## Course unit description

| Course unit title | Code |
|---|---|
| **Object-Oriented Programming** | |

| Annotation |
|---|
| The subject is intended for the study of the paradigm and principles of use of object-oriented programming. Students are introduced to the concepts of abstract types and their realization and use. The basics of object-oriented programming are provided, and the formation of object-oriented program models is taught. In the course are analyzed class description tools, hiding of class internal structure, external interface description tools, feature inheritance in class families and their polymorphism, class compositions, critical situation control and template programming tools, linked list management classes. |

| Lecturer(s) | Department, Faculty |
|---|---|
| **Coordinating: lect dr. Donatas Dervinis**<br>**Other:** | Šiauliai Academy |

| Study cycle | Type of the course unit |
|---|---|
| First cycle studies | Compulsory |

| Mode of delivery | Semester or period when it is delivered | Language of instruction |
|---|---|---|
| Face-to-face | 2 semester | English |

| Requisites | |
|---|---|
| **Prerequisites: Procedural programming** | **Co-requisites (if relevant):** |

| Number of ECTS credits allocated | Student's workload (total) | Contact hours | Individual work |
|---|---|---|---|
| 5 | 133 | 56 | 77 |

| Purpose of the course unit: programme competences to be developed |
|---|
| The aim of the module is to acquaint students with the key concepts and principles of object-oriented programming; provide a basics to develop object-oriented systems.<br>***General competence:***<br>• Communication and collaboration.<br>• Continuous learning.<br>**Subject competences:**<br>• Knowledge and skills of conceptual foundations.<br>• Knowledge and skills of software development.<br>• Technological, methodological knowledge and skills, professional competence. |

| Learning outcomes of the course unit | Teaching and learning methods | Assessment methods |
|---|---|---|
| Will be able to understand the object-oriented (OO) programming paradigm. | Lectures, problem-based teaching, case studies, information search, reading literature, independent work, team work, learning from teammates, | Laboratory works and defense of results, written exam (open, semi-open and closed type questions and tasks). |

| | | |
|---|---|---|
| Will know the basic principles of programming and the basic constructions of object language | case studies, consultations, laboratory work. | |
| Will be able to apply OO methods in modeling real world phenomena.<br>Will be able to create classes and their hierarchies. | | |
| Understand the role of standard modeling language (UML) in OO analysis and design, read and create the most common UML diagrams.<br>Will be able to recognize and apply basic OO design templates using a variety of programming languages. | | |

| **Course content: breakdown of the topics** | **Contact hours** | | | | | | | **Individual work: time and assignments** | |
|---|---|---|---|---|---|---|---|---|---|
| | Lectures | Tutorials | Seminars | Workshops | Laboratory work | Internship/work placement | **Contact hours, total** | **Individual work** | **Assignments** |
| 1. Introduction, history, key concepts of object-oriented programming, features and differences of programming languages. | 1 | | | | 0 | | **1** | **2** | Independent reading for deeper knowledge.<br>Preparation for laboratory work. |
| 2. Keywords, data types, operators, phenomena, sentences, control structures. | 1 | | | | 2 | | **3** | **2** | |
| 3. Classes and objects, fields, methods, visibility control, encapsulation, and information concealment. | 2 | | | | 2 | | **4** | **4** | |
| 4. Classes and objects, object representation, statics, object life cycle, object creation, initiation, use, garbage collector. | 2 | | | | 4 | | **6** | **6** | |
| 5. Composition, inheritance, overlap of methods, dynamic coupling, polymorphism. | 1 | | | | 4 | | **5** | **6** | |
| 6. Interfaces, abstract methods and classes, non-overlapping methods and classes, fundamental design templates | 1 | | | | 2 | | **3** | **6** | |
| 7. Arrays, collections, inner classes, packages. | 2 | | | | 2 | | **4** | **4** | |
| 8. Object copying, design templates. | 1 | | | | 2 | | **3** | **4** | |
| 9. Exception handling, types of exceptions, standard exceptions. | 2 | | | | 4 | | **6** | **4** | |
| 10. Introduction to object-oriented analysis and design, UML language. | 2 | | | | | | **2** | **2** | |
| 11. Input and output flows, family of flow classes, serialization. | 1 | | | | 2 | | **3** | **4** | |
| 12. Multi-threading, synchronization, invariance of the state of an object. | 2 | | | | 2 | | **4** | **4** | |
| 13. Compiler Versions: An Overview of Advanced Techniques. | 2 | | | | | | **2** | **4** | |
| 14. Graphical user interface, design templates. | 2 | | | | 4 | | **6** | **4** | |
| 15. Tools and technologies. | 2 | | | | 2 | | **4** | **4** | |
| 16. Exam preparation and exam (written). | | | | | | | | **9** | Repeat of literature and preparation for the exam |
| **Total** | **24** | | | | **32** | | **56** | **77** | |

| Assessment strategy | Weight % | Deadline | Assessment criteria |
|---|---|---|---|
| Laboratory works | 60 | During the semester | During the laboratory work, the student must complete one project (create an application for a given real-world scenario). After defending the laboratory work (solving related problems and answering the given questions), the student can get from 0 to 10 points (the criteria depend on the specific laboratory work). Each week of delay reduces the maximum allowable score by 1 point, but by no more than 5 points after 5 weeks. In total, it is possible to score a maximum of 100 points, which corresponds to 60% of the final score. A minimum of 50 points is required to pass the exam.<br>Additional points are awarded earlier for the task (maximum 10% of the points received, 5% for each previous week).<br>Each student's individual contribution to the overall group work is assessed both during the assessment and during each exercise by a demonstration, as well as by assessing the scope and weight of the code reviews performed and the code generated.<br>In case of late payment every week, the maximum rating is reduced by 20% of the initial (1.5).<br>If the task is not completed, the scores are reduced proportionately. |
| Exam (written) | 40 | During the exam session | Up to 4 points can be scored during the exam, which corresponds to 40% of the final grade. The exam consists of three parts. In the first part, the student must provide answers to different questions of varying complexity (0-2 points). In the second part, the student must provide a practical solution to the given problem, which includes writing the code (0-1 points). In the third part, the student has<br>demonstrate an understanding of the given topic by writing a completed summary and providing explanatory examples (0-1 points). |

| Author | Publishing year | Title | Issue of a periodical or volume of a publication; pages | Publishing house or internet site |
|---|---|---|---|---|
| **Required reading** | | | | |
| John Sharp | 2018 | Microsoft Visual C# Step by Step, | | Microsoft |
| Joseph Albahari | 2021 | C# 9.0 in a Nutshell | | O'Reilly |
| Steve Prettyman | 2016 | Learn PHP 7: Object-Oriented Modular Programming using HTML5, CSS3, JavaScript, XML, JSON, and MySQL. | | Apress |
| **Recommended reading** | | | | |
| W3schools | | C# ir PHP OOP Tutorial | | https://www.w3schools.com/ |