Appendix No. 1.5 to the Resolution No. 61/2025
of the Rector of the University of Rzeszów

SYLLABUS

REGARDING THE QUALIFICATION CYCLE FROM 2026 TO 2029
ACADEMIC YEAR 2026/2027.

1. BASIC COURSE/MODULE INFORMATION

Course/Module title	Object-oriented programming
Course/Module code *	
Faculty (name of the unit offering the field of study)	Faculty of Exact and Technical Sciences
Name of the unit running the course	Institute of Mathematics
Field of study	Mathematics
Qualification level	First-cycle studies (Bachelor's)
Profile	General academic
Study mode	Full-time
Year and semester of studies	Year 1, Semester 2
Course type	Major subject
Language of instruction	English
Coordinator	Paweł Pasteczka, PhD
Course instructor	Paweł Pasteczka, PhD

* - as agreed at the faculty

1.1.Learning format – number of hours and ECTS credits

Semester (no.)	Lectures	Classes	Laboratories	Seminars	Practical classes	Internships	others	**ECTS credits**
2	15		30					4

1.2. Course delivery methods
☒ traditional classroom-based instruction
☐ no distance learning / or optional use of distance learning methods and techniques

1.3. Course/Module assessment (exam, pass with a grade, pass without a grade)
Lecture – pass without a grade
Laboratory – pass with a grade

2. PREREQUISITES

Fundamentals of programming.

3. OBJECTIVES, LEARNING OUTCOMES, COURSE CONTENT, AND INSTRUCTIONAL METHODS

3.1. Course/Module objectives

O1	Preparation for solving problems (including mathematical ones) using computer science tools and using computers in mathematical classes.
O2	Familiarizing students with problem modeling and solving them using an object-oriented approach.
O3	Introduction to object-oriented programming issues – domain modeling and coding in a programming language.
O4	Familiarizing students with basic methods of designing algorithms, writing, running and testing programs. Analysis of calculation results.

3.2. COURSE/MODULE LEARNING OUTCOMES (TO BE COMPLETED BY THE COORDINATOR)

Learning Outcome	The description of the learning outcome defined for the course/module	Relation to the degree programme outcomes
LO_01	The student knows the basics of computational techniques and programming that can be used to support the work of a mathematician and understands the limitations arising from their use.	K_W05
LO_02	The student is able to recognize and specify a problem that can be solved algorithmically, student is able to arrange and analyze an algorithm consistent with the specification for a given problem and write it in an appropriate programming language, and then test the independently written computer program and make necessary corrections.	K_U14
LO_03	The student expresses their own opinions on theoretical and practical algorithmic issues concerning a given problem and formulates questions to understand the examined problem, and supplements their competencies if necessary.	K_K01

LO_04	The student is ready to present a critical attitude towards the received content, is aware of errors that may appear when arranging algorithms and writing in the programming languages used.	K_K02, K_K03

3.3. Course content (to be completed by the coordinator)

A. Lectures

Content outline
Object-oriented domain modeling: conceptual classes, relationships, attributes.
Objects and programming: attributes, methods, class hierarchy, visibility, encapsulation, UML.
Java: identifiers, literals, simple statements, imperative statements in Java, variables, primitive and reference types, arrays, objects, collections (selected classes implementing java.util.Collection), streams, functions (methods), parameters.
Encapsulation, constructors and initialization: declaration syntax, inheritance, virtual classes, access modifiers, polymorphism, generic types, classes parameterized by types, interfaces.
Exceptions: The java.lang.Throwable interface and its subclasses. Checked and unchecked exceptions. Try/catch.

B. Classes, laboratories, seminars, practical classes

Content outline
Object-oriented domain modeling: finding relations, attributes, class hierarchy, UML.
Introduction to Java: Compilation, Java virtual machine, packages, IDE, documentation, tools for concurrent work (CodeWithMe).
Java: Imperative instructions of the Java language, simple classes, variables, methods, checked and unchecked exceptions, generic types, collections.
Distributing responsibilities between classes: inheritance, interfaces, class hierarchy, packages, access modifiers.
Solving problems using object-oriented programming methods.

3.4. Methods of Instruction

Laboratory classes: problem solving, discussion, group work.
Lectures: problem-based lecture / multimedia presentation.

4. Assessment techniques and criteria

4.1 Methods of evaluating learning outcomes

Learning outcome	Methods of assessment of learning outcomes (e.g. test, oral exam, written exam, project, report, observation during classes)	Learning format (lectures, classes,...)
LO-01	project	lectures, lab
LO-02	project	lectures, lab
LO-03	project	lab
LO-04	project, observation during classes	lab

4.2 Course assessment criteria

The condition for passing the laboratory is obtaining at least 51 percent of points possible to obtain from projects. If the above conditions are not met, the student may send projects in the retake session. Activity during classes is rewarded with additional points.
Grading scale:

- 50–59% – satisfactory (3.0)
- 60–69% – satisfactory plus (3.5)
- 70–79% – good (4.0)
- 80–89% – good plus (4.5)
- 90–100% – very good (5.0)

5. Total student workload needed to achieve the intended learning outcomes – number of hours and ECTS credits

Activity	Number of hours
Course hours	45
Other contact hours involving the teacher (consultation hours, examinations)	5
Non-contact hours - student's own work (preparation for classes or examinations, projects, etc.)	50
Total number of hours	100
Total number of ECTS credits	4

* One ECTS point corresponds to 25-30 hours of total student workload

6. Internships related to the course/module

Number of hours	*Not applicable*
Internship regulations and procedures	*Not applicable*

7. Instructional materials

Compulsory literature: 1. Eckel, Bruce. Thinking in Java. 4th ed. Upper Saddle River, NJ: Prentice Hall, 2006. 2. Horstmann, Cay S. Core Java, Volume I—Fundamentals. 13th ed. Hoboken, NJ: Oracle Press, 2024. 3. Weisfeld, Matt. The Object-Oriented Thought Process. 5th ed. Boston, MA: Addison-Wesley, 2019.
Complementary literature: 1. Booch, Grady; Rumbaugh, James; Jacobson, Ivar. The Unified Modeling Language User Guide. 2nd ed. Boston, MA: Addison-Wesley, 2005. 2. Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. Design Patterns: Elements of Reusable Object-Oriented Software. 1st ed. Reading, MA: Addison-Wesley, 1994. 3. Shalloway, Alan; Trott, James R. Design Patterns Explained: A New Perspective on Object-Oriented Design. 2nd ed. Boston, MA: Addison-Wesley, 2004.

Approved by the Head of the Department or an authorised person