VIDZEMES AUGSTSKOLA

FACULTY OF ENGINEERING
STUDY COURSE DESCRIPTION

Course Title:	**Basics of Programming I**			
Course code (LAIS):	*DatZB018*			
Study programme:	**Information Technologies**			
Level of Study programme:	☐	1st level professional higher education		
	☒	Professional Bachelor		
	☐	Professional Master		
	☐	Academic Master		
	☐	PhD level		
Type of Study programme:	☒	Compulsory course (Part A)		
	☐	Professional specialization courses (Part B, compulsory)		
	☐	Professional specialization optional courses (Part B, optional)		
	☐	Elective courses (Part C)		
Course Workload:	**Credits[1]**	Academic hours	Contact hours	Independent work hours
Full time:	6	150	60	90
Part time:	6	150	18	132
Course Author/ Tutor:	**Medjon Hysenaj**			
	Associate visiting professor, Ph.D.			
	e-mail: medjon.hysenaj@unishk.edu.al			
Study Form:	Full time studies, part time studies			
Study year, semester:	1st year, 1st semester			
Language:	English			
Prerequisites for the Course:	-			
Course Summary:	The aim of this course is to introduce students to structured problem-solving techniques, fundamental principles of algorithms, and object-oriented programming (OOP). This foundational course provides students with the skills required to analyze and solve programming challenges, fostering the ability to write efficient and maintainable code in Java. Students will gain a comprehensive understanding of programming constructs, including data types, control structures, arrays, classes, and file processing. By completing this course, students will be equipped to continue learning more advanced programming languages and concepts in future studies. By the end of the course, students will be able to: • Understand the basics of Java programming and object-oriented paradigms. • Utilize simple data types, variables, and arrays effectively. • Perform operations using operators and control statements (e.g., loops, conditions). • Develop and use classes, objects, and dynamic collections. • Handle data input, error management, and file processing. • Follow principles of good-quality programming. • Create and debug simple Java applications.			
Assessment:	Exam			
Requirements for Credits:	Final grade is calculated of: Class Participation and Interactive Tasks: 10% Homework Assignments: 20% Projetcs Coding Task: 40% Final Exam: 30% Exam is calculated as THEORY x 0.4 + CODE_READ x 0.2 + CODING x 0.4			

[1] *Eiropas kredītpunktu pārneses un uzkrāšanas sistēmas studiju uzskaites vienība*

	Note: A minimum of 50% in the final exam is required to pass the course.	
Abiding by the Academic Ethics	Students must abide by the academic and research ethics, Vidzeme University of Applied Sciences Ethics Regulations, incl.: - study papers must be independently developed; - the study work should reference all statements, ideas and data used that have been authored by someone else; - appropriate data acquisition methods should be used in the acquisition of data, the research ethics must be respected, empirical data must be collected independently and cannot be distorted or falsified; - the examination must be carried out by the student independently, without the use of supporting materials and/or consultations with other students, unless the lecturer states otherwise. In the event of non-compliance with the academic and research ethics, punishment is imposed in accordance with the ViA Ethics Regulations and the study course must be re-taken, unless the punishment is exmatriculation.	
Learning Outcomes; the evaluation methods and criteria	**Learning Outcomes**	**The evaluation methods and criteria**
	Knowledge	
	Basic principles of Java programming, including the implementation of simple data types, variables, arrays, and their applications across different programming languages. Core concepts of methods, clean programming, and defensive programming techniques to prevent code vulnerabilities and ensure maintainability. Fundamental understanding of objects, classes, inheritance, and encapsulation, enabling students to design reusable and scalable solutions. Overview of error handling and file processing, focusing on managing program flow and data storage in real-world scenarios. Familiarity with programming tools and environments, such as integrated development environments (IDEs), to streamline coding tasks.	Code reading and writing tasks, lectures, practical assignments, and group discussions.
	Skills	
	Ability to create method-based programs to solve fundamental programming problems, using both procedural and object-oriented paradigms.	Practical tasks, hands-on coding assignments, discussions, tests, coding projects, and interactive lectures.
	Proficiency in writing clean, modular, and well-documented code that follows industry best practices and design patterns.	Practical tasks, hands-on coding assignments, discussions, tests, coding projects, and interactive lectures.
	Capability to develop programs that handle input validation, error management, and file processing, ensuring robustness and reliability.	Practical tasks, hands-on coding assignments, discussions, tests, coding projects, and interactive lectures.
	Competence in implementing dynamic collections and algorithms to solve data-driven problems effectively.	Practical tasks, hands-on coding assignments, discussions, tests, coding projects, and interactive lectures.
	Competency	
	Independently analyze, design, and implement method-based programs to address both simple and complex data-driven challenges.	Practical tasks, discussions, tests, coding projects, course work, and the final exam.
	Apply object-oriented principles such as encapsulation, inheritance, and	Practical tasks, discussions, tests, coding projects, course work, and the

	polymorphism to create maintainable and extensible software solutions.	final exam.
	Solve real-world programming problems using structured approaches, integrating algorithms and data structures effectively.	Practical tasks, discussions, tests, coding projects, course work, and the final exam.
Course Compulsory literature:	**All the compulsory and additional literature is available in the course materials on the school's web platform and external sources (Youtube)**	
Course additional literature:	-	
Course confirmation date:	08.12.2022	
Date of course description update:	09.01.2025	

Study Course Plan for Full Time Students:

Date	Theme	Academic hours		Study Form/ Organization of independent work of students and task description
		Contact hours	**Independent work hours**	
The date is specified before the implementation of the course				
	Introduction to Programming, Java, OOP, and the Purpose of Programming Languages	5	7	Lecture, individual research on programming evolution
	Simple Data Types, Variables, Arrays, and Differences Across Programming Languages	5	7	Lecture, hands-on coding task to implement variable operations
	Operations and operators	5	7	Lecture, coding exercise on arithmetic and logical operations
	Methods	5	7	Lecture, create modular code by designing functions
	Control Keywords: break, continue, return	5	7	Lecture, interactive problem-solving with real-life scenarios
	String operations	5	7	Lecture, project task to manipulate and analyze strings
	Data Input, Validation, and Error Handling	5	7	Lecture, practical debugging and error-trapping task
	Classes and objects	5	7	Lecture, code implementation of a class-based program
	Dynamic collections (e.g., Lists, Maps)	4	7	Lecture, exercise on managing and iterating over collections
	File processing	4	7	Lecture, task to create programs reading/writing files
	Good Quality Programming Principles (Clean Code)	4	7	Lecture, refactor existing code to meet quality standards
	Presentation of course work	4	13	Seminar, feedback on course projects
	Exam	4		Theoretical discussion, written and coding exam
Hours total:		***60***	***90***	

Study Course Plan for Part Time Students:

Date	Theme	Academic hours	Study Form/

		Contact hours	Independent work hours	Organization of independent work of students and task description
The date is specified before the implementation of the course				
	Introduction to Programming, Java, OOP, and the Purpose of Programming Languages	2	16	Lecture, individual research on programming evolution
	Simple Data Types, Variables, Arrays, and Differences Across Programming Languages	2	16	Lecture, hands-on coding task to implement variable operations
	Operations and operators	2	10	Lecture, coding exercise on arithmetic and logical operations
	Methods	2	10	Lecture, create modular code by designing functions
	Control Keywords: break, continue, return	2	10	Lecture, interactive problem-solving with real-life scenarios
	String operations	1	10	Lecture, project task to manipulate and analyze strings
	Data Input, Validation, and Error Handling	1	10	Lecture, practical debugging and error-trapping task
	Classes and objects	1	10	Lecture, code implementation of a class-based program
	Dynamic collections (e.g., Lists, Maps)	1	10	Lecture, exercise on managing and iterating over collections
	File processing	1	10	Lecture, task to create programs reading/writing files
	Good Quality Programming Principles (Clean Code)	1	10	Lecture, refactor existing code to meet quality standards
	Presentation of course work	1	10	Seminar, feedback on course projects
	Exam	1		Theoretical discussion, written and coding exam
Hours total:		***18***	***132***	