**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 | **Design Patterns** | | | | | | |
| 2.2 Course coordinator | **Ph.D. Silviu Dumitrescu** | | | | | | |
| 2.3 Seminar coordinator | **Lucian Brăescu** | | | | | | |
| 2.4. Year of study | **3** | 2.5 Semester | **5** | 2.6. Type of evaluation | **C** | 2.7 Type of discipline | **Optional** |

**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 | | | | | 8 |
| Additional documentation (in libraries, on electronic platforms, field documentation) | | | | | 7 |
| Preparation for seminars/labs, homework, papers, portfolios and essays | | | | | 8 |
| Tutorship | | | | | 2 |
| Evaluations | | | | | 8 |
| Other activities: ..... | | | | | |

| | |
|---|---|
| 3.7 Total individual study hours | 33 |
| 3.8 Total hours per semester | 75 |
| 3.9 Number of ECTS credits | 6 |

**4. Prerequisites** (if necessary)

| | |
|---|---|
| 4.1. curriculum | • OOP, Programming Fundamentals |
| 4.2. competencies | • Good programming skills in at least one of the languages Java, C# |

**5. Conditions** (if necessary)

| | |
|---|---|
| 5.1. for the course | • Course hall with projector |
| 5.2. for the seminar /lab activities | • Laboratory |

**6. Specific competencies acquired**

| | |
|---|---|
| **Professional competencies** | C 4.3 Identify models and methods adequate to real life problem solving<br>C 2.1 Identify adequate software systems development methodologies<br>C 1.1 Proper description of programming paradigms and language specific mechanisms, and identification of semantical an syntactical differences |
| **Transversal competencies** | **CT1** Apply organized and efficient work rules and responsible attitude towards didactical and research field, in order to creatively use work potential; respect professional ethical principles<br>**CT3** Use efficient methods and techniques for: learning, information search, research and development of capacities to adapt to the requirements of a dynamic society and to communicate in an international language |

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

| | |
|---|---|
| 7.1 General objective of the discipline | • Enhance the students understanding of software design concepts through a practical and pragmatic approach<br>• Provide the students with an environment in which they can explore the usage and usefulness of software design concepts in various business scenarios<br>• 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<br>• Improve the students abilities to tackle business requirements<br>• Enhance the students understanding of business needs and business value<br>• 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 | 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, example, case studies, dialogue, debate | |
| 3. Creational Design Patterns: Factory, Abstract Factory, Builder, Prototype, Singleton | exposure: description, explanation, example, | |

| | | |
|---|---|---|
| | case studies, dialogue, debate | |
| 4. Creational Design Patterns: part 2 | exposure: description, explanation, example, case studies, dialogue, debate | |
| 5. Structural Design Patterns: Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy | exposure: description, explanation, example, case studies, dialogue, debate | |
| 6. Structural Design Patterns: part 2 | exposure: description, explanation, example, case studies, dialogue, debate | |
| 7. Behavioural Design Patterns: Chain of responsibility, Command, Iterator, Mediator, Memento, Observer, State | exposure: description, explanation, example, case studies, dialogue, debate | |
| 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, Mediator, Thread pool, Thread local storage | 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 | exposure: description, explanation, example, case studies, dialogue, debate | |
| 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:<br>1. M. Fowler – Patterns of Enterprise Application Architecture, Aison Wesley, 2003<br>2. E. Freeman, E. Freeman, B. Bates – Head First Design Patterns, Oreilly, 2004<br>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 part 1 | Explation, dialogue, case studies | |
| 4. Structural Design Patterns workshop based on business use cases part 1 | Explation, dialogue, case studies | |
| 5. Behavioural Design Patterns workshop based on business use cases part 1 | Explation, dialogue, case studies | |
| 6. Antipatterns workshop based on business use cases part 1 | Explation, dialogue, case studies | |
| 7. Final project turn-in | Explation, dialogue, case studies | |

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 abroad;
- The content of the course is considered important for advanced programming skills by the software companies

**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 3: behavioural design patterns | Mini-project grading | 20% |
| 10.6 Minimum performance standards | | | |
| ➢ 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.<br>➢ No more than 3 absences are allowed for the seminar/lab activities. | | | |

| Date | Signature of course coordinator | Signature of seminar coordinator |
| --- | --- | --- |
| ..... | PhD. Silviu Dumitrescu | Lucian Brăescu |

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