ A minimum passing grade is defined by attaining at least 50% (5/10) points for the final project and each of the three lab assignments respectively.			

Date	Signature of course coordinator	Signature of seminar coordinator
20.04.2018	Lect. PhD. Arthur Molnar	Lect. PhD. Arthur Molnar

Date of approval ..... Signature of the head of department .....….....