КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

КАФЕДРА ІНТЕЛЕКТУАЛЬНИХ ПРОГРАМНИХ СИСТЕМ

«ЗАТВЕРДЖУЮ»

Заступник декана
з навчальної роботи

_____ Кашпур О.Ф.

«____»_____2019 року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

ПРОГРАМНА ІНЖЕНЕРІЯ

для студентів

галузь знань	**12 Інформаційні технології**
спеціальність	**121 Інженерія програмного забезпечення**
освітній рівень	**бакалавр**
освітня програма	**Програмна інженерія**
вид дисципліни	**обов'язкова**

Форма навчання	**денна**
Навчальний рік	**2022/2023**
Семестр	**7**
Кількість кредитів ECTS	**4**
Мова викладання, навчання та оцінювання	**українська**
Форма заключного контролю	**іспит**

Викладач: **к.ф.-м.н., доцент Ченцов О.І.** (лекції, лабораторні заняття)
к.ф.-м.н., асистент Терлецький Д.О. (лабораторні заняття)

Пролонговано: на 20__/20__ н.р. _____(_____) «__»___ 20__р.

на 20__/20__ н.р. _____(_____) «__»___ 20__р.

КИЇВ – 2019

Розробник: Ченцов Олексій Ілліч, к. ф.-м. н., доцент, доцент кафедри інтелектуальних програмних систем.

ЗАТВЕРДЖЕНО

Завідувач кафедри інтелектуальних програмних систем

_____ О.І. Провотар

Протокол № ___ від «____» _____ 2019 р.

Схвалено науково-методичною комісією факультету комп'ютерних наук та кібернетики

Протокол № ___ від «___» _____2019 року

Голова науково-методичної комісії _____ Л.Л. Омельчук

«_____» _____ 20___ року

Затверджено вченою радою факультету комп'ютерних наук та кібернетики

Протокол від «____» _____ 2019 року №___

Голова вченої ради факультету _____ А.В. Анісімов

1. Мета дисципліни – розвинути у студента розуміння концепцій, принципів та методів розробки якісних програмних систем різного ступеня складності, на практичному прикладі підготувати студента до виконання програмних проектів на виробництві.

2. Попередні вимоги до опанування або вибору навчальної дисципліни:

1. **Знати:** основні стилі програмування, технології програмування на мові програмування Java (або подібній до неї), основні принципи спадкування.
2. **Вміти:** створювати програми на мові програмування Java (чи подібній до неї), будувати основні діаграми UML і розуміти їх призначення, створювати web-орієнтовані системи на мові програмування Java (чи подібній до неї).
3. **Володіти:** прийомами об'єктно-орієнтованого програмування.

3. Анотація навчальної дисципліни. Навчальна дисципліна «Програмна інженерія» є складовою освітньо-професійної програми підготовки фахівців за першим (бакалаврським) рівнем вищої освіти у галузі знань 12 Інформаційні технології за спеціальністю 121 Інженерія програмного забезпечення в рамках освітньої програми «Програмна інженерія».

Дана дисципліна є обов'язковою навчальною дисципліною в рамках освітньої програми «Програмна інженерія». Викладається у 7 семестрі в **обсязі – 120 год. (4 кредити ECTS)** зокрема: лекції – 26 год., лабораторні – 28 год., самостійна робота – 64 год., консультації – 2 год. У курсі передбачено 2 змістові частини та 1 контрольна робота. Завершується дисципліна – **іспитом.**

В результаті вивчення навчальної дисципліни студент повинен

знати: принципи розробки програмного забезпечення, моделі та складові процесу розробки, підходи до моделювання програмних систем, способи оцінки та підвищення якості програмного забезпечення.

вміти: виконувати програмні проекти різного рівня складності у складі команди, організовувати та управляти проектами середнього розміру, оформлювати технічну супроводжуючу документацію, підсумкові та проміжні звіти про виконання проекту

4. Завдання (навчальні цілі). Основними завданнями дисципліни «Програмна інженерія» є набуття знань, умінь та навичок (компетентностей) на рівні новітніх досягнень у програмній інженерії відповідно до освітньої кваліфікації бакалавр з інженерії програмного забезпечення. Зокрема, розвивати:

- Здатність до абстрактного мислення, аналізу та синтезу (ЗК01).
- Здатність застосовувати знання у практичних ситуаціях (ЗК02).
- Здатність спілкуватися державною мовою як усно, так і письмово (ЗК03).
- Здатність вчитися і оволодівати сучасними знаннями (ЗК05).
- Здатність до пошуку, оброблення та аналізу інформації з різних джерел (ЗК06).
- Здатність працювати в команді (ЗК07).
- Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування (СК02).
- Здатність розробляти архітектури, модулі та компоненти програмних систем (СК03).
- Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу (СК05).

- Здатність застосовувати і розвивати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення (СК08).
- Здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення (СК11).

5. Результати навчання за дисципліною.

Результат навчання (1. знати; 2. вміти; 3. комунікація; 4. автономність та відповідальність)		**Форми (та/або методи і технології) викладання і навчання**	**Методи оцінювання та пороговий критерій оцінювання (за необхідності)**	**Відсоток у підсумковій оцінці з дисципліни**
Код	**Результат навчання**			
РН1.1	Знати принципи розробки програмного забезпечення, моделі та складові процесу розробки, підходи до моделювання програмних систем, способи оцінки та підвищення якості програмного забезпечення.	Лекції	Контрольна робота, 60% балів, іспит	40%
РН2.1	Вміти виконувати програмні проекти різного рівня складності у складі команди, організовувати та управляти проектами середнього розміру, оформлювати технічну супроводжуючу документацію, підсумкові та проміжні звіти про виконання проекту	Лабораторні заняття, самостійна робота	Захист лабораторної роботи, іспит	40%
РН3.1	Обґрунтовувати власний погляд на задачу, спілкуватися з колегами з питань проектування та розробки програм	Лабораторні заняття	Захист лабораторної роботи	10%
РН4.1	Організовувати свою самостійну роботу для досягнення результату	Самостійна робота	Захист лабораторної роботи	10%

6. Співвідношення результатів навчання дисципліни із програмними результатами навчання.

Результати навчання дисципліни / Програмні результати навчання	РН1.1	РН2.1	РН3.1	РН4.1
ПРН03. Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення.	+			
ПРН16. Мати навички командної розробки, погодження, оформлення і випуску всіх видів програмної документації.		+	+	+
ПРН22. Знати та вміти застосовувати методи та засоби управління проектами.		+		+

7. Схема формування оцінки.

7.1 Форми оцінювання студентів:

Семестрове оцінювання:

1. Контрольна робота: РН 1.1 – **20 балів/12 балів**.
2. Лабораторні роботи 1-6: РН2.1, РН 3.1, РН4.1 – **50 балів/30 балів**.

Підсумкове оцінювання (у формі іспиту):

- Максимальна кількість балів які можуть бути отримані студентом: 30 балів.
- Результати навчання які будуть оцінюватись: РН1.1, РН2.1.
- Форма проведення і види завдань: письмова робота.
- Види завдань: 3 письмових завдань.

Студент **не допускається** до іспиту, якщо під час семестру набрав **менше ніж 30 балів**.

7.2 Організація оцінювання:

Терміни проведення форм оцінювання:

1. Контрольна робота: до 14 тижня семестру.
2. Лабораторна робота 1: до 3 тижня семестру.
3. Лабораторна робота 2: до 6 тижня семестру.
4. Лабораторна робота 3: до 6 тижня семестру.
5. Лабораторна робота 4: до 11 тижня семестру.
6. Лабораторна робота 5: до 14 тижня семестру.
7. Лабораторна робота 6: до 14 тижня семестру.

Якщо студент з поважних причин, які підтверджено документально, був відсутній при написанні контрольної роботи, він має право на одне перескладання з можливістю отримання максимальної кількості балів. Термін перескладання визначається викладачем.

У разі неякісного виконання лабораторної роботи, викладач має право не зарахувати лабораторну роботу, або знизити за неї бали.

Студент має право здавати лабораторні роботи після закінчення визначеного для них терміну, але з втратою одного балу за кожен тиждень, який пройшов з моменту закінчення терміну її здачі.

7.3 Шкала відповідності оцінок

Відмінно / Excellent	90-100
Добре / Good	75-89
Задовільно / Satisfactory	60-74
Незадовільно / Fail	0-59

8. Структура навчальної дисципліни. Тематичний план лекцій і лабораторних занять.

№ лекції	Назва теми/лекції	Кількість годин		
		Лекції	Лабораторні заняття	Самост. робота
Частина 1. Основні процеси розробки програмного забезпечення та управління ними				
1	**Тема 1.** Вступ до курсу.	2	2	6
2	**Тема 2.** Процес розробки програмного забезпечення.	2	2	4
3	**Тема 3.** Управління програмними проектами.	2	2	4
4	**Тема 4**. Гнучка методологія розробки.	2	2	4
5	**Тема 5**. Інженерія вимог до програмного забезпечення. **Лекція 5**. Вимоги до програмного забезпечення.	2	2	4
6	**Тема 5**. Інженерія вимог до програмного забезпечення. **Лекція 6.** Інженерія вимог до ПЗ.	2	2	4
7	**Тема 6**. Моделювання систем.	2	2	4
8	**Тема 7**. Архітектурне проектування.	2	2	6
9	**Тема 8**. Проектування та реалізація.	2		4
Контрольна робота			2	
Всього по частині 1		18	18	40
Частина 2. Тестування та еволюція програмного забезпечення				
10	**Тема 8**. Тестування програмного забезпечення. **Лекція 10.** Процес тестування програмного забезпечення.	2	2	4
11	**Тема 8**. Тестування програмного забезпечення. **Лекція 11.** Методи тестування програмних компонентів.	2	2	4
12	**Тема 8**. Тестування програмного забезпечення. **Лекція 12.** Методи тестування програмних систем.	2	2	4
13	**Тема 9**. Еволюція програмного	2	2	8

	забезпечення у ході його експлуатації.			
14	**Тема 10**. Реверсна інженерія програмного забезпечення		2	4
Всього по частині 2		8	10	24
ВСЬОГО		**26**	**28**	**64**

Загальний обсяг – **120** год., в тому числі:
Лекцій **– 26** год.
Лабораторні роботи – **28** год.
Самостійна робота – **64** год.
Консультації – **2** год.

Теми, винесені на самостійне вивчення:

1. Кодекс етики програміста.
2. Програмний інструментарій підтримки виконання проектів.
3. Рекомендації щодо оформлення специфікації вимог до програмного забезпечення.
4. Виявлення вимог на основі етнографічного дослідження.
5. Специфікація вимог в екстремальному програмуванні.
6. Архітектурне проектування багаторівневих застосунків. Проектування рівня доступу до даних. Проектування бізнес-рівня та рівня представлення.
7. Реверсна інженерія програмного забезпечення

Умови лабораторних робіт:

- **Лабораторна робота 1:** Вибір програмного проекту, формування команди, підготовка статуту.
- **Лабораторна робота 2:** План проекту.
- **Лабораторна робота 3**: Специфікація вимог до програмної системи.
- **Лабораторна робота 4**: Звіт з проектування програмної системи.
- **Лабораторна робота 5**: Звіт з тестування програмної системи.
- **Лабораторна робота 6**: Індивідуальний звіт з проекту.

Деталізовані умови лабораторних робіт розміщено за посиланням:
https://docs.google.com/document/d/1ak-sI-QetalHsbkWzl3Yoba3AvEGkc5W7aBLEJZy_54

9. Рекомендовані джерела:

Основні:

1. *Sommerville I.* Software Engineering, 10th ed. / Sommerville I. — Addison-Wesley / Pearson Education Limited, 2015. — 816 p.
2. *Соммервилл И.* Инженерия программного обеспечения, 6 изд. / Соммервилл И. — М.: "Вильямс", 2002. — 624 с.
3. *Pressman R.* Software Engineering: A Practitioner's Approach, 8th ed. / Maxim B., Pressman R. — McGraw-Hill, 2014. — 976p.

Додаткові:

1. *McConnell S.* Code Complete: A Practical Handbook of Software Construction, 2nd Edition. / McConnell S. — Microsoft Press, 2004. — 960 p.
2. *Hunt A.* The Pragmatic Programmer: From Journeyman to Master / Hunt A., Thomas D. — Addison-Wesley Longman Publishing, 1999. — 320p.
3. *Ghezzi C.* Fundamentals of Software Engineering, 2nd ed. / Ghezzi C., Jazayeri M., Mandriol D. — Prentice Hall, 2003. — 604p.
4. *Лавріщева К.М.* Програмна інженерія. — Київ, 2008. — 319с.
5. *Липаев В.* Программная инженерия. Методологические основы. — М.: Теис, 2006. — 608с.
6. *Brooks F.* The Mythical Man-Month, Anniversary Edition (2nd ed.) — Addison-Wesley Professional, 1995. — 336p.
7. Guide to Software engineering body of knowledge – 2004 Version / Exec. eds: Alain Abran, James W. Moore. — IEEE Computer Society, 2004. — 200p. — http://www.computer.org/portal /web/swebok/htmlformat
8. Software engineering: Report of a conference sponsored by the NATO Science Committee / Eds.: Peter Naur, Brian Randell. — Scientific Affairs Division, NATO, 1969. — 231p.
9. *Wirth Niklaus*. A Brief History of Software Engineering // IEEE Annals of the History of Computing. — v. 30, No. 3, 2008. — pp. 32–39.
10. *Brooks F*. No silver bullet // IEEE Computer. — vol. 20, no. 4, April 1987. — pp. 10–19.
11. ISO/IEC 12207:2008, Systems and software engineering – Software life cycle processes. — ISO, 2008. — 123p.
12. Software Engineering Code of Ethics and Professional Practice (Version 5.2). — http://www.acm.org/about/se-code
13. *Boehm B.* A Spiral Model of Software Development and Enhancement. — ACM SIGSOFT Software Engineering Notes", "ACM". — v. 11(4), August 1986. — pp. 14–24.
14. 830-1998, IEEE Recommended Practice for Software Requirements Specifications. — IEEE, 1998. — 31p.
15. 1233-1998, IEEE Guide for Developing System Requirements Specifications. — IEEE, 1998.
16. *Viller S.* Ethnographically informed analysis for software engineers. / Viller S., Sommerville I. // Int. J. Human-Computer Studies. — 53 (1), 2000. — pp.169–196.
17. *Mellor S.* Executable UML: A Foundation for Model-Driven Architecture. / Mellor S., Balcer M. — Addison-Wesley, 2002. — 416p.
18. Object Management Group: Semantics of a foundational subset for executable UML models (fUML), v1.0. — OMG, 2011. — 404p. — http://www.omg.org/spec/FUML/.
19. Object Management Group: Concrete Syntax For UML Action Language (Action Language For Foundational UML – ALF). — OMG, 2010. — 425p. — http://www.omg.org/spec/ALF/.
20. *Schwaber K.*. — The Scrum Guide: The Definitive Guide to Scrum – The Rules of the game. / Schwaber K., Sutherland J. — 2011. — 17p. — http://www.scrum.org/Scrum-Guides.
21. *Kruchten P.* The 4+1 View Model of Architecture // IEEE Software. — vol. 12, no. 6, November 1995. — pp. 42–50.

22. Microsoft Application Architecture Guide, 2nd ed. / Microsoft Patterns & Practices Team. — Microsoft Press, 2009. — 496p. — http://msdn.microsoft.com/en-us/library/dd673617.aspx
23. Руководство Microsoft по проектированию архитектуры приложений, 2-е изд. / Microsoft Patterns & Practices Team. — Microsoft Press, 2010. — 529p. — http://apparchguide.ms/
24. *Mills H.* Cleanroom Software Engineering / *Mills H., Dyer M., Linger R.* // IEEE Software. — v. 4, no. 5, September 1987. — pp.19–25.
25. *Chikofsky E.* Reverse engineering and design recovery: a taxonomy / Chikofsky E., Cross J. // IEEE Software. — vol. 7, no. 1, January 1990. — pp.13–17.
26. *Rugaber S.* White paper on reverse engineering. / Rugaber S. — 1994. — 11p. — http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.199.9825
27. *Leffingwell D.* Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. / Leffingwell D. — Addison-Wesley, 2011. — 518p.

10. Додаткові ресурси.

Приклади звітів, переліки запитань, що виносяться на іспит, допоміжні матеріали розміщено за посиланнями:
https://drive.google.com/drive/folders/0B-BUpwNPP_9JcHVHdEUxTXJjXzQ
https://drive.google.com/drive/folders/1em5oEvsrHaEOuXN0yAMLRDTLho3_khCS