COURSE UNIT DESCRIPTION

Course unit title	Course unit code
Object-Oriented Programming	ITOP

Annotation
Course focuses on essential object-oriented concepts and principles, encouraging information hiding and abstraction-based thinking, and provides ideas and techniques to improve code readability, maintenance, and adaptability to change. Course uses C++ and Java programming languages, and also addresses various tools and technologies as well as fundamental sofware design patterns as solutions for typical problems. Course is intended for those having some background in programming (ideally C, C++, Java, or a similar language).

Lecturer	Department where the course unit is delivered
Coordinator(s): Irmantas Radavičius	Department of Computer Science II Faculty of Mathematics and Informatics Vilnius University

Cycle	Type of the course unit
First	Compulsory

Mode of delivery	Semester or period when the course unit is delivered	Languages of instruction
Auditorinė	2nd semester	English and Lithuanian

Course requirements
Prerequisites: Programming fundamentals and basic IT knowledge

Number of ECTS credits allocated	Student's workload	Contact hours	Individual work
5	141	66	75

Purpose of the course unit: programme competences to be developed
Generic competences to be developed • Ability for abstract thinking , processing and analysing information (BK3) • Ability to use information and communications technologies (BK5) **Subject-specific competences to be developed** • Ability to apply general methods of the program design, make and analyse software requirements (DK1) • Ability to analyse the algorithmic process of the task based on the general properties of the algorithm (DK2) • Ability to develop the software project (or IT service) and to write its specification (DK3)

Learning outcomes of the course unit	Teaching and learning methods	Assessment methods
Ability to understand, write, modify, and execute given code (in C++ or Java)	Lectures Studies of literature Programing projects	Code reviews Programming tests, exam
Ability to understand key concepts and principles of object oriented programming paradigm, recognize and apply them in practice		
Ability to test and review given code, evaluate its correspondence to the requirements and good practices	Programing projects Code reviews	Programming projects Code reviews
Ability to understand, write, and evaluate reports and specifiations and their correspondence to the requirements and the provided code		

Ability to search the official documentation for packages, classes, methods, examples, and select the necessary information.	Studies of literature Programing projects	Programming tests, exam

Course content: breakdown of the topics	**Contact hours**						**Individual work: time and assignments**	
	Lectures	Consultations	Seminars	Exercises	Laboratory work	**Contact hours**	**Individual work**	**Assignments**
Course overview. Programming primitives. Data types, values and variables. Control structures.	4				4	**8**	**4**	Individual projects
Style requirements and good practices. Quality code. Code reviews and unit testing.	4				4	**8**	**4**	
Object-oriented programming paradigm. Classes and objects. Fields and methods. Instance and static members, access control, encapsulation. Principle of information hiding. Object lifecycle, constructors and destructors, memory management.	4				4	**8**	**4**	
Exception handling. Multifile programs, modular programming. Input and output streams. User interface.	4				4	**8**	**4**	
Composition. Delegation and single responsibility principle. Containers. Shallow and deep object copying.	4				4	**8**	**4**	
Object oriented analysis and design. UML language and tools, fundamental UML diagrams	4				4	**8**	**4**	Team project
Abstract classes and interfaces. Inheritance. Polymorphism. Object-oriented principles.	4				4	**8**	**4**	
Programming tests and feedback	4				4	**8**	**4**	
Programming projects							**8**	
Exam						**2**	**3**	
Total:	32				32	**66**	**75**	

Assessment strategy	**Weight (perc.)**	**Deadline**	**Assessment criteria**
Individual projects	30	During the semester	During the semester each student is asked to provide five submissions for three different projects, as well as five types of reviews for other people in the course. Each stage (out of 5) is worth 0.6 points.
Team project	30	Last month of the semester	The team project is carried out in small groups of students (2-4 people). The project has three parts (each worth 1 point): 1) project proposal (team roles, UML diagrams, proof of concept etc) 2) project essentials (essential functionality) 3) project demonstration (demonstration and documentation) Each student is evaluated individually, based on their contribution and competence corresponding their role(s) in the current project stage, their own work as well as quality of the reviews given for other projects in the course.

Programming tests	20	First test – Week 7 Second test – Week 15	Each test is worth 1 point, while exam is worth 2 points. Each of them contains multiple problems of various types and difficulty to solve (50%) as well as a programming assignment (50%). Evaluation is based on the correctness of the solution as well as (for the programming assignment) adherence to proposed good coding practices and general code quality. If a student gets less than 50% of the maximal score, it is evaluated as 0. To be allowed to take the exam, during the semester each student has to provide at least 24 (proper) reviews and make at least 5 (proper) submissions (out of 8 for both individual and team projects). To pass the course, the total sum of points for the exam has to be at least 50% of the maximum, otherwise the student must repeat the course.
Exam	20	During the exam session	
Externe students		This course can be attended by externe, provided it was attended normally before, and the points received previously are considered appropriate. In such case the externe gets to only repeat the exam. For this, the externe must inform the lecturer in the beginning of the semester, getting a written confirmation with an account of points to be transfered from an earlier year. If the amount of points is not sufficient and thus work and re-evaluation during the semester is required, externe attendance is not allowed.	

Author	**Year**	**Title**	**Issue or vol.**	**Publishing house or Internet site**
Required reading				
A. Brilingaitė	2012	Object-Oriented Programming. Study Guide		
Bruce Eckel	2000	Thinking in C++	2nd ed.	https://archive.org/details/TICPP2ndEdVolOne
Bruce Eckel	2003	Thinking in C++, Volume Two: Practical Programming	1st ed.	https://archive.org/details/TICPP2ndEdVolTwo
Oracle		The JavaTM Tutorials		https://docs.oracle.com/javase/tutorial/index.html
Bruce Eckel	2006	Thinking in Java	4th ed.	Pearson
Recommended reading				
Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides	1994	Design Patterns: Elements of Reusable Object-Oriented Software	1st ed.	Addison-Wesley Professional
C. Larman	2015	Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative development		Prentice-Hall