GDAŃSK UNIVERSITY OF TECHNOLOGY

Subject card

Subject name and code	Object-oriented programming languages III, PG_00064060		
Field of study	Technical Physics		
Date of commencement of studies	October 2026	Academic year of realisation of subject	2028/2029
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	Department of Theoretical Physics and Quantum Computing -> Faculty of Applied Physics and Mathematics -> Faculties of Gdańsk University of Technology		
Name and surname of lecturer (lecturers)	Subject supervisor	dr hab. Jan Franz	
	Teachers		

Lesson types

Lesson type	Lecture	Tutorial	Laboratory	Project	Seminar	SUM
Number of study hours	15.0	0.0	45.0	0.0	0.0	60

E-learning hours included: 0.0

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	60	5.0	60.0	125

Subject objectives

The aim of the course is to introduce students to object-oriented programming in Java, with particular emphasis on applications in physics and applied computer science. Students will learn to design, implement, and test scientific software using modern tools, libraries, and design patterns. Emphasis will be placed on creating reliable, readable, and maintainable code, as well as on developing skills useful in larger research and technological projects.

Learning outcomes

Course outcome	Subject outcome	Method of verification
[K6_K01] demonstrates readiness for continuous learning and updating knowledge in physics and related fields, critically evaluating it and recognising its importance in solving practical and theoretical problems.	is prepared to independently and continuously expand their knowledge of object-oriented programming, critically evaluate current tools and design patterns, and recognize their importance in solving scientific and practical problems.	[SK5] Assessment of ability to solve problems that arise in practice
[K6_W05] has knowledge of programming methodologies and techniques, as well as the use of selected IT tools in physics and engineering.	has knowledge of the methods and techniques of object-oriented programming and knows the principles of using selected IT tools to solve problems in the field of physics and technology.	[SW1] Assessment of factual knowledge
[K6_W01] demonstrates an understanding of the civilisational significance of physics and its applications.	understands the importance of modeling simple physical systems using object-oriented programming and knows the role of computational methods in supporting applications of physics in science, technology, and civilizational development.	[SW3] Assessment of knowledge contained in written work and projects
[K6_U03] possesses programming skills in a selected language and the ability to use selected software packages.	is able to create programs in a selected object-oriented language, use tools supporting software project management, apply testing frameworks, and use selected scientific libraries to solve problems in the field of physics and technology.	[SU1] Assessment of task fulfilment

Subject contents	Course content – lecture **1. The Java Ecosystem & Project Setup** Java Virtual Machine (JVM), Java Development Kit (JDK), Integrated Development Environments (IDEs); project management with Maven and Gradle. **2. Classes, Objects & Testing** Classes, fields, methods, constructors; introduction to unit testing with JUnit. **3. Primitive Types, Wrappers, Arrays & Efficient Java Matrix Library (EJML)** Primitive types vs objects; arrays; wrapper classes; first look at EJML. **4. Inheritance and Interfaces** Inheritance, overriding, abstract classes, interfaces. **5. Exceptions and Robust Code** Checked vs unchecked exceptions; error handling strategies. **6. Collections Framework** List, Set, Map; iterators; when to use collections. **7. Design Patterns I** Factory, Singleton, Observer (with light Unified Modeling Language (UML) illustrations). **8. Generics & Collections in Practice** Generic classes and methods; collection implementations. **9. Refactoring & Testing Practices** Cohesion, coupling, SOLID (Single responsibility, Openclosed, Liskov substitution, Interface segregation, Dependency inversion) principles; test-driven development (TDD). **10. Lambda Expressions (Basics)** Functional interfaces, lambda syntax. **11. Streams and Applications of Lambdas** Stream Application Programming Interface (API): map, filter, reduce; parallel streams. **12. Scientific Libraries in Java** EJML in more depth; Apache Commons Math; JavaScript Object Notation (JSON) and Extensible Markup Language (XML) parsing.

13. Design Patterns II & Project Organization

Strategy, Composite; modular project structure with Maven/Gradle.

14. Student Project Presentations

Recap of object-oriented programming (OOP) in Java and integration of tools.

15. Summary & Outlook

Future directions in programming (Java trends, concurrency, functional style, artificial intelligence (AI assisted coding).

Course content – laboratory

1. The Java Ecosystem & Project Setup

Create first Maven project; run a simple physics-related program.

2. Classes, Objects & Testing

Implement a Particle class and basic unit tests.

3. Primitive Types, Wrappers, Arrays & Efficient Java Matrix Library (EJML)

Vector operations with arrays and EJML.

4. Inheritance and Interfaces

Class hierarchy for different particle types.

5. Exceptions and Robust Code

Robust file input/output (I/O) and simple simulation error handling.

6. Collections Framework

Store and analyze particle trajectories with collections.

7. Design Patterns I

Implement a particle factory and observer for logging.

8. Generics & Collections in Practice

Generic containers for results; use sorted sets/maps.

9. Refactoring & Testing Practices

Refactor earlier code and extend test coverage.

---

10. Lambda Expressions (Basics)

Apply lambdas to simple numerical transformations.

11. Streams and Applications of Lambdas

Analyze simulation results with streams.

12. Scientific Libraries in Java

Solve linear systems and parse input from files.

13. Design Patterns II & Project Organization

Apply Strategy pattern to select simulation models.

14. Student Project Presentations

Final project demos with short presentations.

15. Summary & Outlook

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	

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.

Practical activites within the subject

Not applicable

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