



# PROGRAMMING 1. OBJECT ORIENTED PROGRAMMING AND DESIGN PATTERNS

## Syllabus

### Requisites of the Course

<b>Cycle of Higher Education</b>	<i>First cycle of higher education (Bachelor's degree)</i>
<b>Field of Study</b>	<i>F Information Technologies</i>
<b>Speciality</b>	<i>F2 Software engineering</i>
<b>Education Program</b>	<i>Software Engineering of Multimedia and Information Retrieval Systems</i>
<b>Type of Course</b>	<i>Normative</i>
<b>Mode of Studies</b>	<i>full-time</i>
<b>Year of studies, semester</b>	<i>2 year (3 semester)</i>
<b>ECTS workload</b>	<i>4 credits (ECTS). Time allocation: 54 hours for lectures, 18 hours for programming assignments, 78 hours for self-study.</i>
<b>Testing and assessment</b>	<i>Exam</i>
<b>Course Schedule</b>	<i>According to rozklad.kpi.ua</i>
<b>Language of Instruction</b>	<i>English</i>
<b>Course Instructors</b>	<i>Assitant, Artur Oleksii, PhD <a href="mailto:oleksii.artur@iit.kpi.ua">oleksii.artur@iit.kpi.ua</a></i>
<b>Access to the course</b>	<i>Google classroom</i>

### Outline of the Course

#### 1. Course description, goals, objectives, and learning outcomes

*The study of the Programming 1. Object Oriented Programming and Design Patterns course allows students acquire competencies necessary for solving practical problems related to the designing and development of object-oriented software.*

*The purpose of studying the Programming 1. Object Oriented Programming and Design Patterns course is to build capacity to choose and implement the principles of OOP as well as design pattern for efficient programming code development.*

*Studying the discipline “Programming. Part 1. Object-Oriented Programming and Design Patterns” enables students to acquire the competencies necessary to solve practical tasks in professional activities related to the design and development of object-oriented software.*

*The purpose of studying the discipline “Programming. Part 1. Object-Oriented Programming and Design Patterns” is to form students’ ability to analyze software product requirements and formulate the technical specification for program development, reasonably select and implement the necessary OOP principles in code, ensure minimization of intermodule dependencies in the developed software, and promote code reuse.*

*The subject of the discipline “Programming. Part 1. Object-Oriented Programming and Design Patterns” is the process of software development.*

*In the context of the educational program, studying the discipline “Programming. Part 1. Object-Oriented Programming and Design Patterns” develops students’ professional competencies (PC), necessary to solve practical problems of professional activity related to the development of object-oriented software:*

**PC01** *Ability to identify, classify, and formulate software requirements.*

**PC02** *Ability to participate in software design, including modeling (formal description) of its structure, behavior, and functioning processes.*

**PC03** *Ability to develop architectures, modules, and components of software systems.*

**PC05** *Ability to comply with specifications, standards, rules, and recommendations in the professional field when implementing life-cycle processes.*

**PC07** *Knowledge of information data models; ability to create software for data storage, retrieval, and processing.*

**PC08** *Ability to apply fundamental and interdisciplinary knowledge to successfully solve software engineering tasks.*

**PC10** *Ability to accumulate, process, and systematize professional knowledge regarding software development and maintenance, and to recognize the importance of lifelong learning.*

**PC13** *Ability to reasonably select and master tools for software development and maintenance.*

**PC14** *Ability for algorithmic and logical thinking.*

**PC16** *Ability to develop software for information retrieval systems.*

**PC17** *Ability to develop software for multimedia and multiservice systems.*

*Learning Outcomes (LO):*

**LO01** *Analyze, purposefully search for, and select the information and reference resources and knowledge necessary to solve professional tasks, considering modern achievements in science and technology.*

**LO03** *Know the main processes, phases, and iterations of the software life cycle.*

**LO04** *Know and apply professional standards and other regulatory documents in the field of software engineering.*

**LO06** *Ability to choose and apply an appropriate methodology for software development depending on the task.*

**LO07** *Know and apply in practice the fundamental concepts, paradigms, and basic principles of programming languages, tools, and computational means of software engineering.*

**LO08** *Ability to design human-computer interfaces.*

**LO13** *Know and apply methods of algorithm development, software construction, and data and knowledge structures.*

**LO15** *Ability to reasonably choose programming languages and development technologies for solving tasks related to software development and maintenance.*

**LO18** *Know and be able to apply information technologies for data processing, storage, and transmission.*

**LO23** *Ability to document and present software development results.*

**LO30** *Ability to apply programming technologies to develop software for multimedia and information retrieval systems.*

**L031** Know and be able to apply principles of search engine construction, methods and algorithms for executing various types of information search, and criteria for evaluating the efficiency of information retrieval.

**L032** Know the main models of representing textual and multimedia information and the methods of their preprocessing for application in the design of information retrieval systems.

**L033** Know and be able to use software resources and libraries for processing textual information and multimedia data in information retrieval systems. The subject of the Programming 1. Object Oriented Programming and Design Patterns course is the process of software development.

## **2. Prerequisites and post-requisites of the course (the place of the course in the scheme of studies in accordance with curriculum)**

The Programming 1. Object Oriented Programming and Design Patterns course is a normative discipline and students do not need any specific initial knowledge for its study.

Theoretical knowledge and practical skills acquired in the Programming 1. Object Oriented Programming and Design Patterns course provide the necessary background for studying other disciplines in Bachelor and Master programs of F2 Software Engineering specialty.

## **3. Content of the course**

Topic 1. Introduction to C# and .NET

Topic 2. Fundamental principles of OOP

Topic 3. Software development with OOP

Topic 4. Design patterns

Exam

## **4. Coursebooks and teaching resources**

### **Main literature:**

1. Teaching materials for the course “Programming 1. Object oriented programming and design patterns.”

Read the sections on these topics: Introduction to .NET Platform, Basics of C# programming; Classes and objects; Encapsulation; Inheritance and Polymorphism in C#; Abstraction in C#; Generics and collections; Exception handling; Delegates, Events, and Lambdas; Additional OOP features in C#; Structural design patterns; Creational design patterns; Behavioral design patterns; The materials are available in Google Classroom. Access is provided to registered students. Use to master theoretical knowledge and practical skills in the discipline.

2. Professional C# and .NET / Christian Nagel. – Apress, 2021. – 823 p.

3. C# 12 in a Nutshell / Joseph Albahari, Ben Albahari. – O’Reilly Media, 2024. – 1080 p.

4. Pro C# 10 with .NET 6 / Andrew Troelsen, Philip Japikse. – Apress, 2022. – 1370 p.

### **Additional literature:**

5. C# Corner – Abstract Classes and Interfaces in C#. URL: <https://www.c-sharpcorner.com/UploadFile/8ef97c/abstract-classes-and-interfaces-in-C-Sharp>

6. Tutorials point. C# Tutorial. URL: <https://www.tutorialspoint.com/csharp/index.htm>

7. Farahmandian, Vahid. .NET 7 Design Patterns In-Depth: Enhance code efficiency and maintainability with .NET Design Patterns (English Edition). BPB Publications, 2023.

8. C# Design Patterns. URL: <https://www.dofactory.com/net/design-patterns>

**Educational content**

**5. Methodology**

No	Type of a class	Materials for self-studying
<i>Topic 1. Introduction to C# and .NET</i>		
1.	<i>Lecture 1. Introduction to .NET Platform</i>	<i>.NET – modern platform overview (LTS vs STS, release cycle). .NET runtime &amp; libraries. ASP.NET Core Cross-platform workloads (Console, Web, MAUI, Blazor, Cloud, IoT). Target framework and compatibility (e.g., net9.0). Unified execution model(managed code, JIT, GC). Release evolution</i>
2.	<i>Lecture 2. Basics of C# programming. Part 1</i>	<i>Program structure. Variables and constants. Literals. Data types. Console I/O (input and output). Arithmetic operations. Assignment operations. Base data type transformations</i>
3.	<i>Lecture 3. Basics of C# programming. Part 2</i>	<i>Conditionals - if / else and ternary operation. Cycles. Switch design. Enum enumerations. Arrays.</i>
4.	<i>Lecture 4. Basics of C# programming. Part 3</i>	<i>Array tasks. Methods. Method parameters. Return value and return statement. Passing parameters (by reference, by value, output parameters, params keyword). Recursive functions. Local functions</i>
5.	<i>Programming assignment 1. Console application in C#</i>	<i>Task: Create console application to solve the tasks using arrays, loops and methods</i>
<i>Topic 2. Fundamental principles of OOP</i>		
6.	<i>Lecture 5. Classes and objects</i>	<i>Classes and Objects. Constructors, initializers. Scope (context) of variables. Namespaces.</i>
7.	<i>Lecture 6. Encapsulation</i>	<i>Access modifiers. Properties. Read-only and write-only properties,</i>
8.	<i>Programming assignment 2. Implementation of encapsulation in a C# application</i>	<i>Task: Create software to solve the tasks at hand using classes and encapsulation.</i>
9.	<i>Lecture 7. Inheritance and Polymorphism in C#</i>	<i>Inheritance. Type Conversion. Virtual methods and properties. Hiding methods and properties. The</i>

		<i>difference between overriding and hiding methods</i>
10.	<i>Programming assignment 3. Implementation of inheritance and polymorphism in a C# application</i>	<i>Task: Create software to solve the tasks at hand using encapsulation, inheritance and polymorphism.</i>
11.	<i>Lecture 8. Abstraction in C#</i>	<i>Abstract classes and interfaces</i>
12.	<i>Programming assignment 4. Implementation of abstraction in a C# application</i>	<i>Task: Create software to solve the tasks at hand using all learned OOP paradigms.</i>
<i>Topic 3. Software development with OOP</i>		
13.	<i>Lecture 9. Generics</i>	<i>Generalized types. Limitations of generalizations. Inheriting generalized types.</i>
14.	<i>Programming assignment 5. Development of a C# application</i>	<i>Task: Create software to solve the tasks at hand using generalized types.</i>
15.	<i>Lecture 10. Exception handling</i>	<i>The construction try catch finally. Catch block and exception filters. Types of exceptions. Exception class. Throwing an exception and the throw statement. Creating Exception Classes.</i>
16.	<i>Programming assignment 6. Development of a C# application</i>	<i>Task: Create software to solve the tasks at hand based on exception handling.</i>
17.	<i>Lecture 11. Additional OOP features in C#. Part 1</i>	<i>Operator overloading, Type conversion operators, Indexers, Reference variables and reference returns, Partial classes and methods, Extension methods</i>
18.	<i>Lecture 12. Additional OOP features in C#. Part 2</i>	<i>Delegates, Anonymous Methods and Lambda Expressions, Generic Delegates, Events, Covariance and Contravariance in Delegates</i>
19.	<i>Programming assignment 7. Development of a C# application using advances features of OOP</i>	<i>Task: Create software to solve the tasks at hand based on a material of previous lectures.</i>
<i>Topic 4. Design patterns</i>		
20.	<i>Lecture 13. Structural design patterns</i>	<i>Structural patterns: Adapter; Bridge; Composite; Decorator; Facade; Flyweight; Proxy.</i>
21.	<i>Lecture 14. Creational design patterns</i>	<i>Creational patterns: Singleton; Factory Method; Abstract Factory; Builder; Prototype; Object Pool.</i>

22.	<i>Lecture 15. Behavioral design patterns</i>	<i>Behavioral patterns: Chain of Responsibility; Command Interpreter; Iterator; Mediator; Memento; Observer; State; Strategy; Template Method; Visitor.</i>
23.	<i>Programming assignment 8. Implementation of design patterns in a C# application</i>	<i>Task: Create software to solve the tasks at hand using design patterns</i>
<i>Midterm test</i>		

## 6. Self-study

<i>No</i>	<i>Topic for self-studying</i>	<i>Hours</i>	<i>Literature</i>
1.	<i>Preparation to a lecture 1</i>	2	1, 4
2.	<i>Preparation to a lecture 2</i>	2	1, 3, 5, 6
3.	<i>Preparation to a lecture 3</i>	2	1, 3, 5, 6
4.	<i>Preparation to a lecture 4</i>	2	1, 3, 5, 6
5.	<i>Preparation to a programming assignment 1</i>	3.5	1, 2, 3, 5, 6
6.	<i>Preparation to a lecture 5</i>	2	1, 4-6
7.	<i>Preparation to a lecture 6</i>	2	1, 4-6
8.	<i>Preparation to a programming assignment 2</i>	3.5	1,4-6
9.	<i>Preparation to a lecture 7</i>	2	1, 4-6
10.	<i>Preparation to a programming assignment 3</i>	3.5	1, 4-6
11.	<i>Preparation to a lecture 8</i>	2	1, 4-6
12.	<i>Preparation to a programming assignment 4</i>	3.5	1, 4-6
13.	<i>Preparation to a lecture 9</i>	2	1, 3, 5, 6
14.	<i>Preparation to a programming assignment 5</i>	3.5	1, 3, 5, 6
15.	<i>Preparation to a lecture 10</i>	2	1, 3, 6
16.	<i>Preparation to a programming assignment 6</i>	3.5	1, 3, 6
17.	<i>Preparation to a lecture 11</i>	2	1, 4
18.	<i>Preparation to a lecture 12</i>	2	1, 4
19.	<i>Preparation to a programming assignment 7</i>	3.5	1, 3, 4, 6
20.	<i>Preparation to a lecture 13</i>	2	1, 7, 8
21.	<i>Preparation to a lecture 14</i>	2	1, 7, 8
22.	<i>Preparation to a lecture 15</i>	2	1, 7, 8
23.	<i>Preparation to a programming assignment 8</i>	3.5	1, 7, 8
24.	<i>Preparation to a midterm test</i>	10	1-8

25.	Preparation to an exam	10	1-8

## Policy and Assessment

### 7. Course policy

The forms of organization of the educational process, types of educational activities and assessment of learning outcomes are regulated by the Regulations on the Organization of the Educational Process at the National Technical University of Ukraine 'Igor Sikorsky Kyiv Polytechnic Institute' ([https://document.kpi.ua/files/2020\\_7-124.pdf](https://document.kpi.ua/files/2020_7-124.pdf)).

**Class attendance policy.** Attendance at lectures is mandatory. Attendance at laboratory classes may be sporadic and, if necessary, for the defense of laboratory work. If a student is unable to attend classes, they are still responsible for studying the theoretical material and completing practical assignments.

**Policy on ethical standards in class:** discipline; observance of subordination; honesty; responsibility; respect for those present; turning off phones. The standards of ethical behavior for students and academic staff are governed by the provisions set out in Section 2 of the Code of Honor of the National Technical University of Ukraine 'Igor Sikorsky KPI' (<https://kpi.ua/code>).

**Policy on assessment of learning outcomes:** each grade is awarded in accordance with criteria developed by the teacher and announced to students in advance. If a student fails to complete all types of classes (laboratory work, modular tests) specified in the curriculum, he or she will not be allowed to take the exam; missed classes during which tests were conducted must be made up (during consultations or at a time agreed upon with the instructor on an individual basis).

**Academic conduct and integrity policy.** The policy and principles of academic integrity are governed by the rules set out in Section 3 of the Code of Honor of the National Technical University of Ukraine 'Igor Sikorsky KPI' (<https://kpi.ua/code>); Regulations on the organization of the educational process ([https://document.kpi.ua/files/2020\\_7-124.pdf](https://document.kpi.ua/files/2020_7-124.pdf)), Regulations on the system for preventing academic plagiarism (<https://osvita.kpi.ua/node/47>), Regulations on the Commission on Ethics and Academic Integrity (<https://osvita.kpi.ua/node/171>). Plagiarism and other forms of violation of the principles of academic integrity are unacceptable. All laboratory work must be performed independently by the student using open sources of information and acquired knowledge and skills.

**Policy on appealing the results of assessment measures.** According to the 'Regulations on Conflict Resolution at Igor Sikorsky Kyiv Polytechnic Institute' (<https://osvita.kpi.ua/node/169>), students have the right to appeal the results of assessment measures, explaining which criteria they disagree with in accordance with the assessment. Students may raise any issue related to the assessment procedure and expect it to be considered in accordance with pre-defined procedures.

**Policy on awarding bonus and penalty points.** In accordance with the 'Regulations on the system of assessment of learning outcomes at Igor Sikorsky KPI' (<https://osvita.kpi.ua/node/37>), bonus and penalty points are not included in the main RSO scale, and their sum cannot exceed 10% of the starting scale (10 points).

Incentive points are awarded for a creative approach to laboratory work (the maximum number of points for all work is 4 points).

Penalty points are awarded for violations of academic integrity principles, in particular plagiarism in laboratory work (the program code does not correspond to the task option, the program code is identical among different works), non-independent performance of modular control work (-5 points for each attempt to violate academic integrity principles).

## 8. Monitoring and grading policy

In the first lecture, the students are being acquainted with the grading policy which is based on Regulations on the System of Learning Outcomes Assessment ([https://document.kpi.ua/files/2020\\_1-273.pdf](https://document.kpi.ua/files/2020_1-273.pdf)).

The student's rating in the course consists of points that they receive for programming assignment ( $R_1$ ), a midterm test ( $R_2$ ) and an exam ( $R_3$ ).

$$R_S = R_1 + R_2 + R_3 = 64 + 16 + 20 = 100 \text{ points}$$

According to the university regulations on the monitoring of students' academic progress ([https://kpi.ua/document\\_control](https://kpi.ua/document_control)), there are two assessment weeks, usually during 7th/8th and 14th/15th week of the semester, when students take the Progress and Module tests respectively, to check their progress against the criteria of the course assessment policy.

The table of compliance between overall points and the final grade:

Points	Grade
95-100	Excellent
85-94	Very good
75-84	Good
64-74	Satisfactory
60-64	Fair
Less than 60	Unsatisfactory
Course requirements are not met	Not Graded

## 9. Additional information about the course

### Syllabus of the course

Is designed by PhD, Assistant, Artur Oleksii

Approved by the Department of PZKS (Protocol No. 3 dated 29.09.2025)

Approved by the Methodological Commission of the Faculty of Applied Mathematics (Protocol No. 2 dated 16.10.2025)