



CROSS-PLATFORM PROGRAMMING

Working program of the academic discipline (Syllabus)

Details of the educational component	
Level of higher education <i>First (Bachelor)</i>	
Branch of knowledge	<i>12 Information Technologies</i>
Specialty	<i>121 Software Engineering</i>
Educational program	<i>Software Engineering of Multimedia and Information-Retrieval Systems</i>
Status of the educational component	<i>Elective</i>
Form of education	<i>Full-time</i>
A year of training	<i>3rd year of training, 6th semester</i>
Scope of the discipline	<i>Lectures: 36 hours, laboratory work: 18 hours, independent work: 66 hours.</i>
Semester control/ control measures	<i>Credit, modular control work, calendar control</i>
Schedule of classes	<i>According to the schedule for the autumn semester of the current academic year (rozklad.kpi.ua)</i>
Language of instructions	<i>English</i>
Information about head of the course / teachers	<i>Lecturer: Ph.D., associate professor, O. A. Tkachenko, Laboratory work: Ph.D., associate professor, O. A. Tkachenko, e-mail: aatokq@gmail.com</i>
Course location	<i>Google classroom. Access is given to registered students.</i>

Program of educational component

1. Description of the educational component, its purpose, subject of study and learning outcomes

Studying the discipline "Cross-platform Programming" allows students to develop the competencies necessary for solving practical tasks of professional activities related to the development of cross-platform software.

The purpose of studying the discipline "Cross-platform Programming" is the formation of students' abilities to independently design and develop cross-platform software.

The subject of the discipline "Cross-platform programming" is methods, means of modeling and development of cross-platform software.

*The study of the discipline "Cross-platform programming" contributes to the formation of students of **professional competences (PC)** necessary for solving practical tasks of professional activities related to the development, improvement and operation of software:*

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

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

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

***PC05** Ability to follow specifications, standards, rules and recommendations in the professional field during the life cycle processes implementation.*

***PC06** Ability to analyze, select and apply methods and tools to ensure information security (including cybersecurity).*

***PC11** Ability to implement phases and iterations of the life cycle of the software systems and information technology based on appropriate models and approaches to software development.*

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

***PC19** Ability to identify, analyze and document software requirements for multimedia and information retrieval systems.*

The study of the discipline "Cross-Platform Programming" contributes to the formation in students of the following **program learning outcomes (PLO)** according to the educational program:

PLO03 To know the software life cycle basic processes, phases and iterations.

PLO05 To know and apply relevant mathematical concepts, domain methods, system and object-oriented analysis and mathematical modeling for software development.

PLO06 Ability to select and use the appropriate task of software development methodology.

PLO09 To be able to use collecting, formulating and analyzing software requirements methods and tools.

PLO10 To conduct a pre-project survey of the subject area, system analysis of the design object.

PLO11 To select initial data for design, guided by formal methods of describing requirements and modeling.

PLO12 To apply effective approaches to software design in practice.

PLO15 To choose programming languages and development technologies to solve the problems of creating and maintaining software.

PLO17 To be able to apply methods of component software development.

PLO21 To know the tools, analyze, select, skillfully apply the information security (including cybersecurity) and data integrity means in accordance with the applied tasks and software systems.

PLO23 To be able to document and present the software development results.

2. Pre-requisites and post-requisites of the discipline (place in the structural and logical scheme of training according to the relevant educational program)

The successful study of the discipline "Cross-Platform Programming" is preceded by the study of the disciplines "Fundamentals of Programming", "Algorithms and Data Structures", "Fundamentals of Computer Systems and Networks", "Components of Software Engineering", "Databases" and "Programming" of the bachelor's training plan in the specialty F2 Software engineering.

The theoretical knowledge and practical skills obtained during the mastering of the discipline "Cross-Platform Programming" ensure the successful implementation of course projects and master's theses in the specialty F2 Software engineering.

3. Content of the academic discipline

The discipline "Cross-Platform Programming" involves the study of the following topics:

Topic 1. Requirements for cross-platform software. Basics of the java language.

Topic 2. Implementation of the principles of object-oriented programming in the Java language.

Topic 3. Working with classes and objects. Collections.

Topic 4. Development of a graphical user interface. Organization of multithreading.

Modular control work

Test

4. Educational materials and resources

Basic literature:

1. Software Development Studio: Guide to Laboratory Work/O. Tkachenko - K.: HAY, 2021. - 32 p.

2. C. Ullenboom. Java: The Comprehensive Guide to Java Programming for Professionals. - Rhein werk Computing, 2022. - 1128p.

3. Y. Daniel Liang. Introduction to Java Programming and Data Structures. - Pearson, 2020. - 1240p.

4. Aniche M. Effective Software Testing: A developer's guide. Shelter Island, USA: Manning, 2022. - 328p.

5. Dennis A., Wixom B., Tegarden D. Systems Analysis and Design : An Object-Oriented Approach with UML. Hoboken, USA : Wiley, 2020. 544 p.

6. <https://docs.oracle.com/javase/tutorial/>

Additional literature:

7. Threat Modeling. URL: <https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling> (accessed on: 01.06.2022).

8. Tarandach I., Coles M. J. Threat Modeling. A Practical Guide for Development Teams. Sebastopol: O'Reilly Media, 2020, 201 p.

9. Holt J, Perry S. SysML for Systems Engineering: A model-based approach (Computing and Networks). London, United Kingdom: The Institution of Engineering and Technology, 2019. 880 p.

10. MITER ATT&CK. URL: <https://attack.mitre.org/> (accessed on: 01.06.2022).

11. <https://www.cs.cmu.edu/afs/cs.cmu.edu/user/gchen/www/download/java/LearnJava.pdf>

12. <https://book.huihoo.com/goalkicker.com/JavaBook/JavaNotesForProfessionals.pdf>

Educational content

5. Methods of mastering an educational discipline (educational component)

№	Type of training session	Description of the training session
<i>Topic 1. Requirements for cross-platform software. Fundamentals of the Java language.</i>		
1	<i>Lecture 1. Course content, introduction to cross-platform programming</i>	<i>Overview of course content. The concept of cross-platform as a property of software. Varieties of cross-platform software. Life cycle of cross-platform software development. Ways and means of achieving cross-platform software. Task on self-study: item 6 No. 1.</i>
2	<i>Lecture 2. Basics of the JAVA programming language</i>	<i>General characteristics of the Java language. The history of the creation of the JAVA language. Differences between the JAVA and C++ languages. Vocabulary and syntax of the language. Data types. Definition of local variables. Console I/O. Structure of the program in the JAVA language. Assignment on self-study: item 6 No. 2.</i>
3	<i>Lecture 3. Language operations and operators. Working with arrays and strings.</i>	<i>Arithmetic and logical operations. Priorities of operations. Program execution management operators. Work with arrays. Classes and methods of working with strings. String, StringBuilder, StringBuffer, StringTokenizer classes. Assignment on self-study: item 6 No. 3.</i>
4	<i>Laboratory work 1. Basic Java language constructs. Working with character strings</i>	<i>Task: To get acquainted with the basic elements of the Java language and learn how to process data sets in the form of arrays. Learn to process numeric and text data sets. Assignment on self-study: item 6 No. 4.</i>
5	<i>Lecture 4. Exception handling.</i>	<i>The concept of exception. Types of exception. Sections try, catch and finally. The throw operator. Keyword throws. Features of exception handling in overridden methods. Multiple exception handler. Exception merge condition. Create your own exception. Task on self-study: item 6 No. 5.</i>
6	<i>Laboratory work 2. Working with arrays of class objects. Exception handling</i>	<i>Task: Learn to work with arrays of class objects. To study the concept of exceptions and the exception handling mechanism in Java. Creating your own exceptions. Assignment on self-study: item 6 No. 6.</i>
<i>Topic 2. Implementation of the principles of object-oriented programming in the Java language</i>		
7	<i>Lecture 5. Classes and objects. Implementation of the principles of object-oriented programming.</i>	<i>Classes and objects. Inheritance, encapsulation and polymorphism. Garbage collection. Methods and constructors. Built-in methods. Passing parameters to methods. Overloading and overriding methods and constructors. Assignment on self-study: item 6 No. 7.</i>
8	<i>Lecture 6. Packages. Abstract classes and interfaces</i>	<i>Packages. Abstract classes and methods, their features. Interfaces. Functional interfaces. Implementation of the Interface. Interface variables as constants. Implementation of polymorphism. Assignment on self-study: item 6 No. 8.</i>
9	<i>Laboratory work 3. Implementation of the principles of object-oriented programming</i>	<i>Task: to study the implementation of OOP principles (inheritance and polymorphism), as well as mechanisms of interaction between classes. Assignment on self-study: item 6 No. 9.</i>

10	Lecture 7. Execution threads: basic tools.	The concept of execution threads. Thread creation: Thread class and Runnable interface. Thread priorities. Threads synchronization. Interaction of threads. Mechanism of asynchronous execution. Assignment on self-study: item 6 No. 10.
11	Lecture 8. Input / output data streams	Streams for different data types. Byte and character I/O streams. File I/O streams. Buffered input/output of characters and strings. Streams for working with primitive data types. Object streams Task on self-study: item 6 No. 11.
12	Laboratory work 4. Input / output data streams	Task: To study classes and methods of file input/output in Java. To get acquainted with Java tools for processing texts and tables of the MS Office package. Task on self-study: item 6 No. 12.
13	Lecture 9. Work with databases	ODBC and JDBC. Creating a database and tables. Create a connection to the database. Create and execute queries. Processing of query results. Assignment on self-study: item 6 No. 13.
14	Lecture 10. Nested classes	Types of nested classes. Static class. Inner and local class. Anonymous local class. Using nested classes. Compiler and virtual machine processing of nested classes. Assignment on self-study: item 6 No. 14.
<i>Topic 3. Working with classes and objects. Collections.</i>		
15	Lecture 11. Generics	The concept of Generics. Generalized method. Constraints on the type parameter. Limitations of Generics. Generics and inheritance. Substitute type. Container classes. Basic interfaces of container classes. Enumeration class. Assignment on self-study: item 6 No. 15.
16	Lecture 12. Serialization	Serialization and deserialization. Serialization of complex objects. Interface java.io.Serializable. Static field serialVersionUID. ObjectOutputStream and ObjectInputStream classes. Serialization and inheritance. Custom serialization. Assignment on self-study: item 6 No. 16.
17	Lecture 13. Collections	The Collection interface and the Collections class. The Iterator interface. List interfaces: List, Queue, Deque. Set. Map, HashSet, LinkedHashSet and HashMap classes. TreeSet class. Implementations of thread-safe collections. Assignment on self-study: item 6 No. 17.
18	Lecture 14. Reflection	Purpose of reflection. Methods of obtaining an object of type Class. Getting information about data type modifiers. Getting information about a class element. Dynamic method invocation. Accessing private members of a data type. Class loaders. Class loading delegation model. Assignment on self-study: item 6 No. 18.
19	Laboratory work 5. Collections framework (Part 1)	Task: Familiarize yourself with the java collections framework, explore what affects the efficiency of collection processing and which collections to use depending on search, add, and delete operations. Task on self-study: item 6 No. 19.

20	Laboratory work 6. Collections framework (Part 2)	Task: Use the Java collections framework to store, access, and manipulate data, as well as to pass data from one method to another. Task on self-study: item 6 No. 20
<i>Topic 4. Graphical user interface development. Implementation of multithreading</i>		
21	Lecture 15. Graphical user interface development	Standard Java libraries for developing a graphical user interface. Advantages of the JavaFX library compared to AWT and Swing. Placement of components. Event handling. Assignment on self-study: item 6 No. 21.
22	Lecture 16. Multithreading: package concurrent (part 1)	Benefits of the java.util.concurrent package. Lock, Condition, Callable interfaces. Methods of the Lock interface. ReentrantLock class. Comparison of Lock and synchronized usage. Future class. Task on self-study: item 6 No. 22.
23	Laboratory work 7. Implementation of multithreading (Part 1)	Task: to study the mechanisms for creating and managing threads of execution in Java using the java.util.concurrent package Task on self-study: item 6 No. 23.
24	Lecture 17. Multithreading: package concurrent (part 2)	Thread pool and executors. Blocking queues. Synchronizing collection classes. General-purpose synchronizers. Atomic classes and variables. Task on self-study: item 6 No. 24.
25	Lecture 18. Stream API. Lambda- expression.	The concept of stream. Ways to create streams. Methods of working with streams. Lambda expression concept. Using streams with Lambda expression. Task on self-study: item 6 No. 25.
26	Laboratory work 8. Implementation of multithreading (Part 2)	Task: to study the mechanisms of creating and managing execution threads in Java using the java.util.concurrent package; explore ways of synchronizing threads; learn more about synchronization and blocking access to resources.. Task on self-study: item 6 No. 26.
27	Test	Task on self-study: item 6 No. 27.
<i>Modular test work</i>		

6. Independent work of a student/graduate student

The discipline "Cross-Platform Programming" is based on independent preparations for classroom classes on theoretical and practical topics.

Noz з/р	The name of the topic submitted for independent processing	Number of hours	literature
1	Preparation for the lecture 1	1	1-3; 8; 9
2	Preparation for the lecture 2	1	1; 2; 8; 9
3	Preparation for the lecture 3	1	1; 2; 8; 9
4	Preparation for laboratory work 1	4	1; 2; 5-9; 12-14
5	Preparation for the lecture 4	1	1; 2; 5-9; 12-14
6	Preparation for laboratory work 2	4	1; 2; 5-9; 12-14

7	Preparation for the lecture 5	2	1-3; 8; 9
8	Preparation for the lecture 6	2	1-3; 8; 9
9	Preparation for laboratory work 3	4	1-3; 8; 9
10	Preparation for the lecture 7	2	1-3; 8; 9
11	Preparation for the lecture 8	2	1-3; 8; 9
12	Preparation for laboratory work 4	4	1-3; 8; 9
13	Preparation for the lecture 9	2	1-3; 8; 9
14	Preparation for the lecture 10	2	1-3; 8; 9
15	Preparation for the lecture 11	2	1-3; 8; 9
16	Preparation for the lecture 12	2	1-3; 8; 9
17	Preparation for the lecture 13	2	1-3; 8; 9
18	Preparation for the lecture 14	2	1-3; 8; 9
19	Preparation for laboratory work 5	4	1-3; 8; 9
20	Preparation for laboratory work 6	4	1-3; 8; 9
21	Preparation for the lecture 15	2	1-3; 8-10
22	Preparation for the lecture 16	2	1-3; 8-10
23	Preparation for laboratory work 7	4	1-3; 8-10
24	Preparation for the lecture 17	2	3, 4, 11
25	Preparation for the lecture 18	2	3, 4, 11.
26	Preparation for laboratory work 8	4	3, 4, 11
27	Preparation for the test	2	1-14

7. Policy and control

Attending lectures is mandatory.

Attending laboratory work classes may be occasional and as needed for consultation/protection of laboratory work.

Rules of behavior in classes: activity, respect for those present, turning off phones.

Adherence to the policy of academic integrity.

Rules for the protection of laboratory work: the work must be performed in accordance with the assigned tasks and according to the option chosen by the student.

8. Types of control and rating system for evaluating learning outcomes

During the semester, students perform 8 laboratory works.

The maximum number of points for laboratory works 1-4 10 points each, for two works 5 and 6 - a total of 10 points, for two works 7-8 - a total of 10 points.

Points are awarded for:

- *quality of performance of the computer workshop: 0-8 points;*
- *timely presentation of work for defense: 0-2 points.*

Performance evaluation criteria:

7-8 points - the work is done with quality, in full;

5-6 points - the work is done qualitatively, in full, but has flaws;

3-4 points - the work is completed in full, but contains minor errors;

1-2 points - the work is completed in full, but contains significant errors;

0 points - the work is not completed in full.

Criteria for evaluating the timeliness of work submission for defense:

2 point – the work is submitted to the defense no later than 1 day after the specified deadline;

0 points – the work is submitted for defense more than 1 day late after the specified deadline.

The maximum number of points for the performance and defense of laboratory work:
10 points x 6 comp. practice = 60 points.

The assignment for the modular test consists of 40 test questions. Answer to each question is worth 1 point.

Evaluation criteria for test questions:

1 point - the answer is correct;

0 points - there is no answer or the answer is incorrect.

The maximum number of points for a modular control work:

1 point X 40 questions = 40 points.

The rating scale for the discipline is equal to:

$R = R_{lab. works} + R_{modular control work} = 60 \text{ points} + 40 \text{ points} = 100 \text{ points}.$

Calendar control: is conducted twice a semester as a monitoring of the current state of fulfillment of the syllabus requirements.

At the first certification (7th week), the student receives "credited" if his current rating is at least 15 points (50% of the maximum number of points a student can receive before the first certification).

At the second certification (13th week), the student receives "passed" if his current rating is at least 30 points (50% of the maximum number of points a student can receive before the second certification).

Semester control: assessment

Conditions for admission to semester control:

With a semester rating (RC) of at least 60 points and the completion of all computer lab work, the student receives credit "automatically" according to the table (Table of correspondence of rating points to grades on the university scale). Otherwise, he has to perform the final control work.

Completion and defense of the computer workshop is a necessary condition for admission to the credit control work.

If the student does not agree with the "automatic" grade, he can try to improve his grade by writing a credit test, while his points received for the semester are kept/

Table of correspondence of rating points to grades on the university scale:

Scores	Rating
100-95	Perfectly
94-85	Very good
84-75	Fine
74-65	Satisfactorily
64-60	Enough
Less 60	Unsatisfactorily
Admission conditions not met	Not allowed

9. Additional information on the discipline (educational component)

The list of questions submitted for semester control is given in Appendix 1.

Working program of the academic discipline (syllabus):

Compiled by Ph.D., Associate Professor O.A. Tkachenko.

Adopted by Computer Systems Software Department (protocol № 8 from 22.01.2025)

Approved by the Faculty Board of Methodology (protocol № 8 from 03.02.2025)

Appendix 1. List of questions submitted for semester control

- 1. The concept of cross-platform as a property of software.*
- 2. Varieties of cross-platform software.*
- 3. Means of achieving cross-platform software.*
- 4. Methods of determining functional requirements for cross-platform software.*
- 5. Implementation of the cross-platform nature of the java language. Differences of the Java language from C++.*
- 6. Application Development Environments (IDE).*
- 7. Primitive data types. Wrapper classes of data types.*
- 8. Literals, variables, built-in functions.*
- 9. Operators.*
- 10. Arrays.*
- 11. Tools for working with symbols and strings.*
- 12. Exception handling.*
- 13. Processing of text and binary files.*
- 14. Packages.*
- 15. Encapsulation. Classes and class objects. Access specifiers.*
- 16. Class constructors.*
- 17. Methods. Ways of passing parameters.*
- 18. Inheritance: means of implementation.*
- 19. Polymorphism: means of implementation.*
- 20. Overloading and overriding methods.*
- 21. Abstract methods and classes. Interfaces.*
- 22. Nested classes.*
- 23. Generalized types.*
- 24. Collections. Enumeration*
- 25. Serialization.*
- 26. Reflection.*
- 27. Creating a graphical user interface.*
- 28. Implementation of multithreading.*
- 29. Stream API.*
- 30. Lambda expressions.*