МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ЗАТВЕРДЖЕНО
на засіданні кафедри
інформаційних систем.
Протокол № 7 від 19.01.2026 р.

ПОГОДЖЕНО
Проректор з навчально-методичної роботи

_____ Карина НЕМАШКАЛО

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

робоча програма навчальної дисципліни (РПНД)

Галузь знань	**F "Інформаційні технології"**
Спеціальність	**F2 "Інженерія програмного забезпечення"**
Освітній рівень	**перший (бакалаврський)**
Освітня програма	**"Інженерія програмного забезпечення"**

Статус дисципліни	**обов'язкова**
Мова викладання, навчання та оцінювання	**англійська**

Розробник:
к.т.н., с.н.с. _____ Юрій ПАРФЬОНОВ

к.т.н., доцент _____ Дмитро БОНДАРЕНКО

Завідувач кафедри
інформаційних систем _____ Дмитро БОНДАРЕНКО

Гарант програми _____ Олег ФРОЛОВ

Харків
2026

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
SIMON KUZNETS KHARKIV NATIONAL UNIVERSITY OF ECONOMICS

APPROVED
at the meeting of the department
information systems
Protocol № 7 of 19.01.2026

AGREED
Vice-rector for educational and methodical work

_____ Karina NEMASHKALO

OBJECT-ORIENTED PROGRAMMING

Program of the course

Field of knowledge	**F "Information technologies"**
Specialty	**F2 "Software Engineering"**
Study cycle	**first (bachelor)**
Study programme	**"Software Engineering"**

Course status	**mandatory**
Language	**English**

Developer:
PhD (Information technology),
Senior Research Assistant _____ Yurii PARFONOV

PhD (Information technology),
Associate Professor _____ Dmytro BONDARENKO

Head of Information systems
Department _____ Dmytro BONDARENKO

Head of Study Programme _____ Oleg FROLOV

Kharkiv
2026

INTRODUCTION

Nowadays, the most common method of dealing with software complexity is an object-oriented approach to its development. This requires the relevant specialists to have a clear understanding of the concepts of object-oriented programming, which makes it possible to use them in practice in the process of developing applications in any programming language.

"Object-Oriented Programming" is a mandatory course that is studied in accordance with the curriculum for the bachelor's degree in specialty F2 "Software Engineering". A necessary element of successful mastering of the course's educational material is the independent work of students with technical literature and modern program development environments.

The aim of the course "Object-Oriented Programming" is to develop competencies in using the basic elements of the object-oriented approach and modern programming languages necessary to develop appropriate software applications.

The objectives of the course are the formation of object-oriented thinking; mastering of the basic concepts of object-oriented design, analysis, and programming; mastering of object-oriented programming languages; development of skills in the development of object-oriented software systems; improvement of skills in using programming environments.

The object of the course is the basic elements of the object-oriented approach to software development.

The subject of the course is the principles of object-oriented programming and the basics of object-oriented technology.

The learning outcomes and competencies formed by the course are defined in Table 1.

Table 1

Learning outcomes and competencies formed by the course

Learning outcomes	Competencies
LO5	SC2, SC3, SC8, SC14, IC
LO7	SC10
LO8	SC1, SC10
LO 13	SC2, SC3, SC7
LO 15	SK10, SK11, SK13
LO 17	GC7

where, LO5. Know and apply relevant mathematical concepts, methods of domain, system and object-oriented analysis and mathematical modeling for software development.

LO7. To know and apply in practice the fundamental concepts, paradigms, and basic principles of functioning of language, technological and computing tools of software engineering.

LO8. Be able to develop a human-machine interface.

LO13. Know and apply methods of developing algorithms, designing software and data and knowledge structures.

LO15. Motivated to choose programming languages and development technologies to solve the problems of creating and maintaining software.

LO17. Be able to apply the methods of component-based software development.

SC1. Ability to identify, classify and formulate software requirements.

SC2. Ability to participate in the design of software, including modeling (formal description) of its structure, behavior, and processes of functioning.

SC3. Ability to develop architectures, modules, and components of software systems.

SC7. Knowledge of data information models, ability to create software for data storage, extraction, and processing.

SC8. The ability to apply fundamental and interdisciplinary knowledge to successfully solve software engineering problems.

SC10. Ability to accumulate, process and systematize professional knowledge of software development and maintenance and recognize the importance of lifelong learning.

SC11. Ability to implement phases and iterations of the life cycle of software systems and information technologies based on appropriate software development models and approaches.

SC13. Ability to reasonably choose and master tools for software development and maintenance.

SC14. Ability to think algorithmically and logically.

GC7. Ability to work in a team.

IC. Ability to solve complex specialized tasks or practical software engineering problems characterized by complexity and uncertainty of conditions, using information technology theories and methods.

COURSE CONTENT

Content module 1: Fundamentals of the object-oriented paradigm

Topic 1: .NET and Java SE basics

1.1. Programming platforms .NET and Java SE: architecture, compilation, and execution of programs, type system, standard class libraries, program development tools.

Topic 2. Fundamentals of an object-oriented programming language

2.1. General information about C# and Java languages: alphabet, data types, operations, operators, program structure, basics of using standard libraries of .NET and Java SE classes. Formatting strings.

2.2. One-dimensional and multidimensional arrays in C# and Java: creating, initializing, processing, supporting arrays in standard .NET and Java SE libraries.

2.3. Methods in C# and Java: definitions, mechanisms for passing parameters, using an array as a parameter, returning an array from a method, calling a method.

2.4. Overloading class methods. Principles of overloading operations.

Topic 3. Concepts of object-oriented analysis, design, and programming

3.1 Object-oriented decomposition. Principles of object-oriented approach: abstraction, encapsulation, hierarchy, polymorphism.

3.2. The concept of an object. Characteristics of the object. The concept of class. The relationship between a class and its object.

3.3 Object-oriented analysis and its purpose. The main types of requirements for a software system. Object-oriented design. Object-oriented programming.

3.4. Software modeling. Basic concepts of visual modeling. Fundamentals of the UML language. UML diagrams. UML class diagrams. Relationships in the class diagram. CASE tools.

Content module 2: OOP technology

Topic 4. Data abstraction and encapsulation

4.1 Classes and structures. Elements of the class. Features of using static elements. Access to class elements, access modifiers.

4.2. The concept of creating, initializing, and using objects. Reference this.

4.3. The life cycle of objects. The sequence of creating and initializing an object. Constructors. The default constructor. Basic properties of constructors. Overloading of constructors. Freeing up memory. The "garbage collection" system.

Topic 5. Reusing the code

5.1. The concept of association. Relationships of composition and aggregation as types of association. Implementation of composition and aggregation in C# and Java.

5.2. The inheritance relationship. Implementation of the inheritance in C# and Java. Initialization of the object of base class. Using constructors during inheritance. Options for using inheritance. Overriding methods.

5.3. Early and late binding. Virtual methods. Implementation of the principle of polymorphism in C# and Java. String representation of an object. Abstract classes and methods. Implementation of polymorphic behavior based on an abstract class. Interfaces. Implementation of polymorphic behavior based on an interface.

Topic 6: Basics of using Git version control system

6.1 Basic concepts of version control systems. Local, centralized, and distributed version control systems. The basics of Git. Using local and remote repositories.

Topic 7. Principles of object-oriented design (SOLID)

7.1. Overview of SOLID. The single responsibility principle. The open-close principle. The dependency inversion principle. Barbara Liskov's principle of substitution. The interface isolation principle.

Topic 8: Introduction to design patterns

8.1. General information about design patterns. Advantages and disadvantages of using design patterns. Elements of a design patterns.

8.2. Classification of GoF design patterns. Application of the main GoF design

patterns.

8.3. Overview of the GRASP patterns.

Topic 9: Class libraries

9.1. Libraries and their use. Static and dynamic libraries.

9.2. Developing libraries on the Java SE platform. DLL-libraries. Development of DLL-libraries on the .NET platform.

Content module 3: Exception handling and Class libraries

Topic 10. Handling exceptional situations

10.1. Types of errors in programs. Problems of the traditional approach to error handling.

10.2. Mechanism for handling exceptions. Exception classes of standard libraries .NET and Java SE. Syntax of exception handling.

Topic 11. Standard class libraries

11.1. General information about collections. The basic data structures of the standard collection libraries of .NET and Java SE. Typed collections.

11.2. Using lambda expressions

11.3. Using LINQ and Stream API

11.4. Features of the implementation of the string data type on the .NET and Java SE platforms. Classes of standard .NET and Java SE libraries for representing strings and features of their use.

11.5. Purpose and use of regular expressions. Support for regular expressions on the .NET and Java SE platforms. Special characters used in regular expressions.

11.6. Sources and consumers of data. General information about data input and output streams. Algorithms of data input and output. The main classes of standard .NET and Java SE libraries to support data input and output.

11.7. Saving and restoring the state of objects on the .NET and Java SE platforms. Serialization and deserialization. "Graph" of objects during serialization. Creating classes whose objects can be serialized. Serialization and deserialization processes. Serialization formats.

11.8. General information about object-relational data mapping. The concept of ORM frameworks for .NET and Java SE platforms. Introduction to models. The class of the model. Setting up models. Database migrations.

11.9. Distributed software systems. Software clients and servers. General information about TCP sockets. Basics of using TCP sockets on .NET and Java SE platforms.

The list of laboratory studies in the course is given in Table 2.

Table 2

The list of laboratory studies

Name of the topic	Content
Topic 1, Topic 2. Laboratory session 1	Basics of using C# and Java programming languages
Topic 3. Laboratory session 2	Designing classes using the UML language
Topic 4. Laboratory session 3	Developing applications using basic elements of OOP
Topic 5. Laboratory session 4	Application of inheritance and polymorphism
Topic 6. Laboratory session 5	Using Git version control system
Topic 8: Laboratory session 6	Using design patterns
Topic 11. Laboratory session 7	Using collections
Topic 11. Laboratory session 8	Using lambda expressions, LINQ and Stream API
Topic 11. Laboratory session 9	Using regular expressions
Topic 11. Laboratory session 10	Using data input and output
Topic 11. Laboratory session 11	Using object-relational data mapping
Topic 11. Laboratory session 12	Using TCP sockets

The list of self-studies in the course is given in Table 3.

Table 3

List of self-studies

Name of the topic	Content
Topics 1 - 11	Study of lecture material
Topics 1 - 11	Preparing for laboratory classes
Topics 1 - 11	Preparing for the final exam

The number of hours of lectures, laboratory studies and hours of self-study is given in the technological card of the course.

TEACHING METHODS

In the process of teaching the course, to acquire certain learning outcomes, and activate the educational process, it is envisaged to use such teaching methods as:

Verbal (lecture (Topics 2, 4, 5, 6, 8 - 11), problem-based lectures (Topics 1, 3, 7)).

Visual (demonstration (Topics 1-11)).

Practical (laboratory studies, interactive distance learning, coaching, presentation, role-playing games (Topics 1-11); case studies (Topics 6, 11)).

FORMS AND METHODS OF ASSESSMENT

The University uses a 100-point cumulative system for assessing the learning outcomes of students.

Current control is carried out during lectures, practical, laboratory and seminar classes and is aimed at checking the level of readiness of the student to perform a specific job and is evaluated by the amount of points scored:

– for courses with a form of semester control as grading: maximum amount is 100 points; minimum amount required is 60 points.

– for courses with a form of semester control as an exam: maximum amount is 60 points; minimum amount required is 35 points.

The final control includes current control and assessment of the student .

The final control includes current control and an exam .

Semester control is carried out in the form of a semester exam or grading.

The final grade in the course is determined:

– for disciplines with a form of grading, the final grade is the amount of all points received during the current control.

– for disciplines with a form of exam, the final grade is the amount of all points received during the current control and the exam grade.

During the teaching of the course, the following control measures are used:

Current control: 1 semester - defense of laboratory work (83 points), final control work (17 points). 2nd semester - defense of laboratory work (50 points), control work (10 points).

Semester control: Grading / Grading including Exam (40 points).

More detailed information on the assessment system is provided in technological card of the course.

An example of an exam card and assessment criteria.

Example of an exam card

Task 1.

Develop a Point3D class (a point in three-dimensional space) with fields X, Y, Z, which are the coordinates of the point in the Cartesian system. This class should also have a method for calculating the distance from the origin to the point using the formula $(X^2 + Y^2 + Z)^{21/2}$, and **set** and **get** properties for setting the point coordinates and getting their current values, respectively.

Develop another class that contains the Main() method in which it is executed:

1. Entering the coordinates of a single point from the console, creating the corresponding object of the Point3D class and initializing it using the **set** properties.
2. Display on the console the coordinates of this point and the distance from it to the origin using the created object of the Point3D class and the **get** property.

Task 2.

Create three classes: Group, Student, MainClass (contains the main method of the program). The Group and Student classes must be linked by an **aggregation** relationship, in which a **group consists of three students**.

The Student class at least has the LastName field - the student's last name. The Group class must at least contain the AddStudent method with parameters, designed to add a student to the group, and the PrintList method for displaying the names of the group's students on the console.

The main method of the program must at least execute:

1. Creating a single object of the Group class.
2. Entering initial data about students from the console and calling the AddStudent method.
3. Calling the PrintList method.

Assessment criteria

The exam card consists of two heuristic tasks. Each task is rated from 0 to 6 conditional points according to the following scale:

6 points	The task has been completed in full. The program works correctly. The interface and source code of the program meet the requirements specified in the assignment.
5 points	The task is completed. The program works correctly, but one of its features is implemented in violation of the requirements specified in the assignment.
4 points	The task is mostly completed. The program works, but two of its features are implemented in violation of the requirements specified in the assignment.
3 points	The task is completed, but not in full. The program works, but one of the functional requirements specified in the assignment is not implemented, or three of the program's features are implemented in violation of the requirements specified in the assignment.
2 points	The task is not completed. The program runs, but two of the functional requirements specified in the task are not implemented, or four of the program's features are implemented in violation of the requirements specified in the task.
1 point	The program is launched and partially meets the task formulation, but more than two of the functional requirements specified in the task are not implemented, or more than four of the program's features are implemented in violation of the requirements specified in the task.

	The program does not start or crashes, but there is a program code developed by the student that meets the task formulation.
0 points	There is no program. The program does not contain program code developed by the student. The program does not meet the task formulation. The program has obvious technical signs of not being developed independently.

Failure to comply with or significant violation any of the general requirements for the program reduces the number of conditional points for the task by one point.

General requirements for programs

1. *The user interface of the console program developed in accordance with the task formulation must consist of text messages in English related to the input and output of basic and auxiliary data. The use of transliteration is prohibited.*
2. *Adherence to the principle of encapsulation regarding access levels to fields, properties, and methods of classes.*
3. *The source code of each program class should be in a separate file.*

The final grade for the exam is determined by the sum of the conditional points gained for the two tasks:

The amount of conditional points gained	Final grade for the exam on a 40-point scale
12	40
11	38 - 39
10	36 - 37
9	34 - 35
8	32 - 33
7	30 - 31
6	28 - 29
5	26 - 27
4	24 - 25
3	16 - 23
2	8 - 15
1	1 - 7
0	0

RECOMMENDED LITERATURE

Main

1. Booch G. Object-Oriented Analysis and Design with Applications / G.Booch, R.A.Maksimchuk et. al: Addison-Wesley, 2019. - 717 p.

2. Troelsen A.Pro C# 9 with .NET 5 : Foundational Principles and Practices in Programming / A. Troelsen, P. Japikse - Berkly: Apress, 2021. - 1353 p.

Additional

3. Lewis J. Java Foundations. Introduction to Program Design and Data Structures / J. Lewis, P. DePasquale, J. Chase - Lancing: Pearson, 2020. - 1104 p.

Information resources

4. What are UML diagrams, and how can you use them. [Electronic resource]: https://miro.com/blog/uml-diagram/.

5. C# documentation [Electronic resource]: https://docs.microsoft.com/en-us/dotnet/csharp/.

6. C# Tutorial [Electronic resource]: https://www.tutorialspoint.com/csharp/index.htm.

7. Java Tutorial [Electronic resource]. - Access mode: https://www.w3schools.com/java/default.asp.

8. JDK 20 Documentation [Electronic resource]: https://docs.oracle.com/en/java/javase/20/.

9. Щербаков, О. В. Основи об'єктно-орієнтованого програмування : навч. посіб. / О. В. Щербаков, Ю. Е. Парфьонов, В. М. Федорченко - Харків : ХНЕУ ім. С. Кузнеця, 2019. - 236 с. [Електронний ресурс]. - Режим доступу: http://repository.hneu.edu.ua/handle/123456789/23847

10. Personal learning system "Object-oriented programming (F2, Fall semester)" [Electronic resource]. - Access mode: https://pns.hneu.edu.ua/course/view.php?id=9125.

11. Personal learning system "Object-oriented programming (F2, Spring semester)" [Electronic resource]. - Access mode: https://pns.hneu.edu.ua/course/view.php?id=7819.