COURSE UNIT DESCRIPTION

Course unit title	Course unit code
Software engineering I	

Lecturer(s)	Department where the course unit is delivered
Coordinator: Lect. Tomas Smagurauskas **Other lecturers: -**	Department of Software Engineering Faculty of Mathematics and Informatics Vilnius University

Cycle	Type of the course unit
1st (BA)	Compulsory

Mode of delivery	Semester or period when the course unit is delivered	Language of instruction
Face-to-face	3rd semester	English, Lithuanian

Prerequisites
Prerequisites: Procedural programming, Object-oriented Programming

Number of credits allocated	Student's workload	Contact hours	Individual work
10	270	82	188

Purpose of the course unit: programme competences to be developed

Purpose of the course unit – to get acquainted with software development methods using C # programming language and .NET framework, to consolidate knowledge of object-oriented programming.

Generic competences:

- Communication and collaboration *(GC1)*.
  - An ability to present information, ideas, problems, and suggested solutions convincingly in official and second (foreign) language for specialists and non-specialists in written and verbal form *(GC1.1)*.
- Life-long learning *(GC2)*.
  - Recognition of the need for, and engagement in life-long learning *(GC2.1)*.
- Social responsibility *(GC3)*
  - An ability to analyse the economic, social, ethical, and legal impact of engineering solutions on individuals, organizations, and society *(GC3.2)*.

Specific competences:

- Knowledge and skills of underlying conceptual basis *(SC4)*.
  - Knowledge and understanding of the key aspects and concepts of software engineering, including some at the forefront of the discipline, insight into possible application fields, and an awareness of the wider spectrum of the discipline *(SC4.1)*.
  - An ability to apply mathematical foundations, knowledge of science and engineering, computer science theory, and algorithmic principles in software systems development *(SC4.2)*.
- Software development knowledge and skills *(SC5)*.
  - An ability to analyse a problem, identify needs and define the computing requirements appropriate to its solution *(SC5.2)*
  - An ability to design, implement, and evaluate a computer-based system, process, component, or service to meet desired needs *(SC5.3)*.
  - An ability to select the software life cycle suitable for building new, and maintaining and commissioning existing, software systems *(SC5.4)*.
- Technological and methodological knowledge and skills, professional competence *(SC6)*.
  - An ability to combine theory and practice to complete software engineering tasks from different application areas while taking into account the existing technical, economic, and social context *(SC6.1)*.
  - An ability to select and use appropriate current techniques, models, solution patterns, skills, and tools necessary for software engineering practice involving emerging application areas *(SC6.2)*.

Learning outcomes of the course unit: students will be able to	Teaching and learning methods	Assessment methods

Design, implement and develop applied programs, apply code reviews.	Lectures, problem-oriented teaching, teamwork, case studies, information retrieval, literature reading, individual work, learning from teammates, examples analysis, tutorials, laboratory works.	Laboratory works, results presentation, written exam (open, semi-open and close-ended questions and tasks).
Apply knowledge of software systems engineering, make qualified design and architectural decisions while expanding the functionality of the developed system.		
Combine theory and practice using .NET framework technologies and developing OO application systems.		
Develop the knowledge about data types, named and optional arguments as well as other new features of C# programming language.		
Program in C# independently and in a team, applying basic OO design templates using C# programming environment.		
Recognize the need for continuous learning and will have the initial skills.		
Work in the team - on site and remotely.		

Course content: breakdown of the topics	**Contact hours**							**Individual work: time and assignments**	
	Lectures	Tutorials	Seminars	Practice	Laboratory work (LW)	*Tutorial during LW*	**Contact hours**	**Individual work**	**Assignments**
1. Course overview. Acquaintance with C# programming language. Applications build tools, .NET framework compatibility with different operating systems. C# Overview for programmers with OP knowledge. Code versioning systems.	2				2	*8*	**4**	**10**	Self-study of literature to deepen the knowledge. Preparation for laboratory works.
2. C#-specific and OOP-specific properties. Web services in .NET framework using ASP.NET. REST, WCF, GraphQL and gRPC overview and examples.	3				3		**6**	**12**	
3. Graphical web interfaces development using .NET framework. Examples in ASP.NET. Introduction to Agile software development.	3				3		**6**	**15**	
4. Data types hierarchy. Classes, their structure and relationships. Generic types and methods. Conversions. Standard .NET interfaces. Creation of objects. Object lifecycle. Typical OOP mistakes and how to avoid them. Dependency Injection.	3				2		**5**	**8**	
5. Working with data. Data input and output, validation. Collections. Delegates, anonymous methods, lambda expressions.	3				2		**5**	**15**	
6. Introduction to LINQ. Working with data. Dependency injection.	4				2		**6**	**12**	

7. Unit and integration tests - principles, terminology. Tests in .NET environment.	4				2		**6**	**15**	
8. Working with databases. Introduction to ORM. Overview of main ORM's. Entity Framework Core ORM implementation in .NET Core applications. LINQ use cases when working with databases.	4				2		**6**	**15**	
9. Software system construction. Key goals and challenges. Business needs analysis. Software system modification and maintenance. Generic types in C# language and .NET Framework.	4				3		**7**	**15**	
10. Introduction to creating and improving the user experience. Introduction to graphical interface development.	4				2		**6**	**12**	
11. Introduction to multithreading. Async/Await. Real world examples.	4				4		**8**	**15**	
12. Improving the software development process. Automated tools for code quality assurance. Continuous integration. Software deployment process.	2				2		**4**	**8**	
13. Debugging and refactoring.	3				1		**4**	**10**	
14. Overview of .NET technologies. Introduction to design patterns (MQ, CQRS etc.). Analysis of modern OO systems.	3				2		**5**	**10**	
15. Preparation for the exam and taking the final exam (written).		2					**4**	**16**	2h. tutorials 2h exam
Total	**48**	**2**			**32**	*8*	**82**	**188**	

Assessment strategy	**Weight %**	**Deadline**	**Assessment criteria**
Laboratory assignment No. 1	15	Week 7	The collaborative laboratory work assigned to the students covers the knowledge and skills that were developed in 1-6 topics. Student teams for collaborative work are recommended to be from 3 to 5 students, and for them not to rotate during the semester. All laboratory assignments must be placed in a code repository and all the code must be reviewed by teammates. Lateness leads to the decrease of the maximal assessment (1.5) by 20% of every delayed week. Each student in the team is evaluated separately, according to the student's responses during the review of the assignment, according to student's activeness during the previous laboratories, according to code reviews and created code scope. Partially finished laboratory work or without following the given requirements is evaluated accordingly.
Laboratory assignment No. 2	15	Week 11	The collaborative laboratory work is continued, covering the knowledge and skills that were developed in 7-10 topics. All laboratory assignments must be placed in code repository and all the code must be reviewed by teammates. Lateness leads to the decrease of the maximal assessment (1.5) by 20% of every delayed week. Each student in the team is evaluated separately, according to the student's responses during the review of the assignment, according to student's activeness during the previous laboratories, according to code reviews and created code scope. Partially finished laboratory work or without following the given requirements is evaluated accordingly.
Laboratory assignment No. 3	20	Week 15	The individual laboratory work assigned to the students covers the knowledge and skills that were developed in 11-13 topics. Assignments require using databases. All laboratory assignments must be placed in the code repository and all the code must be reviewed by teammates. Lateness leads to the decrease of the maximal assessment (2.0) by 20% of every delayed week.

			Each student in the team is evaluated separately, according to the student's responses during the review of the assignment, according to student's activeness during the previous laboratories, according to code reviews and created code scope. Partially finished laboratory work or without following the given requirements is evaluated accordingly.
Mini quizzes during lectures	0-10	During the semester	During the lecture students might get asked random questions in an interactive way, which are not mandatory. Each answer can be rated from 0.01 to 0.05.
Exam in written form	50	Exam session	Exam can be taken only when the total amount of points collected during the semester is 3.0 or more and all laboratory assignments are completed. Maximum 5 points can be collected, which contribute to the 50% of the final score. The exam consists of open, semi-open and close-ended questions and tasks each of them is assessed between 0.1 and 2 points (according to the difficulties). Questions and tasks are formulated from topics set out in lectures. Exam is considered to be passed if at least 1.5 out of 5 points are collected.

Requirements for subject evaluation by external method	
Evaluation is possible externally:	Yes
The student must have met the requirements for taking the exam. Previously earned points for work during the semester are credited. The student only takes the exam.	

Author	**Publishing year**	**Title**	**Number or volume**	**Publisher or URL**
Required reading				
Andrew Troelsen	2020	Pro C# 8 with .NET Core	9th ed.	Apress
Jon Skeet	2019	C# in Depth	4th ed.	Manning Publications
Andy Hunt	2019	The Pragmatic Programmer: your journey to mastery, 20th Anniversary Edition	2nd ed.	Addison-Wesley Professional
Recommended reading				
Tiberiu Covaci, Rod Stephens, Vincent Varallo, Gerry O'Brien	2013	MCSD Certification Toolkit (Exam 70-483)		
Dan Clark	2013	Beginning C# Object-Oriented Programming	2nd ed.	Apress
Jack Purdum	2012	Beginning Object-Oriented Programming with C#		Wiley / Wrox
Andrew Hunt, David Thomas	1999	The pragmatic programmer: from journeyman to master	1st ed.	The Pragmatic Bookshelf
Titus Winters, Tom Manshreck, Hyrum Wright	2020	Software Engineering at Google: Lessons Learned from Programming Over Time	1st edition	O'Reilly Media, Inc.
James Shore	2007	The Art of Agile Development: Pragmatic Guide to Agile Software Development 1st Edition	-	O'Reilly Media, Inc.
Jake Knapp, John Zeratsky, Braden Kowitz	2016	Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days	1st ed,	Simon & Schuster
Roy Osherove	2013	The Art of Unit Testing: with examples in C#	2nd edition	Manning Publications