

# ARCHITECTURE AND DESIGN OF SOFTWARE (PART 1)

## COURSE SYLLABUS

<b>Code and name of specialty</b>	121 – Software Engineering	<b>Institute / faculty</b>	Faculty of Computer Science and Software Engineering
<b>Program name</b>	“Software Engineering”	<b>Department</b>	Software Engineering and Management Information Technologies
<b>Type of program</b>	Educational and Professional	<b>Language of instruction</b>	Ukrainian, English

## LECTURER

**Full name, e-mail**

**DVUKHHLAVOV Dmytro,  
Dmytro.Dvukhhlavov@kspi.edu.ua**



**Ph.D., Associate Professor, Associate Professor of Software Engineering and Information Technology Management. Has prepared and published more than 40 publications, 1 article in publications indexed in Scopus, 2 textbooks, 2 guidelines for making practice tasks.**  
**h-index = 3, i10-index = 0 y Google Academy-<https://scholar.google.com/citations?user=OAZyFg8AAAAJ&hl=ru>; identifier ORCID-<https://orcid.org/0000-0002-3361-3212>).**  
**Leading lecturer of the courses:** Advanced Java programming course (*Bachelors*) (*Ukrainian*), Java-based web applications (*Bachelors*) (*Ukrainian*), Architecture and Design of Software (part 1) (*Bachelors*) (*English and Ukrainian*), Architecture and Design of Software (part 2) (*Bachelors*) (*English and Ukrainian*).

## GENERAL DESCRIPTION OF THE COURSE

<b>Summary</b>	<p>The discipline «Architecture and Design of Software (Part 1)» is study compulsory discipline in the cycle of professional training of preparation level "bachelor" on specialty 121 “Software Engineering”. It is taught in the fifth semester in the amount of 150 hours (5 ECTS credits), in particular: lectures - 32 hours, laboratory classes - 32 hours, independent work - 86 hours. Final control – exam.</p> <p>The presentation of the discipline ensures the readiness of the student to participate in addressing the issues of justification of the choice of software architecture, determining the composition and structure of its components, taking into account the established quality requirements.</p>
<b>Course objectives</b>	<p>The course objective to deepen knowledge about the software life cycle and standards governing the implementation of its stages, providing knowledge about software architecture, architectural styles of modern software, known software design templates, as well as frameworks describing software architecture, as well as developing skills in developing software solutions using some design templates, as well as developing skills in creating software models based on design templates in Enterprise Architect.</p>
<b>Types of classes and control</b>	<p>Lectures, laboratory classes, consultations. Final control – exam.</p>

<b>Terms</b>	5						
<b>Student workload (credits) / Type of course</b>	5 / Mandatory	<b>Lectures (hours)</b>	32	<b>Workshops (hours)</b>	32	<b>Independent work (hours)</b>	86
<b>Program competencies</b>	<p>GC 2. Ability to apply knowledge in practical situations.</p> <p>GC 05. Ability to learn and master modern knowledge.</p> <p>GC 06. Ability to search, process and analyze information from various sources.</p> <p>PC13. Ability to identify, classify and formulate software requirements.</p> <p>PC14. Ability to participate in software design, including modelling (formal description) of its structure, behavior and functioning processes.</p> <p>PC15. Ability to develop architectures, modules and components of software systems.</p> <p>PC17. Ability to adhere to specifications, standards, rules and recommendations in the professional field in the implementation of life cycle processes.</p> <p>PC19. Knowledge of information data models, the ability to create software for data storage, retrieval and processing.</p> <p>PC23. Ability to implement phases and iterations of the life cycle of software systems and information technology based on appropriate models and approaches to software development.</p> <p>PC24. Ability to carry out the system integration process, apply change management standards and procedures to maintain the integrity, overall functionality and reliability of the software.</p> <p>PC25. Ability to reasonably select and master software development and maintenance tools.</p>						
<b>Learning outcomes</b>	<p>PO01. Analyze, purposefully search for and select the necessary information and reference resources and knowledge to solve professional problems, taking into account modern advances in science and technology.</p> <p>PO03. Know the basic processes, phases and iterations of the software life cycle.</p> <p>PO06. Ability to select and use the appropriate task methodology of software development.</p> <p>PO07. Know and apply in practice the fundamental concepts, paradigms and basic principles of operation of language, tools and computing software engineering.</p> <p>PO10. Conduct a pre-project survey of the subject area, systematic analysis of the design object.</p> <p>PO11. Choose source data for design, guided by formal methods of describing requirements and modelling.</p> <p>PO12. Put effective approaches to software design into practice.</p> <p>PO13. Know and apply methods of algorithm development, software design and data and knowledge structures.</p> <p>PO14. Put into practice the tools of domain analysis, design, testing, visualization, measurement and documentation of software.</p> <p>PO15. Being motivated to choose programming languages and development technologies to solve problems of software design and maintenance.</p> <p>PO16. Have the skills of team development, approval, design and release of all types of software documentation.</p> <p>PO17. Be able to apply methods of component software development.</p> <p>PO19. Know and be able to apply methods of software verification and validation.</p> <p>PO20. Know the approaches to evaluating and ensuring the quality of software</p> <p>PO23. Be able to document and present the results of software development.</p>						

<b>Teaching and learning methods</b>	The main method of teaching during lectures is the explanatory-illustrative method. To intensify cognitive activity, students' speeches and organization of discussions on certain issues of lectures are provided. Execution of laboratory works involves the creation by students of a solution model in the form of a set of UML diagrams and program code for the implementation of typical program functionality based on known design templates. For the implementation of the student determines his own subject area. There are no ready-made algorithms for solving, which encourages the manifestation of creative activity of students.
<b>Forms of assessment (continuous assessment CAS, final assessment FAS)</b>	Assimilation of the theory is tested in the form of a rapid survey during lectures (CAS), a survey or automated testing at the beginning of laboratory work (CAS). Control of mastering the material for self-study involves the preparation and defense of abstracts on individual topics (2 abstracts) (CAS). The level of practical skills is assessed by the results of laboratory work (CAS). Final / semester control is carried out during the exam, which involves written experience on two theoretical questions and the creation of code according to the task (FAS).

### ASSESSMENT AND GRADING

Ranges of points corresponding to grades	Total score (points) for all types of learning activities	ECTS grading scale	The national grading scale	<b>Allocation of grade points</b>	Assimilation of theory (topics of independent work)	30 points
	90-100	A	excellent		Working out the tasks of the laboratory workshop	40 points
	82-89	B	good			
	74-81	C			Lab#2-7	6 points for each
	64-73	D	satisfactory		Passing exam	30 points
	60-63	E			Summary	100 points
	35-59	FX	Unsatisfactory (with the exam retake option)			
	0-34	F	Unsatisfactory (with mandatory repetition of the course)			

<b>Course</b>	
---------------	--

<b>policy</b>	Students are required to attend classes according to the schedule. In the absence of a student at the lecture, he works out a syllabus of lectures before the next lesson. Participation in laboratory work involves the need to repeat the lecture material and self-study of recommended sources. At the beginning of the laboratory there is an experience of students for the materials of lectures and independent work. Performing laboratory tasks requires prior preparation and advance processing of all necessary materials for productive discussions during the lesson and their operational implementation. All laboratory work is required to obtain a final grade in the discipline. An important element of training is the need to adhere to the schedule of presentation of laboratory results and abstracts. For delay in execution without an officially confirmed reason, the score is reduced.
---------------	---

### COURSE STRUCTURE AND CONTENT

<b>Topic 1</b>	The concept of "software architecture". The main provisions of the IEEE 1471 standard and other regulatory documents of software engineering. Frameworks for describing software architecture (software) ("4 + 1", RM-ODP, SOMF). (6 years)	<b>Laboratory class 1</b>	Study of the Enterprise Architect installation process and its application to create basic UML diagrams (8 hours) (PT59)	<b>Independent work</b>	Installing Enterprise Architect Creating projects and models in Enterprise Architect. Enterprise Architect toolbars. Create class, sequence, and activity diagrams in Enterprise Architect.
<b>Topic 2</b>	Requirements for modern software architecture. The essence of using templates to ensure modern software requirements. Classification of design patterns (Design pattern by GoF). Typical description of design templates. Detailed description of popular design templates GRASP responsibility allocation templates. (14 years)	<b>Laboratory class 2</b>	Research of features of realization of SINGLETON and FACTORY METHOD templates (4 hours) (PT63)		Assignment of design templates. Typical situations, advantages and disadvantages of using design templates. Principles of application of GRASP responsibilities distribution templates. Overview of implementations of the application of responsibilities distribution templates.
<b>Topic 3</b>	Architectural styles. Definition of architectural style. An overview of modern architectural styles. (8 years)	<b>Laboratory class 3</b>	Research of features of realization of the TEMPLATE METHOD template (4 hours) (PT63)		Detailed description of architectural styles. Analysis of the implementation of style in well-known software products.
<b>Topic 4</b>	Programming paradigms (4 hours)	<b>Laboratory class 4</b>	Research of features of realization of the PROXY template (4 hours) (PT63)		Modern views on approaches to software design
		<b>Laboratory class 5</b>	Research of features of realization of COMMAND and MEMENTO templates (4 hours) (PT63)		
		<b>Laboratory class 6</b>	Research of features of realization of a design template COMPOSITE (4 hours) (PT63)		

**Laboratory class 7**

Research of features of realization of the CHAIN OF RESPONSIBILITY template (4 hours) (PT63)

**RECOMMENDED READING****Compulsory**

- 1 Sommerville Ian. Software Engineering (6th ed.). Retrieved from [https://www.academia.edu/6826193/Ian\\_Sommerville\\_Software\\_Engineering\\_6th\\_Edition](https://www.academia.edu/6826193/Ian_Sommerville_Software_Engineering_6th_Edition).
- 2 Guide to Software Engineering Body of Knowledge (SWEBOOK), version 3.0. (2014). IEEE Computer Society.
- 3 Лаврищева, К. М. (2013). Software Engineering комп'ютерних систем. Парадигми, технології та CASE засоби програмування. Київ :Наукова. Думка.
- 4 Bass Len, Clements Paul, Kazman Rick. (2012). Software Architecture in Practice (SEI Series in Software Engineering). (3rd ed.). Addison-Wesley Professional.
- 5 Табунщик, Г. В., Каплієнко, Т. І., Петрова, О. А. (2016). Проектування та моделювання програмного забезпечення сучасних інформаційних систем. Запоріжжя.
- 6 Freeman Eric, Robson Elisabeth. (2020). Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software (2nd ed.). O'Reilly Media.
- 7 Larman Craig. (2005). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. (3rd ed.). Pearson.
- 8 Clements Paul, Bachmann Felix, Bass Len, Garlan David, Ivers James, Little Reed... Stafford Judith (2010). Documenting Software Architectures: Views and Beyond. (2nd ed.). Addison-Wesley Professional.

**Recommended****Стандарти**

- 1 ISO/IEC/IEEE 12207 Systems and software engineering—Software life cycle processes.
- 2 IEEE 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology.
- 3 ISO/IEC 9126-1. Software engineering—Product quality. Part 1: Quality model.
- 4 IEEE 1471. Recommended Practice for Architectural Description of Software-Intensive Systems.
- 5 ISO/IEC/IEEE 42010. Systems and software engineering-Architecture description.

**Ресурси інтернет**

- 1 Retrieved from: <https://refactoring.guru/ua/design-patterns>.

**ACADEMIC INTEGRITY**

The student must adhere to the Code of Ethics of Academic Relations and Integrity of NTU "KhPI": to show discipline, politeness, friendliness, honesty, responsibility

The content of this syllabus is consistent with the course program.