

## syllabus

### 1. Information regarding the programme

1.1 Higher education institution	<b>Babeş Bolyai University</b>
1.2 Faculty	<b>Faculty of Mathematics and Computer Science</b>
1.3 Department	<b>Department of Computer Science</b>
1.4 Field of study	<b>Computer Science</b>
1.5 Study cycle	<b>Bachelor</b>
1.6 Study programme / Qualification	<b>Computer Science</b>

### 2. Information regarding the discipline

2.1 Name of the discipline	<b>Design Patterns</b>						
2.2 Course coordinator	<b>PhD. Amalia Cifor</b>						
2.3 Seminar coordinator							
2.4. Year of study	<b>3</b>	2.5 Semester	<b>6</b>	2.6. Type of evaluation	<b>C</b>	2.7 Type of discipline	<b>Optional</b>

### 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	1
3.4 Total hours in the curriculum	42	Of which: 3.5 course	28	3.6 seminar/ laboratory	14
Time allotment:					Hours
Learning using manual, course support, bibliography, course notes					20
Additional documentation (in libraries, on electronic platforms, field documentation)					30
Preparation for seminars/labs, homework, papers, portfolios and essays					30
Tutorship					18
Evaluations					10
Other activities: .....					-
3.7 Total individual study hours			108		
3.8 Total hours per semester			150		
3.9 Number of ECTS credits			6		

### 4. Prerequisites (if necessary)

4.1. curriculum	<ul style="list-style-type: none"> <li>• OOP, Programming Fundamentals</li> </ul>
4.2. competencies	<ul style="list-style-type: none"> <li>• Good programming skills in at least one of the languages Java, C#</li> </ul>

### 5. Conditions (if necessary)

5.1. for the course	<ul style="list-style-type: none"> <li>• Course hall with projector</li> </ul>
5.2. for the seminar /lab activities	<ul style="list-style-type: none"> <li>• Laboratory: computers and use of a programming language environment</li> </ul>

### 6. Specific competencies acquired

<b>Professional competencies</b>	<p>C 2.1 Identify adequate software systems development methodologies</p> <p>C 1.1 Proper description of programming paradigms and language specific mechanisms, and identification of semantical and syntactical differences</p> <p>C4.3. Identify models and methods adequate to real life problem solving</p>
----------------------------------	--

<b>Transversal competencies</b>	<p><b>CT1</b> Apply rules to: organized and efficient work, responsibilities of didactical and scientific activities and creative capitalization of own potential, while respecting principles and rules for professional ethics</p> <p><b>CT3</b> 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</p>
---------------------------------	--

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

7.1 General objective of the discipline	<ul style="list-style-type: none"> <li>-Enhance the students understanding of software design concepts through a practical and pragmatic approach</li> <li>- provide the students with an environment in which they can explore the usage and usefulness of software design concepts in various business scenarios</li> <li>-Induce a realistic and industry driven view of software design concepts such as design patterns and their inherent benefits</li> </ul>
7.2 Specific objective of the discipline	<ul style="list-style-type: none"> <li>-Give students the ability to explore various object oriented programming languages</li> <li><input type="checkbox"/> Improve the students abilities to tackle business requirements</li> <li>- Enhance the students understanding of business needs and business value</li> <li><input type="checkbox"/> Provide students with insights into the way of working towards achieving high quality software through skilled trainers from the IT industry</li> </ul>

### 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	exposure: description, explanation, example, case studies, dialogue, debate	
2. SOLID principles: base principles of high quality software: Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion	exposure: description, explanation, examples, discussion of case studies	
3. Creational Design Patterns: Factory, Abstract Factory, Builder, Prototype, Singleton	exposure: description, explanation, example	
4. Creational Design Patterns: part 2	exposure: description, explanation, example; dialogue, case studies	
5. Structural Design Patterns: Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy	exposure: description, explanation, example, dialogue, debate	
6. Structural Design Patterns: part 2	exposure: description, explanation, example, discussion of case studies	
7. Behavioural Design Patterns: Chain of responsibility, Command, Iterator, Mediator, Memento, Observer, State	exposure: description, explanation, example, dialogue, debate, case studies	
8. Behavioural Design Patterns: part 2	exposure: description, explanation, example, case studies, dialogue, debate	
9. Concurrency Design Patterns: Active Object, Barrier, Monitor Object, Read Write Lock, Scheduler,	exposure: description, explanation, example, case studies, dialogue, debate	
10. Concurrency Design Patterns: part 2	exposure: description, explanation, example, case studies, dialogue, debate	
11. Concurrency Design Patterns: part 3		

12. Architectural Patterns: Layered, MVC, MVVM, MVP, Client-Server	exposure: description, explanation, example, case studies, dialogue, debate	
13. Enterprise Integration Patterns	exposure: description, explanation, example, case studies, dialogue, debate	
14. Antipatterns: common responses to recurring problems that are usually ineffective and risk being highly counterproductive	exposure: description, explanation, example, case studies, dialogue, debate	
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	Explation, dialogue, case studies	
2. SOLID workshop based on business use cases	Explation, dialogue, case studies	
3. Creational Design Patterns workshop based on business use cases	Explation, dialogue, case studies	
4. Structural Design Patterns workshop based on business use cases	Evaluation	
5. Behavioural Design Patterns workshop based on business use cases	Explation, dialogue, case studies	
6. Antipatterns workshop based on business use cases	Explation, dialogue, case studies	
7. Final project turn-in	Evaluation	
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 Curricula 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 (%)
10.4 Course	Final project: architecture & design pattern application	Project grading	40%
10.5 Seminar/lab activities	Assignment 1: creational design patterns	-Mini-project grading	20%
	Assignment 2: structural design patterns	-Mini-project grading	20%
	Assignment 1: behavioural design patterns	-Mini-project grading	20%
10.6 Minimum performance standards			
<ul style="list-style-type: none"> <li>- 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.</li> <li>- No more than 3 absences are allowed for the seminar/lab activities.</li> </ul>			

Date  
.....

Signature of course coordinator  
PhD. Amalia Cifor

Signature of seminar coordinator

Date of approval  
.....

Signature of the head of department  
.....