SYLLABUS

Object Oriented Programming

1. Information on academic programme

1.1. University	**„1 Decembrie 1918" din Alba Iulia**
1.2. Faculty	**Faculty of Computer Science and Engeneering**
1.3. Department	**Exact Sciences and Engineering Department**
1.4. Field of Study	**Computer Science**
1.5. Cycle of Study	**Bachelor**
1.6. Academic program / Qualification	**Computer Science**

2. Information of Course Matter

2.1. **Course**	*Object Oriented Programming*			2.2. **Code**		CSE 204	
2.3. **Course Leader**		Rotar Corina					
2.4. **Seminar Tutor**		Cristea Daniela					
2.5. Academic Year	**II**	2.6. Semester	**I**	2.7. Type of Evaluation (E – final exam/ CE - colloquy examination / CA -continuous assessment)	**E**	2.8. Type of course (**C**– Compulsory, **Op** – optional, **F** - Facultative)	**C**

3. Course Structure (Weekly number of hours)

3.1. Weekly number of hours	**4**	3.2. course	***2***	3.3. seminar, laboratory	***2***
3.4. Total number of hours in the curriculum	**56**	3.5. course	***28***	3.6. seminar, laboratory	***28***

Allocation of time:	Hours
Individual study of readers	***10***
Documentation (library)	***20***
Home assignments, Essays, Portfolios	***28***
Tutorials	**-**
Assessment (examinations)	***11***
Other activities…….	**-**

3.7 Total number of hours for individual study	**69**
3.8 Total number of hours in the curriculum	**56**
3.9 Total number of hours per semester	**125**
3.10 Number of ECTS	**5**

3. Prerequisites (*where applicable*)

4.1. curriculum-based	*Data Structures*
4.2. competence-based	**C1 Programming in high-level languages** C1.1 The appropriate description of programming paradigms and of specific language mechanisms, as well as the identification of differences between semantic and syntactic

	aspects. C1.2 The explaining of existing software applications using different abstraction layers (architecture, packages, classes, methods), correctly using base knowledge. C1.3 The development of correct source codes and the testing of various components in a known programming language, given a set of design specifications. C1.4 The testing of various applications given specific testing plans C1.5 Developing program units and their documentation.

4. **Requisites** (*where applicable*)

5.1. course-related	*Room equipped with video projector / boar*
5.2. seminar/laboratory-based	*Laboratory – computer, Software: Visual Studio 2010, BorlandC/Codeblocks/DevC++, Internet access.*

5. Specific competences to be aquired (chosen by the course leader from the programme general competences grid)

Professional competences	C1 Programming in high-level languages C2 Development and maintenance of computer applications
Transversal competences	*Not applicable*

6. Course objectives (as per the programme specific competences grid)

7.1 General objectives of the course	Develop students' ability to design software that is dedicated to solving medium complexity problems by using object oriented paradigm. Deepening the concept of class and object, and gaining the skills to design classes and associated libraries. Creating a rigorous and efficient object oriented programming style
7.2 Specific objectives of the course	Developing students' ability to effectively manage information by using classes and relations between classes. Drawing a coherent documentation on the applications of average complexity.

7. Course contents

8.1 Course (learning units)	**Teaching methods**	**Remarks**
1. Object-oriented programming paradigm. Basic concepts.	***Lecture, conversation, exemplification***	
2. Programming with data abstraction. Features in C + +.	***Lecture, conversation, exemplification***	
3. Classes and objects. Data members and methods.	***Lecture, conversation, exemplification***	
4. Constructors and destructor. Copy constructor	***Lecture, conversation, exemplification***	
5. *Static* keyword in classes.	***Lecture, conversation, exemplification***	
6. *friend* keyword.Overloading binary operators.	***Lecture, conversation, exemplification***	
7. Overloading operators (II).	***Lecture, conversation, exemplification***	

8. Conversions.	***Lecture, conversation, exemplification***	
9. Derived classes, base classes. Inheritance.	***Lecture, conversation, exemplification***	
10. Inheritance. Multiple inheritance.	***Lecture, conversation, exemplification***	
11. Virtual methods	***Lecture, conversation, exemplification***	
12. Polymorphism.	***Lecture, conversation, exemplification***	
13. Generic classes.	***Lecture, conversation, exemplification***	
14. Exceptions. Standard Inputs-Outputs.	***Lecture, conversation, exemplification***	
Seminars-laboratories	**Teaching methods**	
Introduction to OOP	***Project-work, computer-based activities, laboratory activities***	
Classes as abstract data types in C++.	***laboratory activities***	
Classes. Structure of a class. Components: attributes, methods. Examples.	***laboratory activities***	
Public, private, protected. Examples.	***laboratory activities***	
Constructors and destructors. Applications.	***laboratory activities***	
Operators. Operator overloading.	***laboratory activities***	
Visual Studio. NET, C #. Console applications.	***laboratory activities***	
Standard classes and user classes. Defining classes in C #.	***laboratory activities***	
Heritage. Friend classes. Examples in C + + than C #.	***laboratory activities***	
Static and virtual methods. Static and dynamic binding. Design and implementation of virtual methods.	***laboratory activities***	
Windows applications using predefined programming classes in C #.	***laboratory activities***	
Polymorphism. Examples.	***laboratory activities***	

References

1. Bruce Eckel, Thinking in C++, free online.
2. Bjarne Stroustrup, The C++ Programming Language, Addison Wesley, 1997.
3. H. Schildt: C++ manual complet, e-book.
4. Peter Muller: Introduction to Object-Oriented Programming Using C++ , e-book.
5. Rotar C., Object oriented Programming - *Lecture notes*

1. Corroboration of course contents with the expectations of the epistemic community's significant representatives, professional associations and employers in the field of the academic programme

Not applicable

2. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	*Final evaluation*	*Written paper*	60%
	-	-	-
10.5 Seminar/laboratory	*Continuous assessment*	*Laboratory activities portfolio*	40%
	-		-
10.6 Minimum performance standard:			
Implementation and documentation of the software units in an object oriented programming language and efficiently using the related concepts.			

Submission date _____

Course leader signature _____

Seminar tutor signature _____

Date of approval by Department members _____

Department director signature _____