GDAŃSK UNIVERSITY OF TECHNOLOGY

Subject card

Subject name and code	Object-oriented programming languages III, PG_00060228		
Field of study	Technical Physics		
Date of commencement of studies	October 2023	Academic year of realisation of subject	2025/2026
Education level	first-cycle studies	Subject group	Optional subject group Subject group related to scientific research in the field of study
Mode of study	Full-time studies	Mode of delivery	at the university
Year of study	3	Language of instruction	English
Semester of study	5	ECTS credits	5.0
Learning profile	general academic profile	Assessment form	assessment
Conducting unit	Katedra Fizyki Teoretycznej i Informatyki Kwant. -> Faculty of Applied Physics and Mathematics -> Wydziały Politechniki Gdańskiej		
Name and surname of lecturer (lecturers)	Subject supervisor	dr hab. Jan Franz	
	Teachers		

Lesson types and methods of instruction

Lesson type	Lecture	Tutorial	Laboratory	Project	Seminar	SUM
Number of study hours	15.0	0.0	60.0	0.0	0.0	75

E-learning hours included: 0.0

eNauczanie source addresses:

Moodle ID: 1306 Obiektowe języki programowania III
https://enauczanie.pg.edu.pl/2025/course/view.php?id=1306

Learning activity and number of study hours

Learning activity	Participation in didactic classes included in study plan	Participation in consultation hours	Self-study	SUM
Number of study hours	75	5.0	45.0	125

Subject objectives

The aim of this course is to introduce students to object-oriented programming (OOP) in Java with a focus on applications in physics and applied informatics. Students will learn to design, implement, and test scientific software using modern Java tools, libraries, and design patterns. Emphasis is placed on writing robust, maintainable code and developing the skills needed for larger projects in research and technology.

Learning outcomes

Course outcome	Subject outcome	Method of verification
[K6_U03] Knows programming languages and can use basic software packages	is able to write programs in an object-oriented language, use project management tools, apply testing frameworks, and make use of selected scientific libraries to support problem-solving in physics and technology.	[SU1] Assessment of task fulfilment
[K6_W01] Understands the importance of physics and its applications in connection to civilization.	is able to model simple physical systems using object-oriented programming and reflect on how computational skills support the broader use of physics in science and technology.	[SW2] Assessment of knowledge contained in presentation
[K6_K01] Understands the need to learn and improve professional and personal competencies. Can inspire and organize other people’s learning process	is able to independently extend their knowledge of object-oriented programming, critically apply object-oriented tools and design patterns to scientific problems, and collaborate in ways that support and inspire the learning of others.	[SK5] Assessment of ability to solve problems that arise in practice
[K6_W05] Has knowledge of programming methodology and techniques, and the use of selected IT tools in physics and technology.	is able to apply object-oriented programming methodology and techniques, and make effective use of selected computational tools to solve problems in physics and technology.	[SW1] Assessment of factual knowledge

Subject contents	1. The Java Ecosystem & Project Setup Lecture: Java Virtual Machine (JVM), Java Development Kit (JDK), Integrated Development Environments (IDEs); project management with Maven and Gradle. Lab: Create first Maven project; run a simple physics-related program. 2. Classes, Objects & Testing Lecture: classes, fields, methods, constructors; introduction to unit testing with JUnit. Lab: Implement a Particle class and basic unit tests. 3. Primitive Types, Wrappers, Arrays & Efficient Java Matrix Library (EJML) Lecture: primitive types vs objects; arrays; wrapper classes; first look at EJML. Lab: Vector operations with arrays and EJML. 4. Inheritance and Interfaces Lecture: inheritance, overriding, abstract classes, interfaces. Lab: Class hierarchy for different particle types. 5. Exceptions and Robust Code Lecture: checked vs unchecked exceptions; error handling strategies. Lab: Robust file input/output (I/O) and simple simulation error handling. 6. Collections Framework Lecture: List, Set, Map; iterators; when to use collections. Lab: Store and analyze particle trajectories with collections. 7. Design Patterns I

Lecture: Factory, Singleton, Observer (with light Unified Modeling Language (UML) illustrations).

Lab: Implement a particle factory and observer for logging.

8. Generics & Collections in Practice

Lecture: generic classes and methods; collection implementations.

Lab: Generic containers for results; use sorted sets/maps.

9. Refactoring & Testing Practices

Lecture: cohesion, coupling, SOLID (Single responsibility, Openclosed, Liskov substitution, Interface segregation, Dependency inversion) principles; test-driven development (TDD).

Lab: Refactor earlier code and extend test coverage.

10. Lambda Expressions (Basics)

Lecture: functional interfaces, lambda syntax.

Lab: Apply lambdas to simple numerical transformations.

11. Streams and Applications of Lambdas

Lecture: Stream Application Programming Interface (API): map, filter, reduce; parallel streams.

Lab: Analyze simulation results with streams.

12. Scientific Libraries in Java

Lecture: EJML in more depth; Apache Commons Math; JavaScript Object Notation (JSON) and Extensible Markup Language (XML) parsing.

Lab: Solve linear systems and parse input from files.

13. Design Patterns II & Project Organization
Lecture: Strategy, Composite; modular project structure with Maven/Gradle.

	Lab: Apply Strategy pattern to select simulation models. 14. Student Project Presentations Lecture: Recap of object-oriented programming (OOP) in Java and integration of tools. Lab: Final project demos with short presentations. 15. Summary & Outlook Lecture: Future directions in programming (Java trends, concurrency, functional style, artificial intelligence (AI)assisted coding). Lab: Discussion of how course skills transfer to research projects.
Prerequisites and co-requisites	Object-oriented programming languages 1 and 2

Assessment methods and criteria

Subject passing criteria	Passing threshold	Percentage of the final grade
lab credit	50.0%	75.0%
final exam	50.0%	25.0%

Recommended reading

Basic literature	1. Joshua Bloch, Effective Java, 3rd Edition, Addison-Wesley, 2017 2. Raoul-Gabriel Urma, Mario Fusco, Alan Mycroft, Modern Java in Action, Manning Publications, 2018
Supplementary literature	1. Cay S. Horstmann, Core Java Volume 1 Fundamentals. 11Th edition, Prentice Hall, 2018 2. Cay S. Horstmann, Core Java Volume 2 Advanced Features. 11Th edition, Prentice Hall, 2018 3. Herbert Schildt, Java: The Complete Reference. 11Th edition, McGraw-Hill, 2019
eResources addresses	Basic https://docs.oracle.com/en/java/ - The Oracle Java Documentation site offers a comprehensive set of API references, tutorials, code samples, and developer guides spanning Java SE, Java EE, and related technologies to support building robust Java applications. Supplementary https://ejml.org - EJML (Efficient Java Matrix Library) is an open-source Java library for fast and flexible matrix computations, offering support for linear algebra operations such as decomposition, solving systems, and manipulation of dense and sparse matrices.

Example issues/ example questions/ tasks being completed	1.) You are given a single class Simulation that directly handles file input, data storage, calculations, and result output. Identify at least two problems with this design. Suggest a refactoring strategy using separate classes or packages. 2.) Programming Task: Radioactive Decay Simulation Implement a Particle class with attributes (id, half-life, state). Store particles in a collection and simulate decay step by step using random numbers. Handle invalid input with exceptions. Include at least one JUnit test. Print the number of undecayed particles after each step.
Work placement	Not applicable

Document generated electronically. Does not require a seal or signature.