Module/Course Syllabus
Program: COMPUTER SCIENCE
Full-time master degree program

Course:	**Design Patterns and Clean Code Techniques**
Type of the course:	*elective*
Course code:	I2S3.13
Year:	II
Semester:	3
Form of the degree program:	**full-time**
Form of classes and number of hours per semester:	60
Lecture	30
Classes	0
Laboratory	30
Project	0
Number of ECTS credits:	**3**
Form of assessment:	**exam**
Course language:	english

Course objective (CO)	
CO1	Introduce students to design patterns used in programming
CO2	Introduce students to clean code techniques
CO3	Improving the skills of students in the field of code writing techniques and familiarization with the methods of code optimization

Prerequisites in terms of knowledge, skills and other competencies	
1	Object-oriented programming
2	Algotithms and data structures
3	Java programming language

Learning outcomes (LO)	
	In terms of knowledge:
LO 1	Knowledge of categories and their associated design patterns
LO 2	Knowledge of clean code writing techniques
LO 3	Knowledge of techniques for building a clean application architecture
	In terms of skills:
LO 3	The ability to select and apply design patterns depending on the met programming problem
LO 4	Ability to use SOLID techniques and other methods of achieving clean code while building an application
LO 5	The ability to improve the source code of a program in order to improve its readability, extensibility and easier maintenance
	In terms of social competence:
LO 6	Awareness of the need to write clean code and optimize the existing code to improve the quality of the software delivered to the end user

LO 7	Awareness of the impact of software writing techniques on the work of a programming team, in particular on the possibility of mutual understanding of the written code, group development of shared code, and the impact on the ease of its development and maintenance

Course content	
Form of classes – lectures (L)	
	Course content
L1	Introduction. The need to write clean code and apply design patterns
L2	Creational design patterns
L3	Structural design patterns
L4	Behavioral design patterns
L5	SOLID principles
L6	Methods of using naming, comments, formatting and functions
L7	Clean code: classes, objects and data structures
L8	Clean code: error handling, constraints, and unit testing
L9	Concurrency and clean code
L10	Functional programming
L11	Aspect programming
L12	Refactoring of written code
L13	Succesive refinement
L14	Application of tests in software development
Form of classes – laboratories (Lab)	
	Course content
Lab1	Application of creational design patterns
Lab2	Application of structural design patterns
Lab3	Application of behavioral design patterns
Lab4	Applying the principles of single responsibility and open-closed
Lab5	Applying the principles of Liskov substitution, dependency inversion and interface segregation
Lab6	Applying the principles of clean code in programming
Lab7	Applying the functional programming
Lab8	Applying the aspect programming
Lab9	Refactoring of application code
Lab10	Application of tests in software development

Didactic methods	
1	*Informational lecture*
2	*Exercises in laboratory*
3	*Method of programming with use of a computer*

Assessment methods and criteria		
Assessm	**Assessment method description**	**Passing**

ent method symbol		threshold
A1	*Test exam*	**51%**
A2	*Evaluation of laboratory exercises*	**51%**

Required textbooks and other course materials	
1	Cooper J.W., Java. Wzorce projektowe., Helion, Gliwice, 2001
2	Martin R. C., Clean Code. A Handbook of Agile Software Craftsmanship., Prentice Hall, 2008
3	Martin R. C., Clean Architecture. A Craftsman's Guide to Software Structure and Design, Prentice Hall, 2018
4	Martin R. C., Zwinne wytwarzanie oprogramowania. Najlepsze zasady, wzorce i praktyki., Helion, 2015
5	Freeman E., Robson E., Wzorce projektowe. Rusz głową! Tworzenie rozszerzalnego i łatwego w utrzymaniu oprogramowania obiektowego. Wydanie II, Helion, 2021
Recommended textbooks and other course materials	
1	Martin R. C., The Clean Coder, A Code of Conduct for Professional Programmers, Prentice Hall, 2011
2	Horstman C. S., Java. Techniki zaawansowane. Wydanie 11, Helion, 2020
3	Vernon V., DDD dla architektów oprogramowania., Helion, 2016

Student workload	
Form of activity	**Average number of hours to complete the activity**
Contact hours with the lecturer, including:	60
participation in lectures	30
participation in laboratories	30
Student's own work, including:	15
preparation for the exam	10
preparation for the laboratory	5
Total student workload	75
Total number of ECTS credits	3

Learning outcomes matrix					
Learning outcome	**Reference to learning outcomes defined for the masters program**	**Course objectives**	**Course content**	**Didactic methods**	**Assessment methods**
LO 1	I2A_W01 + I2A_W02 + I2A_W07 +++ I2A_W08 +++	CO1	L1 - L4	1	A1
LO 2	I2A_W04 ++ I2A_W07 +++ I2A_W08 +++	CO1, CO2	L5 - L9	1	A1
LO 3	I2A_W04 ++ I2A_W07 +++	CO2, CO3	L9 - L14	1	A1

	I2A_W08 +++				
LO 4	I2A_U07 +++ I2A_U13 +++ I2A_U15 +++	CO1	Lab1 - Lab3	2, 3	A2
LO 5	I2A_U13 +++ I2A_U15 +++	CO1, CO2	Lab5 - Lab7	2, 3	A2
LO 6	I2A_U09 +++ I2A_U13 +++ I2A_U15 +++ I2A_U16 ++	CO2, CO3	Lab7 - Lab10	2, 3	A2
LO 7	I2A_K01 +++	CO1	L1	1	A1
LO 8	I2A_K02 +++	CO3	L1, Lab1-Lab10	1, 2, 3	A1, A2

The author of the program:	Dr. Piotr Kopniak, Ph.D. (Eng.)
E-mail address:	p.kopniak@pollub.pl
Organizational unit:	Department of Computer Science