**KHMELNYTSKYI NATIONAL UNIVERSITY**

APPROVED
Dean of IT Faculty
_____ T. HOVORUSHCHENKO
"01" 09 2025.

# WORKING PROGRAMME OF THE EDUCATIONAL COMPONENT
## *Object-oriented programming*

**Field of Study:** F – Information Technology
**Specialty:** F2 – Software Engineering
**Level of Higher Education:** First (Bachelor's) Level
**Educational and Professional Programme:** Software Engineering
**Course Load:** 21 ECTS credits **Course Code:** CPT.04
**Language of Instruction:** English
**Status of the Educational Component:** Compulsory (Professional Training)
**Faculty:** Faculty of Information Technology
**Department:** Department of Software Engineering

| Form of Study | Year | Semester | Total Credits | | Number of hours | | | | | | Course project | Coursework | Semester control form | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Contact Hours | | | | | Independent Work (incl. Individual Tasks) | | | | |
| | | | ECTS credits | hours | Total | Lectures | Laboratory works | Practical classes | Seminar classes | | | | pass/ fail test | Exam |
| D | 1 | 2 | 4 | 120 | 50 | 16 | 34 | | | 70 | | | + | |
| D | 2 | 1 | 7 | 210 | 82 | 32 | 50 | | | 128 | | | | + |

The working programme is based on the Educational and Professional Programme "Software Engineering" within the specialty F2 "Software Engineering".

Program's author _____ DSc, Prof. V.V. Martynyuk

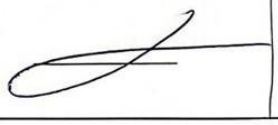
Approved at the meeting of the Department of Software Engineering
Minutes No. 1 dated August 28, 2025
Head of the Department _____ L.P. Bedratyuk

The working programme was reviewed and approved by the Academic Counci of the Faculty of Information Technology

Chair of the Academic Council _____ Tetiana HOVORUSHCHENKO

Khmelnytskyi 2025

**LETTER OF APPROVAL**

| Position | Department Name | Signature | First Name,<br>LAST NAME |
|---|---|---|---|
| Head of *D*epartment<br>DSc, Prof. | Software Engineering |  | Leonid BEDRATIUK |
| Programme Guarantor<br>*DS*c, Prof. | Software Engineering |  | Leonid BEDRATIUK |

# OBJECT-ORIENTED PROGRAMMING

| | |
|---|---|
| **Type of Educational Component** | Compulsory |
| **Level of Higher Education** | First (Bachelor's) Level |
| **Language of Instruction** | English |
| **Semester** | Second, Third |
| **Number of ECTS Credits Assigned** | 11 |
| **Forms of Study the Course is Designed For** | Full-time |

**Learning Outcomes.** Upon successful completion of the course, the student should be able to: *understand* the general structure of a C# program and apply syntax rules, namespaces, and assemblies correctly; *work* with files and streams, performing text and binary input/output operations using standard C# libraries; *design* and implement classes, define class members (fields, methods, properties), and apply encapsulation principles; *apply* inheritance and polymorphism to extend and reuse code, override methods, and implement abstract and virtual members; *use* delegates, anonymous methods, and lambda expressions to encapsulate behavior and implement functional-style programming in C#; *create* and manage events and event handlers, applying the publish-subscribe pattern for communication between objects; *define* and implement interfaces to enforce contracts and support multiple inheritance of behavior; *use* indexers to provide array-like access to objects and collections; *apply* generic templates to create type-safe and reusable data structures and methods; *implement* iterators with yield for efficient traversal of collections; *work* with LINQ (Language Integrated Query) to query and transform collections and databases, including advanced LINQ operators; *understand* the basics of Entity Framework Core, create models, and perform CRUD operations through object-relational mapping (ORM); *translate* SQL queries into Entity Framework syntax and apply LINQ-to-Entities; *use* navigation properties, and apply lazy and eager loading to manage relationships between entities effectively; *perform* complex joins, tagging, transactions, and debugging in Entity Framework to build reliable database applications; *model* software systems using use-case and behavioral modeling, applying UML and diagrams to design object-oriented solutions; *manage* dependencies and ensure modularity, applying principles of clean architecture and design patterns; *develop* multi-threaded and asynchronous programs, ensuring thread safety and responsiveness using async/await patterns; *implement* graphical user interfaces, integrating event-driven programming concepts; *debug*, test, and document programs according to good OOP practices, and use IDE tools such as Visual Studio for compilation, debugging, and profiling.

**Course Content.** General structure of C# programs. Files and streams. Classes, class members, and encapsulation. Inheritance and polymorphism. Delegates, anonymous methods, and lambda expressions. Events and event handlers. Interfaces. Indexers. Generics and type-safe templates. Iterators and collection traversal. LINQ queries and transformations. Entity Framework Core basics and CRUD operations. SQL translation to Entity Framework and LINQ-to-Entities. Navigation properties, lazy and eager loading. Complex joins, tagging, transactions, and debugging in EF. Object-oriented approach to software design. Domain Analysis. Object-Oriented Analysis and Design. Programming paradigms. Object-Oriented Paradigm. Classes and interfaces. Relationships between objects in OOP. Encapsulation, polymorphism, and abstraction. The concept of inheritance. Coercion and parametric polymorphism. Modularity in OOP. Value and reference types. Dependency management. Dependency management: Implementation and libraries. Multithreading, concurrency. Asynchronous programming. Event-driven programming in OOP. Graphical user interface development, application deployment.

**Planned Learning Activities.** The minimum amount of classroom-based learning activities in one ECTS credit for a course at the first (Bachelor's) level of higher education in full-time study mode is 10 hours per 1 ECTS credit.

**Forms (Methods) of Instruction:** Lectures (using problem-based learning and visualization methods), Laboratory works, Independent work

**Assessment Methods:** Laboratory work defense, Testing

**Form of Final Assessment:** Exam

**Learning Resources:**


1. Joe Mayo. C# Cookbook: Modern Recipes for Professional Developers. O'Reilly Media, 2021.
2. Jon P. Smith. Entity Framework Core in Action (Second Edition). Manning Publications, 2021.
3. Brian L. Gorman. Practical Entity Framework Core 6: Database Access for Enterprise Applications. Apress, 2022.
4. Mrs. Anuradha A Puntambekar. Object-Oriented Analysis & Design, 2021.
5. Matthias Noback. Object Design Style Guide: A Set of Practices for Writing Object-Oriented Code. Manning. Publications, 2020.
6. Andrew Troelsen, Philip Japikse. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. Apress, 2022.
7. Ziegler S. Micah. Mastering C# Async Programming: Building Scalable and Responsive Applications, 2021.
8. Anthony Williams. Hands-On Concurrency with C++: Practical Guide for High-Performance Multithreading, 2019.
9. Mark J. Price. C# 10 and .NET 6 – Modern Cross-Platform Development. Packt Publishing, 2021.
10. Joseph Albahari, Ben Albahari. C# 12 in a Nutshell: The Definitive Reference. O'Reilly Media, 2024.
11. Paul Deitel, Harvey Deitel, Chuti Prasertsith. C++ 20 for programmers. An Object-natural Approach, 2022.
12. Christian Nagel. Professional C# and .NET (2021 Edition). Wiley, 2021.
13. Jesse Liberty, Rodrigo Juarez, Maddy Montaquila. .NET MAUI for C# Developers: Build Cross-Platform Mobile and Desktop Applications. Packt Publishing, 2023.
14. University Electronic Library. [Electronic resource]. – Available at: http://library.khmnu.edu.ua/
15. Institutional Repository of Khmelnytskyi National University. [Electronic resource]. – Available at: http://elar.khmnu.edu.ua/jspui/?locale=en
16. Modular Learning Environment. [Electronic resource]. – Available at: https://msn.khmnu.edu.ua/course/view.php?id=9679
17. Modular Learning Environment. [Electronic resource]. – Available at: https://msn.khmnu.edu.ua/course/view.php?id=9344


**Lecturers:** Doctor of Technical Sciences, Full Professor Martynyuk V.V.; Teaching assistant, Boiko V. O.

## 3. EXPLANATORY NOTE

The course "Object-oriented programming" is one of the general training courses and occupies a leading place in the training of students of the first (Bachelor's) level of higher education, full-time mode of study (hereinafter – full-time), who study under the Educational and Professional Programme "Software Engineering" within the specialty F2 "Software Engineering".

**Prerequisites** – CPT.03 Programming
**Postrequisites** – CPT.08 Object-oriented design.

In accordance with the educational programme, the course contributes to the development of: **competences:** GC1. Ability for abstract thinking, analysis, and synthesis. PC11. Ability to apply the principles of object-oriented programming for the design and implementation of software systems. PC12. Ability to use encapsulation, inheritance, polymorphism, and interfaces to build modular and reusable software components. PC13. Ability to reasonably choose and master the toolkit for object-oriented software development, including modern IDEs, libraries, and frameworks. PC14. Ability for algorithmic and logical thinking, with an emphasis on modeling real-world entities and relationships in software systems. PC15. Ability to design, debug, test, and document object-oriented applications according to professional standards and good practices.
**programme learning outcomes:** PLO1 Analyze and apply object-oriented principles to model, design, and implement software systems. PLO07 Understand and use in practice the fundamental concepts of encapsulation, inheritance, polymorphism, and abstraction in C#. PLO10 Perform object-oriented analysis of the subject area and design class hierarchies, relationships, and interaction models. PLO11 Select appropriate object-oriented patterns and techniques for requirement implementation and system modelling. PLO12 Apply effective OOP-based design approaches to ensure modularity, extensibility, and maintainability of software. PLO13 Know and apply methods for developing reusable components, generic templates, and collections in C#. PLO15 Select and use OOP language features, development tools, and frameworks in a well-grounded manner to solve tasks related to the creation and maintenance of object-oriented software.
**Purpose of the course.** To develop in students the competences in object-oriented programming using the OOP language, including problem analysis, system modelling, and implementation of modular, reusable, and maintainable software solutions with attention to efficiency, correctness, and professional programming practices.
**Subject of the course.** Fundamentals of object-oriented programming in the C# language, including class design, implementation of object-oriented principles, use of collections and generic structures, along with software analysis and object-oriented design techniques, and application of modern development tools and frameworks.
**Course objectives.** To provide students with knowledge and practical skills in designing, implementing, and debugging programs in C#, using object-oriented programming techniques, principles of encapsulation, inheritance, and polymorphism, generic structures and collections, LINQ, standard frameworks to solve typical software development problems efficiently and maintainably, and software analysis and object-oriented design techniques.
**Learning Outcomes.** Upon successful completion of the course, the student should be able to: understand the general structure of a C# program and apply syntax rules, namespaces, and assemblies correctly; work with files and streams, performing text and binary input/output operations using standard C# libraries; design and implement classes, define class members (fields, methods, properties), and apply encapsulation principles; apply inheritance and polymorphism to extend and reuse code, override methods, and implement abstract and virtual members; use delegates, anonymous methods, and lambda expressions to encapsulate behavior and implement functional-style programming in C#; create and manage events and event handlers, applying the publish-subscribe pattern for communication between objects; define and implement interfaces to enforce contracts and support multiple inheritance of behavior; use indexers to provide array-like access to objects and collections; apply generic templates to create type-safe and reusable data structures and methods; implement iterators with yield for efficient traversal of collections; work with LINQ (Language Integrated Query) to query and transform collections and databases,

including advanced LINQ operators; understand the basics of Entity Framework Core, create models, and perform CRUD operations through object-relational mapping (ORM); translate SQL queries into Entity Framework syntax and apply LINQ-to-Entities; use navigation properties, and apply lazy and eager loading to manage relationships between entities effectively; perform complex joins, tagging, transactions, and debugging in Entity Framework to build reliable database applications; model software systems using use-case and behavioral modeling, applying UML and diagrams to design object-oriented solutions; manage dependencies and ensure modularity, applying principles of clean architecture and design patterns; develop multi-threaded and asynchronous programs, ensuring thread safety and responsiveness using async/await patterns; implement graphical user interfaces, integrating event-driven programming concepts; debug, test, and document programs according to good OOP practices, and use IDE tools such as Visual Studio for compilation, debugging, and profiling.

## 4. STRUCTURE OF THE COURSE CREDITS

**Semester 2**

| Topic Title | Number of hours allocated to: | | |
|---|---|---|---|
| | Lectures | Lab work | Independent work |
| Topic 1. General structure of C# program. Files and Streams | 2 | 4 | 9 |
| Topic 2. Classes. Class members. Encapsulation, Inheritance and Polymorphism | 2 | 4 | 9 |
| Topic 3. Delegates. Anonymous methods. Lambda expressions. Events. Event Handlers | 2 | 4 | 9 |
| Topic 4. Interfaces, Indexers | 2 | 4 | 9 |
| Topic 5. Generic templates, Iterators | 2 | 4 | 9 |
| Topic 6. LINQ basics. Advanced LINQ | 2 | 4 | 9 |
| Topic 7. Entity Framework Core Basics. SQL to Entity Framework | 2 | 4 | 8 |
| Topic 8. EF: Navigation properties. Lazy and Eager loading. Complex joins. Tagging. Transactions. Debugging | 2 | 6 | 8 |
| **Total** | 16 | 34 | 70 |

**Semester 3**

| Topic Title | Number of hours allocated to: | | |
|---|---|---|---|
| | Lectures | Lab work | Independent work |
| Topic 1. Object-oriented approach to software design | 2 | 4 | 8 |
| Topic 2. Domain Analysis | 2 | 2 | 8 |
| Topic 3. Object-Oriented Analysis and Design | 2 | 4 | 8 |
| Topic 4. Programming paradigms. Object-Oriented Paradigm | 2 | 2 | 8 |
| Topic 5. Classes and interfaces | 2 | 4 | 8 |
| Topic 6. Relationships between objects in OOP | 2 | 2 | 8 |
| Topic 7. Encapsulation, polymorphism, abstraction | 2 | 4 | 8 |
| Topic 8. The concept of inheritance. Coercion and parametric polymorphism | 2 | 2 | 8 |
| Topic 9. Modularity in OOP | 2 | 4 | 8 |
| Topic 10. Value and reference types | 2 | 2 | 8 |

| | | | |
|---|---|---|---|
| Topic 11. Dependency management | 2 | 4 | 8 |
| Topic 12. Dependency management: Implementation and libraries | 2 | 2 | 8 |
| Topic 13. Multithreading, concurrency | 2 | 4 | 8 |
| Topic 14. Asynchronous programming | 2 | 2 | 8 |
| Topic 15. Event-driven programming in OOP | 2 | 4 | 8 |

| Topic Title | Number of hours allocated to: | | |
|---|---|---|---|
| | Lectures | Lab work | Independent work |
| Topic 16. Graphical user interface development, application deployment | 2 | 4 | 8 |
| **Total** | 32 | 50 | 128 |

## 5.1. CONTENT OF THE LECTURE COURSE

**Semester 2**

| Lecture No. | List of Lecture Topics and Annotations | Hours |
|---|---|---|
| **Topic 1. General structure of C# program. Files and Streams** | | |
| 1 | Overview of the C# programming language. History and characteristics of C#. Structure of a C# program, namespaces, and assemblies. Basic syntax, keywords, and identifiers. Variables, constants, and data types in C#. Operators and expressions. Compilation and execution process in .NET. Introduction to the Visual Studio integrated development environment. Writing, compiling, and running simple C# programs. Concept of streams in C#. Input and output classes in the System.IO namespace. File types and file modes. Reading and writing text files using StreamReader and StreamWriter. Working with binary files using FileStream, BinaryReader, and BinaryWriter. Handling directories and file system operations. Exception handling in file operations. Practical examples of creating, opening, reading, writing, and closing files. Introduction to asynchronous file I/O.<br>Ref.: [1, 7, 11] | 2 |
| **Topic 2. Classes. Class members. Encapsulation. Inheritance and Polymorphism** | | |

| Lecture No. | List of Lecture Topics and Annotations | Hours |
|---|---|---|
| 2 | Concept of classes in C#. Defining classes and objects. Class members: fields, properties, methods, and constructors. Access modifiers and their role in controlling visibility. Encapsulation as a fundamental principle of object-oriented programming. Use of properties to implement data hiding and controlled access. Static members and their usage. Practical examples of designing classes, creating objects, and applying encapsulation. Concept of inheritance in C#. Base and derived classes. Access to base class members. Overriding and hiding members. Concept of polymorphism and dynamic method dispatch. Virtual methods and overriding rules. Use of the object class and type casting. Interfaces as a form of multiple inheritance.<br><br>Ref.: [1, 7, 11] | 2 |
| **Topic 3. Delegates. Anonymous methods. Lambda expressions. Events. Event Handlers** | | |
| 3 | Concept of delegates in C# and their role as type-safe references to methods. Declaring and instantiating delegates. Multicast delegates and combining method calls. Introduction to anonymous methods and their syntax. Transition from anonymous methods to lambda expressions. Syntax and types of lambda expressions: expression-bodied and statement-bodied forms. Use of delegates and lambda expressions in event handling and LINQ queries. Practical examples demonstrating the advantages of functional-style programming with delegates and lambdas. Concept of events in C# as a mechanism for communication between objects. Relationship between delegates and events. Declaring and raising events using the event keyword. Standard event design pattern with EventHandler and EventArgs. Subscribing to and unsubscribing from events. Event propagation and multiple subscribers. Practical examples of event handling in user-defined classes and graphical user interface applications. Introduction to custom event arguments and best practices for event-driven programming.<br><br>Ref.: [1, 7, 11] | 2 |
| **Topic 4. Interfaces. Indexers** | | |
| 4 | Concept of interfaces in C# as contracts for defining behavior. Declaring and implementing interfaces. Differences between interfaces and abstract classes. Multiple interface implementation and resolving naming conflicts. Inheritance between interfaces. Use of built-in interfaces such as IComparable, IEnumerable, and IDisposable. Concept of indexers in C# as a mechanism for array-like access to objects. Declaring and implementing indexers with parameters and return types. One-dimensional and multi-dimensional indexers. Read-only and write-only indexers. Overloading indexers and combining with properties. Practical examples of applying indexers in custom collection classes.<br><br>Ref.: [1, 7, 11] | 2 |

| Lecture No. | List of Lecture Topics and Annotations | Hours |
|---|---|---|
| **Topic 5. Generic templates. Iterators** | | |
| 5 | Concept of generics in C# for creating type-safe and reusable code. Declaring generic classes, methods, and interfaces. Type parameters and constraints (where keyword). Advantages of generics compared to non-generic collections. Overview of built-in generic collections in the System.Collections.Generic namespace. Practical examples of implementing generic methods and classes. Concept of iterators in C# as a mechanism for sequential access to collections.<br><br>Ref.: [1, 7, 11] | 2 |
| **Topic 6. LINQ basics. Advanced LINQ** | | |
| 6 | Introduction to Language Integrated Query in C#. Concept and advantages of LINQ as a unified approach to querying data. Syntax of query expressions and method-based syntax. Basic LINQ operations: filtering, projection, ordering, and grouping. Using LINQ with collections. Practical examples of applying LINQ queries to arrays and generic collections. Advanced features of LINQ in C#. Extension methods and lambda expressions in LINQ queries. Complex operations: joins, nested queries, groupings, and aggregations. Working with multiple data sources and query composition. LINQ to Objects, LINQ to XML, and LINQ to Entities overview.<br><br>Ref.: [2, 11] | 2 |
| **Topic 7. Entity Framework Core Basics. SQL to Entity Framework** | | |
| 7 | Introduction to object-relational mapping (ORM) and the role of Entity Framework Core in C#. Overview of EF Core architecture and features. Creating a data model using classes and properties. Configuring the database context (DbContext) and defining DbSet properties. Database-first and code-first approaches. Performing basic CRUD operations: create, read, update, and delete. Applying migrations to manage database schema changes. Practical examples of integrating EF Core into C# applications. Mapping SQL queries to Entity Framework Core. Translation of basic SQL operations (SELECT, INSERT, UPDATE, DELETE) into LINQ-to-Entities queries. Filtering, sorting, and joining data using EF Core. Executing raw SQL queries when necessary. Comparison of SQL syntax and EF Core methods. Advantages and limitations of using LINQ versus direct SQL. Practical examples of rewriting SQL queries into EF Core code.<br><br>Ref.: [2, 3] | 2 |
| **Topic 8. EF: Navigation properties. Lazy and Eager loading. Complex joins. Tagging.** | | |

| Lecture No. | List of Lecture Topics and Annotations | Hours |
|---|---|---|
| **Transactions. Debugging** | | |
| 8 | Concept of navigation properties in Entity Framework Core for representing relationships between entities. One-to-one, one-to-many, and many-to-many associations. Configuring relationships with data annotations and Fluent API. Loading related data. Practical examples of working with entity relationships in real applications. Applying query tagging for diagnostics and performance monitoring. Implementing transactions to ensure data consistency and integrity.<br>Ref.: [2, 3] | 2 |
| | **Total** | **16** |

**Semester 3**

| Lecture No. | List of Lecture Topics and Annotations | Hours |
|---|---|---|
| **Topic 1. Object-oriented approach to software design** | | |
| 1 | Overview of the object-oriented approach to software design. Software design principles. Design process stages.<br>Ref.: [4, 5] | 2 |
| **Topic 2. Domain Analysis** | | |
| 2 | Phases of software design: domain analysis, object-oriented analysis, object-oriented design, programming, refactoring, testing, deployment control. Identifying key entities, relationships, processes. Domain and problem domain. Domain scope. Use-case modeling, use-case diagrams. Features, feature trees, feature model, Feature-oriented domain analysis.<br>Ref.: [4, 5] | 2 |
| **Topic 3. Object-Oriented Analysis and Design** | | |
| 3 | Object-Oriented Analysis and Design: methodology, stakeholder communication, product quality, visual modeling, iterative and incremental process. History of OOAD. OOA vs OOD differences. Behavioral modeling: sequence, activity, state, collaboration UML diagrams.<br>Ref.: [4, 5] | 2 |
| **Topic 4. Programming paradigms. Object-Oriented Paradigm** | | |

| Lecture No. | List of Lecture Topics and Annotations | Hours |
|---|---|---|
| 4 | Software development paradigms: assumptions, concepts, values, practices, paradigm shift. Object-Oriented Paradigm: history, definitions, key elements, core concepts, purpose, disadvantages.<br>Ref.: [4, 5] | 2 |
| **Topic 5. Classes and interfaces** | | |
| 5 | Classes and interfaces: backbone of OOP design, support modularity, abstraction, reusability. Object state, behavior, mutability and immutability. Responsibility, Class Responsibility Collaboration cards.<br>Ref.: [1, 4, 5] | 2 |
| **Topic 6. Relationships between objects in OOP** | | |
| 6 | Concept of object relationships in object-oriented programming. Types of associations: dependency, association, aggregation, and composition. Differences between strong and weak relationships. One-to-one, one-to-many, and many-to-many associations in class design. Implementing relationships using references, collections, and properties. Role of constructors and destructors in managing object lifecycles.<br>Ref.: [4, 5] | 2 |
| **Topic 7. Encapsulation, polymorphism, abstraction** | | |
| 7 | Fundamental principles of object-oriented programming. Concept of encapsulation and its role in data hiding and controlled access to class members. Use of properties and access modifiers to implement encapsulation. Concept of polymorphism as the ability of objects to take different forms. Compile-time polymorphism through method overloading and operator overloading. Runtime polymorphism through virtual methods and overriding. Concept of abstraction as the process of highlighting essential features while hiding implementation details. Use of abstract classes and interfaces to achieve abstraction.<br>Ref.: [1, 4, 5, 6, 7, 11] | 2 |
| **Topic 8. The concept of inheritance. Coercion and parametric polymorphism** | | |
| 8 | Concept of inheritance in object-oriented programming as a mechanism for reusing and extending code. Concept of coercion (type conversion) in OOP, including implicit and explicit casting between base and derived types. Safe type conversions using is, as, and pattern matching. Parametric polymorphism through generics. Advantages of generics for type safety and reusability. | 2 |

| Lecture No. | List of Lecture Topics and Annotations | Hours |
|---|---|---|
| | Ref.: [1, 4, 5, 6, 7, 11] | |
| **Topic 9. Modularity in OOP** | | |
| 9 | Concept of modularity in object-oriented programming as a principle of dividing software into independent and reusable components. Role of classes, interfaces, and namespaces in achieving modular design. Benefits of modularity: readability, maintainability, scalability, and ease of testing. Use of access modifiers to control visibility between modules. Importance of modularity. Levels of modularity. Large-scale modularity. Modularization strategies.<br>Ref.: [4, 5] | 2 |
| **Topic 10. Value and reference types** | | |
| 10 | Concept of value types and reference types. Differences in memory allocation on the stack and heap. Examples of value types. Examples of reference types. Behavior of assignment, parameter passing, and comparison for value and reference types. Use of the “ref” and “out” keywords for passing parameters by reference. Nullable value types and the role of Nullable<T>. Garbage Collection and memory management. Dispose pattern. .<br>Ref.: [1, 6, 11] | 2 |
| **Topic 11. Dependency management** | | |
| 11 | Concept of dependencies in object-oriented programming and their impact on software design. Loose coupling and high cohesion. Techniques for reducing dependencies between classes and modules. Dependency injection as a design pattern for managing dependencies. Constructor injection, property injection, and method injection. Role of interfaces in achieving abstraction and flexibility.<br>Ref.: [6] | 2 |
| **Topic 12. Dependency management: Implementation and libraries** | | |
| 12 | Practical implementation of dependency management. Applying dependency injection to decouple components and improve testability. Use of built-in .NET Core dependency injection container. Registration of services with different lifetimes: transient, scoped, and singleton. Constructor-based, property-based, and method-based injection in practice. Overview of popular dependency injection libraries. Integration of dependency management into layered and modular architectures.<br>Ref.: [6] | 2 |

| Lecture No. | List of Lecture Topics and Annotations | Hours |
|---|---|---|
| **Topic 13. Multithreading and concurrency** | | |
| 13 | Concept of multithreading in C# and its role in parallel execution. Creating and managing threads using the Thread class. Thread states and lifecycle. Synchronization problems: race conditions, deadlocks, and thread safety. Mechanisms for synchronization: locks, monitors, mutexes, and semaphores. Introduction to the ThreadPool and task-based programming model (Task Parallel Library). Concept of concurrency and its difference from parallelism. Designing applications that efficiently use multiple threads while avoiding common pitfalls.<br>Ref.: [6, 8] | 2 |
| **Topic 14. Asynchronous programming** | | |
| 14 | Concept of asynchronous programming in C# and its advantages for responsive applications. Difference between synchronous, multithreaded, and asynchronous execution. Tasks and the Task-based Asynchronous Pattern (TAP). Returning values from asynchronous methods. Exception handling in asynchronous code. Combining multiple asynchronous operations. Practical examples of asynchronous programming in I/O-bound and network operations.<br>Ref.: [6, 8] | 2 |
| **Topic 15. Event-driven programming in OOP** | | |
| 15 | Concept of event-driven programming as a paradigm where program execution is determined by events. Role of objects, events, and event handlers in C#. Delegates as the foundation of event handling. Standard event design pattern. Event subscription and unsubscription mechanisms. Practical examples of event-driven programming in graphical user interfaces and real-time systems. Advantages of event-driven design for building interactive, modular, and responsive applications.<br>Ref.: [1, 6] | 2 |
| **Topic 16. Graphical user interface development** | | |
| 16 | Concept of graphical user interfaces (GUI) in software applications. Overview of GUI frameworks in .NET such as Windows Forms, WPF, and MAUI. Event-driven model of GUI programming. Designing windows, controls, and layouts. Handling user input through events and data binding. Separating logic and presentation using patterns such as MVC and MVVM. Practical examples of building simple GUI applications in C#.<br>Best practices for usability, responsiveness, and maintainability in GUI development. | 2 |

| Lecture No. | List of Lecture Topics and Annotations | Hours |
|---|---|---|
| | Ref.: [13] | |
| | **Total** | **32** |

## 5.2. CONTENT OF LABORATORY WORKS

**Semester 2**

| Topic No. | Laboratory Session Topic | Hours |
|---|---|---|
| 1 | General structure of C# program.<br>Ref.: [1, 7, 11] | 4 |
| 2 | Inheritance and Polymorphism.<br>Ref.: [1, 7, 11] | 4 |
| 3 | Delegates. Anonymous methods. Lambda expressions.<br>Ref.: [1, 7, 11] | 4 |
| 4 | Interfaces, indexers.<br>Ref.: [1, 7, 11] | 4 |
| 5 | Generic templates. Iterators.<br>Ref.: [1, 7, 11] | 4 |
| 6 | LINQ.<br>Ref.: [2, 11] | 4 |
| 7 | Entity Framework Core. Basic CRUD operations.<br>Ref.: [2, 3] | 4 |
| 8 | Entity Framework Core. Advanced usage.<br>Ref.: [2, 3] | 6 |
| **Total** | | **34** |

**Semester 3**

| Topic No. | Laboratory Session Topic | Hours |
|---|---|---|
| 1 | Use case analysis.<br>Ref.: [4, 5] | 6 |

| Topic No. | Laboratory Session Topic | Hours |
|---|---|---|
| 2 | Behavioral modeling.<br>Ref.: [4, 5] | 6 |
| 3 | Class design.<br>Ref.: [1, 4, 5] | 6 |
| 4 | Classes and interfaces implementation.<br>Ref.: [1, 4, 5, 6, 7, 11] | 6 |
| 5 | Modules development.<br>Ref.: [4, 5] | 6 |
| 6 | Dependency management.<br>Ref.: [6] | 6 |
| 7 | Multithreading and asynchronous programming.<br>Ref.: [6, 8] | 6 |
| 8 | Graphical user interface development. Application deployment.<br>Ref.: [13] | 8 |
| **Total** | | **50** |

## 5.3. CONTENT OF INDEPENDENT WORK

Independent work of students of all forms of study involves systematic processing of the course material from relevant sources, preparation for laboratory works and testing. Students have access to the course page in the Modular Learning Environment, which contains the Working Programme of the course and the necessary teaching and learning materials.

**Semester 2**

| Week No. | Type of Independent Work | Hours |
|---|---|---|
| 1 | Study of theoretical material from T1, preparation for Laboratory Work No. 1 | 9 |
| 2 | Study of theoretical material from T2, preparation for Laboratory Work No. 2 | 9 |
| 3 | Study of theoretical material from T3, preparation for Laboratory Work No. 3 | 9 |
| 4 | Study of theoretical material from T4, preparation for Laboratory Work No. 4. Preparation for TC No. 1 | 9 |
| 5 | Study of theoretical material from T5, preparation for Laboratory Work No. 5 | 9 |
| 6 | Study of theoretical material from T6, preparation for Laboratory Work No. 6 | 9 |
| 7 | Study of theoretical material from T7, preparation for Laboratory Work No. 7 | 8 |

| Week No. | Type of Independent Work | Hours |
|---|---|---|
| 8 | Study of theoretical material from T8, preparation for Laboratory Work No. 8. Preparation for TC No. 2 | 8 |
| **Total:** | | **70** |

**Notes:** TC – Test Control, T1–T8 – Topics of the course.

**Semester 3**

| Week No. | Type of Independent Work | Hours |
|---|---|---|
| 1 | Study of theoretical material from T1, preparation for Laboratory Work No. 1 | 8 |
| 2 | Study of theoretical material from T2, preparation for Laboratory Work No. 1 | 8 |
| 3 | Study of theoretical material from T3, preparation for Laboratory Work No. 2 | 8 |
| 4 | Study of theoretical material from T4, preparation for Laboratory Work No. 2 | 8 |
| 5 | Study of theoretical material from T5, preparation for Laboratory Work No. 3 | 8 |
| 6 | Study of theoretical material from T6, preparation for Laboratory Work No. 3 | 8 |
| 7 | Study of theoretical material from T7, preparation for Laboratory Work No. 4 | 8 |
| 8 | Study of theoretical material from T8, preparation for Laboratory Work No. 4. Preparation for TC No. 1 | 8 |
| 9 | Study of theoretical material from T9, preparation for Laboratory Work No. 5 | 8 |
| 10 | Study of theoretical material from T10, preparation for Laboratory Work No. 5 | 8 |
| 11 | Study of theoretical material from T11, preparation for Laboratory Work No. 6 | 8 |
| 12 | Study of theoretical material from T12, preparation for Laboratory Work No. 6 | 8 |
| 13 | Study of theoretical material from T13, preparation for Laboratory Work No. 7 | 8 |
| 14 | Study of theoretical material from T14, preparation for Laboratory Work No. 7 | 8 |
| 15 | Study of theoretical material from T15, preparation for Laboratory Work No. 8 | 8 |
| 16 | Study of theoretical material from T16, preparation for Laboratory Work No. 8. Preparation for TC No. 2 | 8 |

| Week No. | Type of Independent Work | Hours |
|---|---|---|
| **Total:** | | **128** |

**Notes:** TC – Test Control, T1–T16 – Topics of the course.

## 6. TECHNOLOGIES AND TEACHING METHODS

The learning process for the course is based on the use of both traditional and modern teaching technologies and methods, in particular: lectures (using visualisation methods, problem-based and interactive learning, motivational techniques, and information and communication technologies); laboratory works (using training exercises, problem situation analysis, explanation, discussions, etc.); independent work (study of theoretical material, preparation for laboratory works, ongoing and final assessment), with the use of information and computer technologies and distance learning technologies.

## 7. METHODS OF ASSESSMENT

Ongoing assessment is carried out during practical classes, as well as on the days of control activities established by the working programme and the academic schedule.
The following methods of ongoing assessment are used:
– test-based assessment of theoretical material;
– evaluation of the results of laboratory work defense.
When determining the final semester grade, the results of both ongoing assessment and final assessment are taken into account. The final assessment is conducted on all the material of the course according to examination papers prepared in advance and approved at the meeting of the department.
A student who has scored less than 60 percent of the maximum score for any type of academic work is not allowed to undergo the semester assessment until the amount of work stipulated by the Working Programme is completed. A student who has achieved a positive weighted average score (60 percent or more of the maximum score) for all types of ongoing assessment but has failed the examination is considered to have an academic debt.
Elimination of academic debt for the semester assessment is carried out during the examination session or according to the schedule set by the dean's office in accordance with the *Regulation on Control and Assessment of Learning Outcomes of Students at Khmelnytskyi National University*.

## 8. COURSE POLICY

The policy of the academic course is generally determined by the system of requirements for the student as stipulated by the current University regulations on the organization and teaching and learning support of the educational process. In particular, this includes completing the safety briefing; attendance at course classes is compulsory. For valid reasons (documentarily confirmed), theoretical training may, with the lecturer's approval, take place online. Successful completion of the course and the formation of professional competences and programme learning outcomes require preparation for each laboratory work (studying the theoretical material for the topic of the work), active participation during the class, thorough preparation of the report, defense of the results, participation in discussions regarding the constructive decisions made during the laboratory works, etc.
Students must meet the established deadlines for completing all types of academic work in accordance with the Working Programme of the course. A missed laboratory class must be

completed within the deadline set by the lecturer, but no later than two weeks before the end of the theoretical classes in the semester.
The student's mastery of the theoretical material of the course is assessed through testing.
When performing laboratory work, the student must comply with the policy of academic integrity (cheating, plagiarism, including the use of mobile devices, are prohibited). If a violation of academic integrity is detected in any type of academic work, the student receives an unsatisfactory grade and must re-do the task on the relevant topic (type of work) as stipulated by the Working Programme. Any form of academic dishonesty is unacceptable.
Within the framework of studying the course, students are provided with recognition and crediting of learning outcomes acquired through non-formal education, available on accessible platforms (https://prometheus.org.ua/, https://www.coursera.org/), which contribute to the formation of competences and the deepening of learning outcomes defined in the Working Programme of the course, or ensure the study of a relevant topic and/or type of work from the course syllabus (for more details, see the *Regulation on the Procedure for Recognition and Crediting of Learning Outcomes of Students at Khmelnytskyi National University*).

## 9. ASSESSMENT OF STUDENTS' LEARNING OUTCOMES DURING THE SEMESTER

Assessment of a student's academic achievements is carried out in accordance with the *Regulation on the Control and Assessment of Students' Learning Outcomes at Khmelnytskyi National University*. During the ongoing assessment of the work performed by the student for each structural unit and the results obtained, the lecturer awards a certain number of points as set out in the Working Programme for that type of work.
Each structural unit of academic work may be credited only if the student has scored at least 60 percent (the minimum level for a positive grade) of the maximum possible points assigned to that structural unit.
When assessing students' learning outcomes for any type of academic work (structural unit), it is recommended to use the generalised criteria provided below:

**Table – Assessment Criteria for Student Learning Outcomes ( for 3-5 Grade)**

| Grade and Level of Achievement of Intended Learning Outcomes and Competences | General Description of Assessment Criteria |
|---|---|
| **Excellent** *(High)* | The student has deeply and fully mastered the course content, confidently navigates it, and skilfully uses the conceptual framework; The student demonstrates the ability to connect theory with practice, solve practical problems, and clearly express and justify their reasoning. An excellent grade implies a logical presentation of the answer in the language of instruction (oral or written), high-quality formatting of the work, and proficiency in using specialised tools, instruments, or application software. The student demonstrates confidence when answering reformulated questions, is capable of making detailed and summarised conclusions, and shows practical skills in solving professional tasks. The answer may contain two or three minor inaccuracies. |

| Grade and Level of Achievement of Intended Learning Outcomes and Competences | General Description of Assessment Criteria |
|---|---|
| **Good** *(Average)* | The student has shown full understanding of the course content, possesses the conceptual framework, and navigates the material well; applies theoretical knowledge consciously to solve practical tasks. The answer is generally well-articulated, although some minor inaccuracies or vague formulations of rules or principles may occur. The student's answer is based on independent thinking. Two or three minor mistakes are acceptable. |
| **Satisfactory** *(Sufficient)* | The student demonstrates knowledge of the basic course material sufficient for continued learning and practical activity in the profession; is able to complete the practical tasks foreseen by the programme. The answer is usually based on reproductive thinking. The student has limited knowledge of the structure of the discipline, makes inaccuracies and significant errors in the answer, and hesitates when answering reformulated questions. Nevertheless, they possess basic skills to complete simple practical tasks that meet the minimum assessment criteria and, under the lecturer's guidance, can correct their mistakes. |
| **Unsatisfactory** *(Insufficient)* | The student demonstrates fragmented, unstructured knowledge, cannot distinguish between main and secondary ideas, makes conceptual errors, misinterprets definitions, presents material in a chaotic and unconfident manner, and cannot apply knowledge to solve practical problems. An unsatisfactory grade is typically given to a student who is unable to continue learning the subject without additional study. |

**Table – Assessment Criteria for Student Learning Outcomes ( for 6-10 Grade)**

| Grade and Level of Achievement of Intended Learning Outcomes and Competences | General Description of Assessment Criteria |
|---|---|
| **E**xcellent (High)<br>10 | The student has mastered the content of the learning material deeply and comprehensively, easily navigates it, and skillfully uses the conceptual framework; can link theory to practice, solve practical problems, and confidently express and justify their opinions. An excellent grade assumes a logical presentation of the answer in the language of instruction (orally or in writing), demonstrates high-quality work design and proficiency in special devices, tools, and application programs. The student does not hesitate when the question is modified, can make detailed and generalizing conclusions, and demonstrates practical skills in solving professional tasks. Two or three minor inaccuracies were made in the response. |

| | |
|---|---|
| | |
| Very Good (Above Average)<br>9 | The student has demonstrated deep and consistent mastery of the learning material, freely operates with the conceptual framework, and shows a high level of independent thinking. They confidently navigate the material, can connect theory with practice, and give logical, consistent, and well-reasoned answers, although one or two minor mistakes or inaccuracies may be present. The response is generally well-structured, and the written work is of high quality. The student shows initiative in solving practical tasks and sufficient flexibility of thinking under changed conditions of the task. They possess the necessary practical skills and can work independently without substantial support from the instructor. |
| Good (Average)<br>8 | The student has mastered the learning material to the planned extent, generally possesses the conceptual framework, and is oriented in the studied topics but makes minor mistakes or shows an unclear understanding of certain provisions. The response is mostly based on learned examples and models and may contain some inaccuracies, template formulations, or repetitions. Theoretical knowledge is applied to solve typical practical tasks but without initiative or flexibility under changed conditions. The answers are based on independent thinking but have a lower depth of reasoning compared to higher levels. The written work may be of moderate quality, although its structure is preserved. |
| Satisfactory (Sufficient)<br>7 | The student has mastered the basic curriculum material sufficiently to allow further learning and performance of typical practical tasks in the field. They possess basic concepts, although the answers contain inaccuracies or errors that can be identified and corrected with the help of guiding questions from the instructor. The response shows the ability for elementary analysis but is mostly reproductive thinking. The structure of the discipline is partially mastered, and orientation in the material is unstable; however, the answers are logically complete. The student demonstrates skills in performing simple tasks of typical scenarios and usually hesitates when answering modified or complex questions but tries to justify their opinion. |
| Satisfactory (Marginally Sufficient)<br>6 | The student has demonstrated fragmented mastery of the basic curriculum material; their answers are mostly superficial, limited to simple reproduction of individual facts or definitions. They show a low level of orientation in the structure of the discipline and often hesitate even when answering typical questions. The answers contain significant mistakes, and the student does not always understand the meaning of concepts and is unable to independently correct deficiencies. Performance of practical tasks is possible only in the simplest cases and under constant instructor supervision. Nevertheless, the student has minimal but present signs of forming professional skills necessary for further study with appropriate support. |
| Unsatisfactory (Insufficient) | The student has demonstrated disjointed, unsystematic knowledge, is unable to distinguish between the essential and the secondary, makes mistakes in defining concepts, distorts their meaning, presents the material chaotically and insecurely, and cannot use knowledge to solve practical tasks. As a rule, an “unsatisfactory” grade is given to a student who cannot continue learning without additional work on the discipline. |

**Structuring of the Course by Types of Academic Work and Assessment of Student Learning Outcomes**

**Semester 2**

| ***In-Class Work*** | | | | | | | | ***Assessment Activities*** | | ***Semester Final Assessment*** |
|---|---|---|---|---|---|---|---|---|---|---|
| ***Laboratory Work*** №: | | | | | | | | Test control: | | Pass/fail test |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | T 1- | T 5-8 | |
| ***Number of points per type of academic work (min–max)*** | | | | | | | | | | |
| 6-10 | 6-10 | 6-10 | 6-10 | 6-10 | 6-10 | 6-10 | 6-10 | 6-10 | 6-10 | According to the rating |
| **48-80** | | | | | | | | **12-20** | | **60-100*** |

**Semester 3**

| ***In-Class Work*** | | | | | | | | ***Assessment Activities*** | | ***Semester Final Assessment*** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Laboratory Work*** №: | | | | | | | | Test control: | | Exam | Total |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | T 1-4 | T 5-8 | | |
| ***Number of points per type of academic work (min–max)*** | | | | | | | | | | | |
| 3-5 | 3-5 | 3-5 | 3-5 | 3-5 | 3-5 | 3-5 | 3-5 | 6-10 | 6-10 | 24-40 | **60-100*** |
| **24-40** | | | | | | | | **12-20** | | **24-40** | |

**Notes:** If the number of points earned for any type of academic work in the course is below the established minimum, the student receives a failing grade and must retake the work within the deadline set by the lecturer (or dean). The institutional grade is determined in accordance with the table "**Correspondence between the Institutional Grading Scale and the ECTS Grading Scale**".

**Assessment of Laboratory Work Defense Results**

A laboratory work completed and formatted in accordance with the requirements established in the Methodological Guidelines is comprehensively assessed by the lecturer during its defense based on the following criteria:

– independence and accuracy of execution;

– completeness of the answer and understanding of the principles of building machine learning models;

– ability to justify the choice of algorithm or method;

– correctness of model implementation in the Python programming environment using appropriate libraries;

– ability to interpret the results of modelling and evaluate their suitability for solving the given task.

When assessing a laboratory session, the lecturer uses the generalized criteria outlined in the table "Assessment Criteria for Student Learning Outcomes" (minimum passing score – 3 points, maximum – 5 points in 3th semester and minimum passing score – 6 points, maximum – 10 points in 2th semester)

If the student demonstrates a knowledge level below 60 percent of the maximum score established in the Working Programme for each structural unit, the laboratory work is not credited. In such a case, the student must study the topic more thoroughly, review the methodology, correct major mistakes, and redefend the work at the time set by the lecturer.

**Assessment of Test-Based Control Results**

Each test included in the Working Programme consists of 30 test items, each carrying equal weight. According to the table for structuring types of academic work, the student may receive between 3 and 5 points depending on the number of correct answers.

**Distribution of points depending on the number of correct answers to test items:**

The test duration is 30 minutes. Students complete the test online in the Modular Learning

Environment.
If a failing grade is received, the test must be retaken before the next scheduled assessment.

**Distribution of points depending on correct answers to test questions**

| **Number of Correct Answers** | **1–17** | **18–20** | **21–23** | **24–25** | **26–27** | **28–30** |
|---|---|---|---|---|---|---|
| **Percentage of Correct Answers** | 0–59 | 60–69 | 70–79 | 80–86 | 87–90 | 93–100 |
| **Score** | – | 6 | 7 | 8 | 9 | 10 |

The final semester grade according to the institutional grading scale and the ECTS grading scale is determined automatically after the lecturer enters the assessment results in points for all types of academic work into the electronic gradebook. The correspondence between the institutional grading scale and the ECTS grading scale is provided in the table "Correspondence" below.

**Assessment of the Final Semester Control (Exam)**

The educational programme provides for a final semester control in the form of an examination, the purpose of which is to systematically and objectively assess both the theoretical and practical preparation of the student in the course. The examination is conducted according to examination papers prepared in advance and approved at the meeting of the department. In accordance with this, the examination paper contains a combination of both theoretical questions (including in test form) and practical tasks.

**Table – Assessment of Final Semester Examination Results** *for full-time students (40 points allocated for final control)*

| **Type of Task** | **For each individual type of task** | | |
|---|---|---|---|
| | **Minimum (Satisfactory) Score** | ***Potential Positive Score (Good)**** | **Maximum (Excellent) Score** |
| Theoretical Question № 1 | 3 | 4 | 5 |
| Theoretical Question № 2 | 3 | 4 | 5 |
| Practical Tasks *(6 tasks worth 3 points each)* | 18 | 24 | 30 |
| **Total:** | **24** | | **40** |

**Note.** ***A passing score for the exam, different from the minimum (24 points) and the maximum (40 points), falls within the range of 25–39 points and is calculated as the sum of points for all structural elements (tasks) of the exam.***

For each individual type of task in the final semester assessment, the assessment criteria for student learning outcomes provided above (see **Table – Assessment Criteria for Student Learning Outcomes**) are applied.

The final semester grade according to the institutional grading scale and the ECTS grading scale is determined automatically after the lecturer enters the assessment results in points for all types of academic work into the electronic gradebook. The correspondence between the institutional grading scale and the ECTS grading scale is shown below in the **Correspondence Table**.

The final examination grade is recorded if the total number of points accumulated by the student in the course as a result of ongoing assessment falls within the range of 60 to 100 points. In this case, a grade of *Excellent/Good/Satisfactory* is assigned according to the institutional scale, and a letter grade is assigned according to the ECTS scale, corresponding to the total number of points earned by the student as specified in the **Correspondence Table**.

Table – Correspondence between the Institutional Grading Scale and the ECTS Grading Scale

<table>
<tr><th rowspan="2">ECTS Grade</th><th rowspan="2">Rating Scale (Points)</th><th colspan="2">Institutional Grade(Level of Achievement of the Intended Learning Outcomes in the Course)</th></tr>
<tr><th>Pass/ Fail</th><th>Exam / Graded Credit</th></tr>
<tr><td>A</td><td>90-100</td><td rowspan="5">Pass</td><td>Excellent – a high level of achievement of the intended learning outcomes in the course, indicating the learner’s full readiness for further study and/or professional activity in the field.</td></tr>
<tr><td>B</td><td>83-89</td><td rowspan="4">Good – an average (maximally sufficient) level of achievement of the intended learning outcomes in the course and readiness for further study and/or professional activity in the field.<br><br>Satisfactory – the student has demonstrated a minimally sufficient level of achievement of the learning outcomes required for further study and/or professional activity in the field.</td></tr>
<tr><td>C</td><td>73-82</td></tr>
<tr><td>D</td><td>66-72</td></tr>
<tr><td>E</td><td>60-65</td></tr>
<tr><td>FX</td><td>40-59</td><td rowspan="2">Fail</td><td>Fail – several intended learning outcomes in the course have not been achieved. The level of acquired learning outcomes is insufficient for further study and/or professional activity in the field.</td></tr>
<tr><td>F</td><td>0-39</td><td>Fail – no learning outcomes have been achieved.</td></tr>
</table>

## 10. SELF-ASSESSMENT QUESTIONS ON LEARNING OUTCOMES

1. Domain analysis.
2. Problem domain. Identification of the problem domain.
3. Use case diagram.
4. Feature-oriented design of the software system.
5. Object-oriented analysis. Goals, tasks, and stages.
6. Behavioral modeling. Goals, tasks, and stages.
7. Activity diagram.
8. Sequence diagram.
9. State diagram.
10. Collaboration diagram.
11. Programming paradigms.
12. Imperative paradigm. OOP as an imperative paradigm.
13. Declarative paradigm.
14. Object-oriented paradigm.
15. Classes and interfaces.
16. Object, instance of a class.
17. Responsibilities of a class. Single Responsibility Principle.
18. CRC (Class Responsibility Collaboration) cards.
19. Cohesion. High cohesion.
20. Coupling. Low coupling.
21. Categories of programming languages implementing OOP principles.

22. Class diagram.
23. Relationships between classes.
24. Relationships between classes and interfaces. Relationships between interfaces.
25. Categories of classes.
26. Object diagram.
27. Encapsulation.
28. Inheritance.
29. Polymorphism.
30. Generics. Subtype polymorphism.
31. Abstraction.
32. Class extension.
33. Multiple inheritance.
34. Technical implementation of programming principles.
35. Modularity. Module.
36. Package diagram.
37. Technical implementation of modularity.
38. Decomposition of the software system into modules. Goals and tasks.
39. Package managers. Publishing independent modules.
40. Dependency management. Dependency tree.
41. Dependency injection principle. IoC container.
42. Technical implementation of dependency injection. DI libraries.
43. Value types and reference types. Implementation features in OOP languages.
44. Garbage collection in OOP languages. Memory cleanup.
45. Event-driven programming.
46. General concept of an event.
47. Delegates and events.
48. Event implementation in OOP languages.
49. Event-driven programming: mechanisms and implementation.
50. Event-driven programming. Events and OOP.
51. Graphical user interface design.
52. The role of OOP in user interface design.
53. Events and their handling in user interface design.
54. Frozen UI handling.
55. Handling changes and states of user interface elements.
56. Multithreading.
57. Semaphores.
58. Mutexes.
59. Thread locking. Deadlocks.
60. Implementation of multithreading in OOP languages.
61. Asynchronous programming.
62. State machine in asynchronous programming.
63. Concept of an asynchronous task.
64. Multithreaded and asynchronous programming.
65. Problems solved by asynchronous programming.
66. Technical implementation of asynchronous programming in OOP languages.
67. Object-oriented approach to software testing.
68. Cancellation token in asynchronous programming.
69. Threads. Thread-safety programming.
70. Application installer. Implementation of the installer.

## 11. EDUCATIONAL AND METHODOLOGICAL SUPPORT

The educational process for the course *"*Object-oriented programming*"* is supported with all necessary instructional and methodological materials, which are available in the Modular Learning Environment MOODLE:

1. Course *"*Object-oriented programming (2 Semester)*"*: https://msn.khmnu.edu.ua/course/view.php?id=9679
2. Methodological Guidelines for Laboratory Sessions: https://msn.khmnu.edu.ua/course/view.php?id=9679
3. Course *"*Object-oriented programming (3 Semester)*"*: https://msn.khmnu.edu.ua/course/view.php?id=9344
4. Methodological Guidelines for Laboratory Sessions: https://msn.khmnu.edu.ua/course/view.php?id=9344

## 12. RECOMMENDED LITERATURE

### Primary


1. Joe Mayo. C# Cookbook: Modern Recipes for Professional Developers. O'Reilly Media, 2021.
2. Jon P. Smith. Entity Framework Core in Action (Second Edition). Manning Publications, 2021.
3. Brian L. Gorman. Practical Entity Framework Core 6: Database Access for Enterprise Applications. Apress, 2022.
4. Mrs. Anuradha A Puntambekar. Object-Oriented Analysis & Design, 2021.
5. Matthias Noback. Object Design Style Guide: A Set of Practices for Writing Object-Oriented Code. Manning. Publications, 2020.
6. Christian Nagel. Professional C# and .NET (2021 Edition). Wiley, 2021.


### Supplementary


7. Andrew Troelsen, Philip Japikse. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. Apress, 2022.
8. Ziegler S. Micah. Mastering C# Async Programming: Building Scalable and Responsive Applications, 2021.
9. Anthony Williams. Hands-On Concurrency with C++: Practical Guide for High-Performance Multithreading, 2019.
10. Mark J. Price. C# 10 and .NET 6 – Modern Cross-Platform Development. Packt Publishing, 2021.
11. Joseph Albahari, Ben Albahari. C# 12 in a Nutshell: The Definitive Reference. O'Reilly Media, 2024.
12. Paul Deitel, Harvey Deitel, Chuti Prasertsith. C++ 20 for programmers. An Object-natural Approach, 2022
13. Jesse Liberty, Rodrigo Juarez, Maddy Montaquila. .NET MAUI for C# Developers: Build Cross-Platform Mobile and Desktop Applications. Packt Publishing, 2023.
14. Boiko, V. O. (2024). Method of imperative variables for search automation of textual content in unstructured documents. Radio Electronics, Computer Science, Control, (2), 117-125
15. Форкун Ю. Мартинюк В. Праворська Н. Лучицький О. Метрика диференційованої цикломатичної складності аналізу програмнго коду з використанням систем керування базами даних . Міжнародний науково-технічний журнал «Вимірювальна та обчислювальна техніка в технологічних процесах». №3 -2023- стор. 100-105

16. Праворська Н.І., Мартинюк В.В. Конструювання програмного забезпечення за допомогою синхронного підходу: основні процеси та інструменти для ефективної реалізації devops. Вісник Хмельницького національного університету, Том 1, №5, 2023 (325) – стор.182-192

## 13. INFORMATION RESOURCES

1. Electronic Library of the University. [Electronic resource]. – Access: http://library.khmnu.edu.ua/
2. Institutional Repository of Khmelnytskyi National University. [Electronic resource]. – Access: http://elar.khmnu.edu.ua/jspui/?locale=uk
3. Modular Learning Environment. [Electronic resource]. – Access: https://msn.khmnu.edu.ua/course/view.php?id=9344