Міністерство освіти і науки України
Національний університет водного господарства та природокористування

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра обчислювальної техніки

05/03-02М

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з навчальної дисципліни
«Патерни проектування»
для здобувачів вищої освіти
другого (магістерського) рівня
за освітньо-професійною програмою
«Комп'ютерна інженерія»
спеціальності 123 «Комп'ютерна інженерія»
денної та заочної форми навчання

Рекомендовано
науково-методичною радою
з якості ННІКІТІ
Протокол № 4 від 30.03.2026 р.

Рівне – 2026

Методичні вказівки до лабораторних робіт з навчальної дисципліни «Патерни проектування» для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Комп'ютерна інженерія» спеціальності 123 «Комп'ютерна інженерія» денної та заочної форми навчання. [Електронне видання] / Бойчура М. В., Іванчук Н. В., Шатна А. В., Шатний С. В. – Рівне : НУВГП, 2026. – 38 с.

Укладач: Бойчура М. В., к.т.н., доцент кафедри обчислювальної техніки;
Іванчук Н. В., к.т.н., доцент кафедри комп'ютерних наук та прикладної математики;
Шатна А. В., старший викладач кафедри обчислювальної техніки;
Шатний С. В., к.т.н., доцент кафедри обчислювальної техніки.

Відповідальний за випуск: Сидор А. І., к.т.н., завідувач кафедри обчислювальної техніки.

Керівник групи забезпечення
спеціальності
123 «Комп'ютерна інженерія» Шатний С. В.

ЗМІСТ

Вступ

Розробка сучасного програмного забезпечення вимагає від фахівців не лише знання синтаксису мов програмування, але й володіння архітектурними підходами, що забезпечують гнучкість, розширюваність та надійність коду. Ключовим інструментом для вирішення типових проблем проєктування є патерни (шаблони) проєктування – перевірені часом рішення, що описують структуру та взаємодію об'єктів і класів.

Дані методичні вказівки розроблені для здобувачів вищої освіти другого (магістерського) рівня спеціальності 123 «Комп'ютерна інженерія». Курс орієнтований на поглиблення знань об'єктно-орієнтованого програмування та формування практичних навичок застосування патернів у середовищі .NET.

Структура лабораторного практикуму побудована за принципом «від простого до складного» і охоплює такі етапи:

- адаптація до інструментарію: плавний перехід від мови C++ до C#, вивчення базових синтаксичних конструкцій, роботи з класами, методами та принципів Coding Conventions;
- основи проєктування: робота з UML-діаграмами класів, глибоке розуміння принципів ООП (інкапсуляція, наслідування, поліморфізм) та моделювання архітектури додатків (вбудованих, розподілених, мобільних тощо);
- вивчення патернів: практична реалізація породжуючих (Singleton, Factory Method), структурних (Facade, Decorator, Adapter, Proxy) та поведінкових (Memento, Observer) патернів проєктування;
- командна робота та рефакторинг: розвиток навичок роботи з «чужим» проєктом, інтеграція структурних

патернів у існуючий код одногрупників та компонування єдиного програмного продукту.

Особлива увага в курсі приділяється не лише написанню коду, але й вмінню аналізувати архітектурні рішення, обґрунтовувати вибір тих чи інших патернів та працювати в команді, дотримуючись принципів чистого коду.

Виконання лабораторних робіт дозволить студентам сформувати компетентності, необхідні для проєктування складних програмних систем, їх підтримки та масштабування, що є критично важливим для професійної діяльності інженера-програміста.

Лабораторна робота №1
Перехід від C++ до C#
(максимум 4 бали)

Мета:

- початок плавного переходу від мови C++ до мови C#;
- набуття (повторення) базових навичок відлагодження програм.

Завдання

1. Обчислити значення виразу згідно варіанту (**2 бали**). Усі змінні у правих частинах виразів вважаємо заданими у самому коді програми. У методі Main в усіх випадках використовувати лише синтаксис мови C++, окрім наступних випадків:

а) при необхідності доступу до класу Math (методи Math.Sin(…), Math.Pow(…) тощо);

б) при виведені даних на екран (методи Console.Write(…), Console.WriteLine(…)).

2. Продемонструвати викладачу кілька базових навичок відлагодження (**2 бали**) коду програми (наприклад, кроку з обходом, із використанням точок зупинки, вікон «Контрольні значення 1» і «Локальні» тощо).

Варіанти

№	Завдання 1	Завдання 2
1	$\gamma = \dfrac{\omega}{x^{5,3}} \dfrac{e^{lg\lvert x\rvert -1} + \sqrt{\lvert (\omega x - 3) - 1\rvert}}{\arccos^2 \omega^{-1} + arcctg \dfrac{x}{\pi}}$	$y = \begin{cases} \sqrt[3]{lgx + \ln x^2}, & x > 1 \\ e^{-x} + 1, & x \le 1 \end{cases}$
2	$p = 17,4 + \dfrac{(-1)^k \cdot e^{-kx} + tg^4(kx-3)}{\sqrt{\lvert \sin kx \rvert} + \sqrt[4]{\lvert \cos kx \rvert + \lvert \cos x \rvert}}$	$y = \begin{cases} \ln \lvert x^3 \rvert, & 1 < x < 10 \\ e^{-x}, & x \le 0 \\ x^2 - 0,75, & x \ge 10 \end{cases}$

3	$z = \left(\dfrac{\lvert x-1\rvert + e^{-x} - 12,34}{lg\sqrt{\lvert x\rvert} - \cos x^3} \right)^{-0,4}$	$y = \begin{cases} -\cos^2(x-\pi), & -\pi < x < \dfrac{\pi}{4} \\ \sqrt{\lvert x+1\rvert}, & \dfrac{\pi}{4} \le x \le 1 \\ \dfrac{1}{x-1}, & x > 1 \end{cases}$
4	$y = \dfrac{\sqrt[3]{n-1} \cdot \ln 2^n - tg\dfrac{n+1}{n^3}}{\arccos 0,85 + \ln n} + 2,75$	$y = \begin{cases} x\sqrt{x-5,4}, & 0 \le x < 2 \\ arctgx^2, & 2 \le x < 8 \\ lg\lvert x-7,8\rvert, & x \ge 8 \end{cases}$
5	$g = \dfrac{\sin^3 x^2 - \cos^4(x-1)^2}{arctg\lvert x+2,6\rvert + \sqrt[3]{\ln\lvert x\rvert}} - 6$	$y = \begin{cases} \cos^3 x^2 - \sin^3 x^2, & x \le 0 \\ \lvert -arctg\dfrac{x+\pi}{x^2}\rvert, & 0 < x \le 1 \\ 1,4 + x\ln x, & 1 < x < 3,2 \end{cases}$
6	$r = \dfrac{ctg\dfrac{x+k}{k+1} - \sqrt{\lvert \ln x - \ln k\rvert + 1,3}}{\sin^4 c^{-k} + \arcsin^2 \dfrac{1}{k}}$	$y = \begin{cases} e^{2x^2} + 1, & x < 0, x \ne -1 \\ \sqrt{lgx - \ln x^2}, & 1 < x \le 55 \\ \dfrac{x - \ln x}{\ln x}, & x > 55 \end{cases}$
7	$S = \log_c \dfrac{c \cdot e^{-2,5c+x} + arctg^2\lvert c-x\rvert}{(-1)^{-2,5c} + \sqrt{\lvert \ln\lvert x\rvert + lg\lvert c\rvert\rvert}}$	$y = \begin{cases} 2x\sqrt[3]{x^2+z^2}, & 1 < x < 20 \\ arctg(x-z), & 0 < x \le 1 \\ e^{x+z}, & x \le 0 \end{cases}$
8	$k = \sqrt{\dfrac{\sin^2(m\,y) + \cos^2\dfrac{y}{m} + 0,64}{lg\lvert m\,y\rvert + \ln\lvert m^2 - y^3\rvert + e^{y-m}}}$	$y = \begin{cases} e^{-x} + \lvert x^2 - 1\rvert, & x > 1 \\ lg\sqrt{\lvert 1-x\rvert}, & -\pi < x \le 1 \\ 2x - \sin^2 x, & 0 < x \le 1 \end{cases}$
9	$i = \arcsin^3 \left(\dfrac{2,5x^{-2} + \sqrt{\lvert x-3\rvert^3}}{tg\dfrac{5,4}{x} + \ln(x+0,3)^5} + 1 \right)$	$y = \begin{cases} 15, & x < -10 \\ \dfrac{x^2 - 1,2}{x+4}, & 4 < \lvert x\rvert < 10 \\ x, & x \ge 10 \end{cases}$

10	$t = (m-y)^{-\lvert x+1\rvert}\cdot \dfrac{\ln\lvert m-y\rvert+\cos^3(my)+0{,}01}{\sqrt{\lvert m+y\rvert^3}+17{,}14\,lg\dfrac{\pi}{3}}$	$y=\begin{cases} e^{-\lvert x\rvert}, & 1\le x \\ \lg\sqrt{1-x^2}, & \lvert x\rvert<1 \\ arctgx, & x\le -1 \end{cases}$
11	$q=\dfrac{e^{-3{,}5\lvert z\rvert+\sqrt{14}}}{tgz+ctg\dfrac{\pi}{14}+z}-\dfrac{arctg^3(z-1)}{\ln z^4+\ln z^6}$	$y=\begin{cases} -\sqrt[4]{cx}, & c>9, x<-1 \\ ctg\dfrac{c}{x}, & c<0, -1<x\le 1 \\ \ln\lvert c^2-x^2\rvert, & c>0, x<c \end{cases}$
12	$\varphi=\arccos 0{,}13x+ +\dfrac{e^{0{,}6x-1}-\sqrt{(x+6{,}1)^3}}{\ln\lvert\dfrac{\pi}{16}-x\rvert+tg^2x^3}$	$y=\begin{cases} 2^{x-1}+2{,}71, & \pi\le x<8{,}5 \\ \sqrt{\lvert\pi-x\rvert}, & 8{,}5<x<\pi \\ 2{,}7, & 8{,}5\le\lvert x\rvert \end{cases}$
13	$i=c^2\log_2 x^4\dfrac{21(x-0{,}5)^2-\cos\dfrac{\pi}{x}}{\sqrt[3]{\lvert\ln x^{-1}\rvert-\sqrt{\lvert x+1\rvert}}}$	$y=\begin{cases} 0, & x\le -10 \\ ctg\dfrac{x-1}{e}, & -10<x\le 0 \\ \ln x, & x\ge 10 \\ (\sqrt{x})^3, & 0<x<10 \end{cases}$
14	$\alpha= =(-2)^{\frac{k+1}{2}}\,arcctg\dfrac{e^{kx-5{,}1}+\cos^2 kx^3}{\ln\lvert kx+1\rvert-lg\lvert x\rvert}$	$y=\begin{cases} e^{-x^2}, & \lvert x\rvert<1 \\ \pi\ln^2 x, & x>1 \\ 10^{-3}, & \lvert x\rvert=1 \end{cases}$
15	$b=(\beta+z)^{-e}- -\dfrac{\cos^3 z^4+tg^4(\beta-1)^{-2}-0{,}03}{6{,}51+\sqrt{\lvert\beta-z\rvert+lg\lvert z-1\rvert}}$	$y=\begin{cases} 3x^{-3}, & 1<x\le 125 \\ 13{,}44, & x>125 \\ arcctg^2 x+1, & -15{,}4\le x\le 1 \\ 1, & x<-15{,}4 \end{cases}$
16	$f=217{,}5e^{-x+0{,}77}+ +21\left\lvert\dfrac{\gamma\cdot x^c-e^{-x}+0{,}1}{\sqrt{\lvert\sin^3(x-1)+\cos\gamma\rvert}}\right\rvert$	$y=\begin{cases} \arcsin(-x^2+1), & 0\le x\le 3 \\ lg^2(2x)+4{,}4, & 0<x \\ -e^8, & -3<x<0 \end{cases}$

17	$\beta = \dfrac{e^{-x^2} - e^{\sqrt{\lvert x-0,5\rvert}} + 12,47x}{lg\lvert x+1\rvert - \ln^3\lvert 2^x - 1\rvert + ctg\dfrac{x^2-1}{e}}$	$y = \begin{cases} \sin e^x - 2, & \lvert x\rvert \le 4 \\ \dfrac{x^2-1,2}{x+4}, & 10 > \lvert x\rvert > 4 \\ x, & x \ge 10 \\ 1,5, & x \le -10 \end{cases}$
18	$j = (-1)^{lgm}\dfrac{\sqrt{\arcsin m^{-3} + arctg\dfrac{\pi}{3}}}{\log_x(13,4x^{-m} + \lvert\dfrac{\pi}{5} - x\rvert)}$	$y = \begin{cases} \sqrt{x}, & x > 0 \\ 2 - x^2, & x \le 0 \end{cases}$
19	$l = e^{kx-0,5} \cdot \left(\dfrac{lg\lvert k+x\rvert - \sqrt{\sin^4 x}}{\lvert arctg\dfrac{x+1}{x-k} + \dfrac{\pi}{10}\ln\pi\rvert + 1} + 2\right)$	$y = \begin{cases} e^{-x}, & 1 < x < 2 \\ x^e + 1, & 2 \le x \le 5 \\ 1, & x < 1 \text{ або } x > 5 \end{cases}$
20	$a = y\sqrt[3]{y+0,01} + \dfrac{\arccos\dfrac{\pi+y}{3} - \ln\lvert\dfrac{\pi}{y}\rvert}{\sqrt{\lvert\pi - y\rvert + \sin^2\pi y + 1}}$	$y = \begin{cases} \lvert\sin x - x\rvert, & x \ge 2z, \lvert x\rvert > 1 \\ \sin^2(x+z), & z \le x, \lvert z\rvert > 1 \end{cases}$
21	$d = \dfrac{arcctgx^{-0,5e}}{\omega} - \left\lvert\dfrac{\sqrt{\sin^2 x^3} - \sqrt[3]{\cos^5 x^2}}{lgx^2 + \ln\lvert\dfrac{2}{x}\rvert}\right\rvert$	$y = \begin{cases} (-1)^{3x-1}x^2, & \lvert x\rvert \ge 5 \\ \ln x^{-1}, & 0 < x < 1 \\ \cos x - 11, & 1 < \lvert x\rvert < 5 \end{cases}$
22	$h = \dfrac{\pi}{8}\sin^2\left(\dfrac{x-\pi}{8}\right) - \dfrac{e^{-\pi} + \pi^{-e} + 0,15}{\log_x\lvert\dfrac{\pi}{e} + 1\rvert - tg^2 x}$	$g = \begin{cases} \dfrac{\pi}{8}\sin^2\dfrac{x-y}{3}, & 1 \le \lvert x\rvert,\ \lvert y\rvert \le 1 \\ y^{-e}, & 1 < y,\ x < 1 \\ e^{-x}, & \lvert x\rvert < 1,\ \lvert y\rvert \le 1 \\ 0,15, & y < -1,\ x > 1 \end{cases}$

23	$n = \dfrac{\sin^2(\alpha + x)}{0,5 + e^{-\alpha x}} \cdot \sqrt{\dfrac{\ln\lvert \alpha + 2\rvert - lg\lvert x - 2\rvert}{arctg(\sin^2 x + tg^3 \alpha)}}$	$y = \begin{cases} \sqrt{\sin^2(x-1)}, & -2 \le x \le 2 \\ e^{-x}, & x > 6 \\ 4,4 lg^3 \lvert x\rvert, & 2 < \lvert x\rvert < 6 \\ \ln x^2, & x < -6 \end{cases}$
24	$m = \dfrac{(\gamma - y)^{-0,4}}{e^{\gamma} + e^{-y}} + \dfrac{lg^2\lvert y - 5,5\rvert + \sin^2 \dfrac{\gamma}{4}}{\sqrt{\lvert \gamma + y\rvert} + \sqrt[3]{arctgy}}$	$y = \begin{cases} arctg(\pi x), & 0 < x < \pi \\ \ln(x - 3,18), & 2\pi \le x \\ \dfrac{1}{\sqrt{x - \pi}}, & \pi < x < 2\pi \\ \pi, & x \le 0 \end{cases}$
25	$u = 0,3 \log_5 e^{-\lvert x\rvert + 2} + \dfrac{arctg \dfrac{\pi}{x} - \sin^2 \dfrac{x}{\pi}}{\gamma \sqrt{\ln\lvert \pi - x\rvert + lg\lvert \dfrac{\pi}{x}\rvert}}$	$y = \begin{cases} e^{-\pi} + x^{-e}, & 1 < x \le e \\ \ln^2(x - e), & e < x < 10^3 \\ \dfrac{x^2}{e}, & 0 < x \le 1 \\ 2^x ctg^3 x^2, & x < 0 \\ 0, & x \le 0 \end{cases}$
26	$y = \dfrac{\omega \cdot x^{-3,4}}{e^{\omega x}} - \dfrac{tg \dfrac{x}{6} + ctg \dfrac{x^2 - 1,4}{6}}{arcctg(\sqrt{6,6 + x} - \lvert x^4 - \omega\rvert)}$	$y = \begin{cases} \sqrt{x}, & x > 0 \\ 2 - x^2, & x \le 0 \end{cases}$
27	$x = \gamma \cdot \left\lvert \dfrac{15,6\, e^{-y \cdot z}}{\sqrt{\lvert \gamma + \sin^2 z\rvert}} - \dfrac{\arccos \dfrac{\pi - z}{3} + e}{\sqrt[3]{\lvert z + 0,5\rvert} + \sqrt[6]{\gamma + z}} \right\rvert$	$y = \begin{cases} e^{-x}, & 1 < x < 2 \\ x^e + 1, & 2 \le x \le 5 \\ 1, & x < 1 \;\; x > 5 \end{cases}$
28	$p = (k + 1)^{-4,3} \gamma + \dfrac{arctg(k \cdot x) - 0,006\, x}{\sqrt{\lvert x\rvert + 3,41} + \sqrt[3]{\sin^4 \gamma x}}$	$y = \begin{cases} \sin^2 x, & x \le -1 \\ \sqrt{-x}, & -1 < x < 0 \\ x - lgx, & x > 1 \end{cases}$

29	$t = \dfrac{x^2 - y^2}{e^{-(x+y)}} - \sqrt{\dfrac{8,67 + \cos^3(x-y) + e^y}{\arcsin^3 \lvert y - x - 1 \rvert}} + a$	$y = \begin{cases} \dfrac{\sin x}{x}, & x > 0 \\ 0, & x = 0 \\ 2x^2 + \ln \lvert x \rvert, & x < 0 \end{cases}$
30	$c = (3,14)^{m-\gamma} + arctg(\gamma \cdot x) - \dfrac{\sqrt{\lvert \ln \gamma + \ln x \rvert}}{\sqrt[3]{lg\gamma} + \sqrt[5]{\cos x^{-1}}}$	$y = \begin{cases} 124 - e^x, & \lvert x \rvert < 1 \\ tg(x-1), & 1 < x < 10 \\ 1, & x \le -1 \;\; x \ge 10 \end{cases}$

Контрольні запитання

1. Які основні синтаксичні відмінності між `main` у C++ та `Main` у C#?
2. Як отримати доступ до математичних функцій (гріх, степінь) у C#?
3. Які методи класу `Console` використовуються для виведення даних?
4. Що таке «крок з обходом» під час відлагодження?
5. Для чого призначене вікно «Контрольні значення» (Watch) в IDE?
6. Чим відрізняється поведінка точок зупинки (breakpoints) від покрокового виконання?
7. Як ініціалізувати змінні безпосередньо в коді C#?
8. Яка роль методу `Main` у консольному додатку?
9. У чому різниця між `Console.Write` та `Console.WriteLine`?
10. Як працює вікно «Локальні» (Locals) під час дебагу?

Лабораторна робота №2
Базові синтаксичні конструкції мови C#
(максимум 4 бали)

Мета:

- продовження плавного переходу від мови C++ до мови C#;
- закріплення (повторення) навичок відлагодження програм;
- робота з методами.

Завдання

1. Розробити програму для купівлі/продажу деякої валюти із використанням користувацьких функцій. Продемонструйте навички користування відлагоджувачем. Реалізувати наступне:

а) (**1 бал**) обмін гривень на деяку валюту (згідно варіанту) та навпаки;

б) (**1 бал**) обмеження на обмін валют при недостатній кількості коштів в обміннику чи клієнта.

Варіанти

1) злоті;
2) чеські крони;
3) фунти стерлінгів;
4) єни;
5) вони;
6) юані
7) лівійські динари;
8) ларі;
9) ліри;
10) руфії;
11) швейцарські франки;

12) аріарі;
13) бати;
14) кецалі;
15) куни;
16) леви;
17) леки;
18) леї;
19) ліри;
20) реали;
21) рупії;
22) форинти;
23) євро;
24) драхми;
25) наїри.

2. (**2 бали**) Виконайте наведений нижче перелік завдань, які стосуються синтаксичних особливостей мови C#:

1) видумайте яким чином коректно було би доповнити програму масивом рядків. Наприклад, можна створити масив з іменами працівників обмінника;

2) в окремому методі (функції) продемонструйте вивід всіх елементів створено масиву за допомогою циклів `for` і `while` та методу `Length`.

Контрольні запитання

1. Як оголосити та викликати користувацьку функцію (метод) у C#?
2. Як працює властивість `Length` у масивах?
3. Чим відрізняється цикл `for` від `while` при обході масиву?
4. Як реалізувати умовне обмеження при недостатній кількості коштів?

5. Який тип даних найкраще підходить для зберігання імен (масив рядків)?

6. Як передати параметри в метод для обміну валют?

7. Що таке сигнатура методу в C#?

8. Як вивести всі елементи масиву в консоль одним циклом?

9. Для чого використовувати відлагоджувач при роботі з логікою обміну?

10. Як реалізувати «зворотний» обмін валюти в окремому методі?

Лабораторна робота №3
Структура проєкту. Класи. Coding Conventions
(максимум 4 бали)

Мета:

- набуття базових навичок у структуризації проєкту;
- оволодіння базовими навичками роботи з класами в C#;
- засвоєння принципів Coding Conventions.

Завдання

Модифікуйте виконане у лабораторній роботі №2 завдання:

1) створіть клас «Обмінник» в окремому файлі;
2) у класі обмінник має бути хоча би 3 перевантажені конструктори;
3) клас обмінник повинен належати простору імен «Банківські установи»;
4) у класі мають бути елементи як з модифікатором `public`, так і з `private`;
5) у класі має бути хоча би декілька властивостей;
6) у класі має бути хоча би 3 методи для здійснення валютних операцій, частина з яких – перевантажені;
7) створіть масив обмінників (об'єктів) з вивісками: «Швидкий обмін», «Вигідний обмін», «Якісний обмінник»;
8) в основному файлі проєкту реалізуйте просту взаємодію з обмінниками;
9) за допомогою послідовності символів `///` виконайте мінімальне документування розроблених методів;
10) дотримуйтесь Coding Conventions;
11) продемонструйте навички користування технологією IntelliSense.

Контрольні запитання

1. Що таке простір імен (`namespace`) і як він структурує проєкт?
2. Чим відрізняються модифікатори доступу `public` та `private`?
3. Що таке властивість (`property`) і чим вона відрізняється від поля класу?
4. Навіщо в одному класі створювати кілька перевантажених конструкторів?
5. Як працює перевантаження методів?
6. Яка роль символів `///` у документуванні коду?
7. Що таке Coding Conventions і чому вони важливі для розробника?
8. Як технологія IntelliSense допомагає при написанні коду класів?
9. Як створити масив об'єктів певного класу (наприклад, обмінників)?
10. Як винести клас в окремий файл у Visual Studio?

Лабораторна робота №4-5
Робота з класами в C#
(максимум 8 балів)

Мета:

- навчитись проєктувати додатки за допомогою UML-діаграм класів;
- набути навички розробки багатофайлових проєктів (класи, інтерфейси, взаємодія між об'єктами);
- набути навички об’єктно-орієнтованого програмування (інкапсуляція, наслідування, поліморфізм, абстракція);
- набути досвід розробки додатків, які імітують роботу з вбудованими і розподіленими застосуваннями, мобільними і гібридними системами.

Завдання

Продумати та схематично намалювати структуру додатку згідно варіанту. Додаток повинен включати в себе приблизно 10 класів із полями, властивостями та методами. Вимоги до виконаних завдань:

а) (**2 бали**) розробити UML-діаграму класів для додатку;

б) (**3 бали**) усі класи у додатку повинні виглядати повноцінними, тобто містити хоча б основний функціонал; мають бути абстрактні класи, абстрактні методи, інтерфейси, інкапсуляція та наслідування (реалізація).

в) (**2 бали**) зв’язки між класами мають бути повноцінними, тобто “спілкування” між класами має бути зв’язним і схожим на оповідання; презентація роботи має відбуватись із використанням клавіші F11 у вигляді оповідання; уточнення щодо деталей реалізації функціоналу викладач робитиме сам після презентації;

г) (**1 бали**) має бути сформований *.dll-файл.

Варіанти

Вбудовані системи

1. Розумний термостат.
2. Автоматизована система поливу.
3. Система контролю доступу (розумний замок).
4. Моніторинг стану навколишнього середовища.
5. Система керування вуличним освітленням.
6. Розумний акваріум.
7. Розумний годівник для тварин.
8. Автоматизована система сортування сміття.
9. Система розпізнавання автомобільних номерів.
10. Автоматизований диспетчер маршрутного транспорту.
11. Розумний будильник із аналізом сну.
12. Система контролю якості води.
13. Розумний холодильник із підрахунком терміну придатності.
14. Система керування сонячними панелями.
15. Автоматична станція контролю метеоданих.

Розподілені системи

16. Чат для локальної мережі.
17. Розподілена система обробки замовлень.
18. Система віддаленого керування комп'ютером.
19. Система онлайн-голосування.
20. Система розподіленої обробки відео.
21. Хмарна бухгалтерія.
22. Розподілена система управління розумним містом.
23. Система бронювання готельних номерів.
24. Розподілена система моніторингу вантажоперевезень.

Мобільні системи

25. Трекер фізичної активності.

26. Мобільний нотатник.
27. Мобільний контролер розумного будинку.
28. Фінансовий трекер витрат.
29. Персональний тренер (фітнес-додаток).
30. Система пошуку попутників (carpooling).

Гібридні системи

31. Система бронювання конференц-залів.
32. Додаток для розкладу занять студентів.
33. Розумний щоденник продуктивності.
34. Мобільний додаток для контролю доступу.
35. Система управління розумним офісом.
36. Система бронювання місць у кінотеатрі.
37. Система управління бібліотекою.
38. Автоматизована система обліку витрат підприємства.
39. Автоматизована система розкладу занять у ВНЗ.
40. Власний варіант.

ООП в C#

Ключові слова

1) `sealed` – використовується у випадку необхідності заборони наслідувати або клас, або властивість, або метод;

2) `base` – синонім імені базового класу; використовується для доступу до батьківських елементів;

3) `this` – синонім даного класу (точніше об’єкта даного класу);

4) `virtual` – використовується у батьківському класі для подальшого перевизначення методів та властивостей;

5) `override` – використовується у дочірньому класі для перевизначення методів та властивостей;

6) `new` – використовується у дочірньому класі для перевизначення методів та властивостей;

7) `abstract` – використовується у батьківському класі для подальшої реалізації класів, перевизначення методів та властивостей; має багато спільного з `virtual`;

8) `static` – якщо необхідно зробити деякі змінну чи метод, чи властивість спільними для цілого класу; якщо необхідно зробити клас статичним в межах простору імен;

9) `const` – якщо необхідно зробити деяке поле спільним і незмінним для цілого класу.

Приклади застосування ключових слів

(закреслене вказане для наочності і використовувати практично його не потрібно)

- `abstract class Class1` …наслідування… ~~`abstract class Class2`~~
- `abstract class Class1` …наслідування… ~~`class`~~ `Class2`
- `sealed class Class`
- `public abstract type Method()` …наслідування… ~~`public override type`~~ `Method()`
- `public abstract type Method()` …наслідування… ~~`public sealed override type`~~ `Method()`
- `public virtual type Method()` …наслідування… ~~`public override type`~~ `Method()`
- `public virtual type Method()` …наслідування… ~~`public sealed override type`~~ `Method()`
- `public type Method()` …наслідування… ~~`public new type`~~ `Method()`
- `public Constructor1()` …наслідування… ~~`public`~~ `Constructor2() : base()`
- `public virtual type Method()` …наслідування… ~~`public override type`~~ `Method() { base.Method() }`
- `public type Method()` …наслідування… ~~`public new type`~~ `Method() { base.Method() }`

Функціональність ключових слів в ООП

Дія	abstract	new	virtual	override	interface
Може бути застосовано до класу	+	–	–	–	+
Перевизначає /приховує метод	–	+ (приховує)	+ (дозволяє)	+ (перевизначає)	–
Реалізує метод (властивість)	обов'язково	–	–	+	+
Дозволяє подальше наслідування	+	+	+	+	+

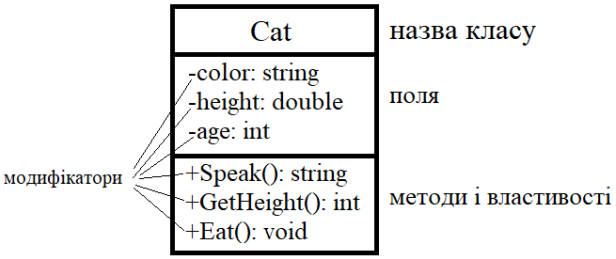
Матриця сумісності

(«+» – сумісні; «–» – несумісні; «x» – перетин одного й того самого слова)

	abstract	virtual	override	new	static	base	sealed
abstract	x	–	+	–	–	–	–
virtual	–	x	–	–	–	–	–
override	+	–	x	–	–	+	+
new	–	–	–	x	+	+	–
static	–	–	–	+	x	–	–
base	–	–	+	+	–	x	+
sealed	–	–	+	–	–	+	x
public	+	+	+	+	+	–	+
private	–	–	–	+	+	–	–

Діаграма класів на мові UML

Позначення класу



Тут:

- зверху вказується назва класу. Якщо цей клас абстрактний, то назва вказується курсивом (*Cat*).
- нижче вказуються поля класу за шаблоном:

модифікатор_доступу назва: тип

- найнижче вказуються методи (та властивості) за шаблоном:

модифікатор_доступу назва(аргумент: тип, аргумент: тип): тип_повернення

Таблиця модифікаторів доступу

Модифікатори доступу	
+	`public`
-	`private`
#	`protected`
~	`internal`

Пояснення відношень між класами

Назва відношення та його графічне позначення	Опис

Асоціація (has-a)
* 1
 	
(в деякій літературі не малюють стрілку)
	Якщо об'єкт одного класу використовує об'єкт іншого класу без співвідношення ціле-частина, і якщо цей об'єкт зберігається у якомусь з полів (чи властивості).
	Коментарі по краях лінії означають множинність зв'язків, тобто кількість об'єктів, які беруть участь у відношенні. Наприклад:
	`public class Order`
	`{`
	`  private Customer customer;`
	`}`

Агрегація (has-a + whole-part) (в деякій літературі не малюють стрілку)	Якщо об'єкт одного класу використовує об'єкт іншого класу у відношенні ціле-частина, і якщо цей об'єкт отримано ззовні та зберігається у якомусь з полів (чи властивості). Наприклад: `public class Computer`

<table>
<tr><td></td><td>

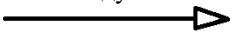
```
{
    private Monitor _monitor;
    public Computer(Monitor monitor)
    {
        _monitor = monitor;
    }
}
```

</td></tr>
<tr><td>Композиція
(has-a + whole-part + ownership)
 
(в деякій літературі не малюють стрілку)</td><td>Якщо об’єкт одного класу використовує об’єкт іншого класу у відношенні ціле-частина, і якщо цей об’єкт створено всередині та зберігається у якомусь з полів (чи властивості). Наприклад:

```
public class Apartment
{
    private Room bedroom;
    public Apartment()
    {
        bedroom = new Room();
    }
}
```

</td></tr>
<tr><td>Залежність
(references)
  </td><td>Якщо об’єкт одного класу використовує об’єкт іншого класу у його методі чи властивості, і якщо цей об’єкт не зберігається у жодному з полів. Наприклад:

```
public class EnrollmentService
{
    public void Move(Student s,
                     Course c)
    {
    }
}
```

</td></tr>
<tr><td>Наслідування
  </td><td>Якщо один клас наслідує інший. Наприклад:

```
public class Student: Person
{
}
```

</td></tr>
<tr><td>Реалізація
 
(в деякій літературі використовують зафарбовану стрілку)</td><td>Якщо один клас реалізовує інтерфейс іншого. Наприклад:

```
public class Student: IMove
{
    public void Move()
    {
    }
}
```

</td></tr>
<tr><td>Яка різниця між залежністю і асоціацією?</td><td>Залежність дуже нагадує асоціацію по-суті і по-позначенню. Залежність працює зі</td></tr>
</table>

	змінними на локальному рівні. Асоціація ж працює зі змінними на глобальному рівні.
Яка різниця між агрегацією і композицією?	Агрегація дуже нагадує композицію по-суті і по-позначенню. Агрегація змінює (чи надає) значення полів ззовні, а композиція – зсередини класу.
Яка різниця між асоціацією/ агрегацією і композицією?	Агрегація і композиція – це підвиди асоціації. Якщо немає бажання вдумуватись з чим маємо справу, то можна (хоч і не бажано) вказувати асоціацію.

Насправді в інтернеті можна знайти різнотлумачення наведених співвідношень і деякі відмінності у позначеннях відношень. Наприклад, часто в асоціації, агрегації і композиції відсутні стрілки. Також буває, що на діаграмі класів вказано одне відношення, а при реалізації коду – вказується інше. Такого роду випадки часто свідчать або про помилковість розуміння автором матеріалу, або про описку, або про непринциповість застосування того чи іншого відношення у тому конкретному випадку.

Алгоритм визначення виду зв'язку між класами

1. Якщо має місце вираз виду `class Student: Person`, то це наслідування.

2. Якщо має місце вираз виду `class Student: IMove`, то це реалізація.

3. Якщо у класі А міститься глобальна змінна (поле) класу Б, то:

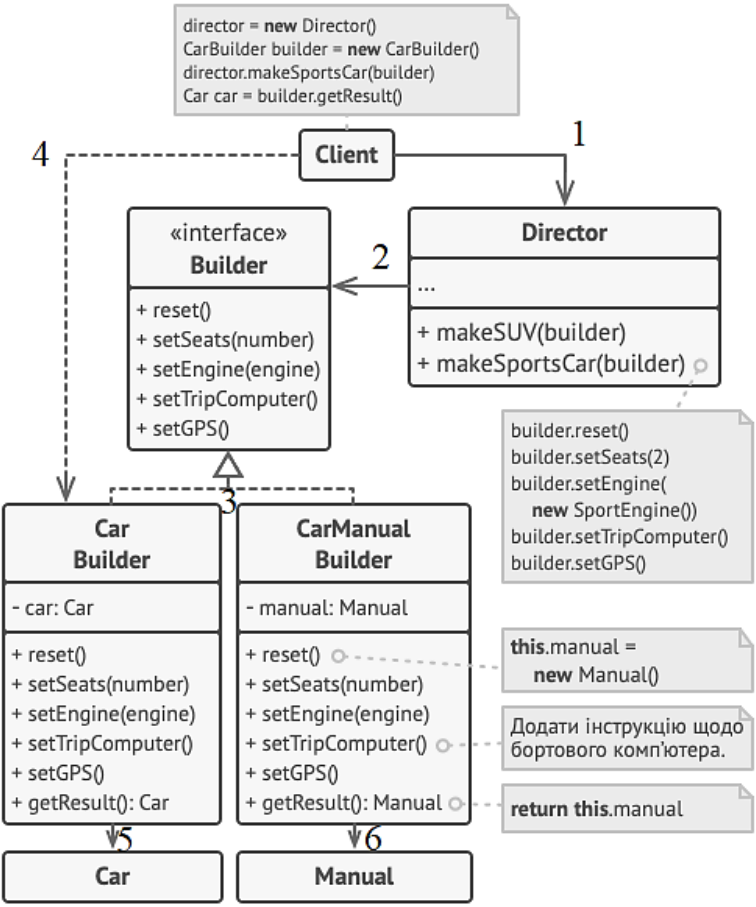
   а) якщо має місце співвідношення ціле-частина і поле класу А набуває свого значення всередині класу А, то це – композиція.

   б) якщо має місце співвідношення ціле-частина і поле класу А набуває свого значення звідкись ззовні, то це – агрегація;

   в) якщо не має місце співвідношення ціле-частина, то це – асоціація;

4. Якщо у класі А міститься локальна змінна класу Б, то це – залежність.

Приклад діаграми класів



Тут зв'язок 1 вказує на те, що в `Client` є глобальна змінна (поле) типу `Director`. Об'єкт класу `Director` аж ніяк не є складовою класу `Client` з точки зору ціле-частина. Тому логічно було позначити даний зв'язок як асоціацію. Та ж сама ситуація із зв'язком 2. Через 3 позначено реалізацію інтерфейсу `Builder`. Автори даної UML-діаграми

передбачають, що клас `Client` використовуватиме клас `Car Builder` на локальному рівні (без використання глобальної змінної). Тому для 4 вказано зв'язок залежності. Для 5 та 6 ситуація та ж сама, що й для 1 і 2.

Загалом на наведеній діаграмі бачимо 6 класів та 1 інтерфейс. Для ключових методів вказані їхні реалізації. Типи ж параметрів методів опущені. А типи повернення в усіх, крім двох методів, є `void`. Тому автори вважали зручним опустити слово `void`.

Наведені спрощення та фрагменти реалізації коду лише спрощують розуміння UML-діаграми класів, тому є допустимими. Безпосередні деталі реалізації коду залежать від самого програміста та можуть бути наведеними на інших видах діаграм.

Контрольні запитання

1. Які основні елементи відображаються на UML-діаграмі класів?
2. У чому різниця між абстрактним класом та інтерфейсом?
3. Як реалізувати інкапсуляцію в C#?
4. Що таке поліморфізм і як він проявляється через наслідування?
5. Як працює клавіша F11 при презентації «оповідання» по коду?
6. Що таке файл з розширенням `*.dll` і як його сформувати?
7. Як пов'язати 10 класів у єдину логічну систему?
8. Яка мета використання абстрактних методів?
9. Чим відрізняється вбудована система від розподіленої в контексті проєктування?
10. Як UML-діаграма допомагає уникнути помилок на етапі кодування?

Лабораторна робота №6-7
Вступ до патернів
(максимум 8 балів)

Мета: набуття практичних навичок роботи з деякими порівняно простими у реалізації патернами в C# на прикладі модифікації вже розробленого консольного додатку, який імітує роботу того чи іншого підприємства або установи.

Завдання

1. Для лабораторної роботи №4-5 у командах вдало реалізувати:

   а) (**1 бали**) породжуючий патерн проєктування “Одинак”; його можна, наприклад, використати для ведення бази даних продажів чи переліку клієнтів тощо;

   б) (**2,5 бали**) структурний патерн проєктування “Фасад”; його можна, наприклад, використати для формування переліку типових послуг чи послідовностей дій;

   в) (**2,5 бали**) поведінковий патерн проєктування “Хранитель” (для зручності можна користуватись вкладенням класів); його поведінку можна поєднати із патерном "Одинак, наприклад, щоб видаляти останній(-ні) введений(-ні) запис(-си)".

2. (**2 бали**) Побудувати відповідні UML-діаграми.

Зрозуміло, що студент повинен повністю розуміти код. Щодо UML-діаграм, то оцінка виставлятиметься, в першу чергу, за старання, а не за правильність побудови.

Коротка характеристика патернів

Породжуючі (якщо ціллю є створення об’єкта)

Одинак (Singleton): якщо щось має бути єдиним у програмі (БД, логер, адмін-панель).

Прототип (Prototype): якщо потрібно просто клонувати існуючий об'єкт, а не створювати новий з нуля.

Будівельник (Builder): якщо об'єкт занадто складний і його треба збирати покроково (як конструктор).

Фабричний метод (Factory Method): якщо програма має сама вирішити, об'єкт якого класу створити залежно від умов.

Абстрактна фабрика (Abstract Factory): якщо треба створювати цілі набори схожих об'єктів (наприклад, всі кнопки й вікна в стилі macOS або в стилі Windows).

Структурні (якщо ціллю є підвищення зручності)

Фасад (Facade): якщо треба сховати складну систему за однією простою кнопкою «зробити все красиво».

Адаптер (Adapter): якщо треба «подружити» старий код із новим (як перехідник для розетки).

Проксі (Proxy): якщо треба поставити «контролера» перед об'єктом, щоб він перевіряв права або кешував дані.

Декоратор (Decorator): якщо треба накинути на об'єкт нові фішки, не переписуючи його код (як обгортка).

Міст (Bridge): якщо треба розділити те, **що** ми робимо, від того, **як** ми це робимо (наприклад, пульт окремо, а телевізор окремо).

Легковаговик (Flyweight): якщо об'єктів мільйони й треба економити пам'ять, винісши спільні дані в одне місце.

Компонувальник (Composite): якщо треба працювати з групою об'єктів так само просто, як з одним (наприклад, папка, в якій можуть бути і файли, і інші папки).

Поведінкові (якщо ціллю є організація зв'язного "спілкування" між об'єктами)

Ланцюг обов'язків (CoR): якщо треба передавати запит по черзі, поки хтось не погодиться його обробити.

Команда (Command): якщо треба перетворити дію на об'єкт, щоб її можна було скасувати (Ctrl+Z) або покласти в чергу.

Ітератор (Iterator): якщо треба зручно перебрати елементи в списку, не знаючи, як він влаштований всередині.

Посередник (Mediator): якщо об'єктів багато і вони всі сваряться між собою – створюємо «диспетчера», через якого вони спілкуються.

Хранитель (Memento): якщо треба зробити «сейв» стану об'єкта, щоб потім можна було відкотитися назад.

Наглядач (Observer): якщо треба реалізувати розсилку: один щось змінив – усі підписники дізналися про це.

Стан (State): якщо об'єкт змінює свою поведінку залежно від того, в якому він «настрої» (статусі).

Стратегія (Strategy): якщо є кілька способів щось зробити (наприклад, різні види доставки) і ми хочемо їх легко міняти.

Шаблонний метод (Template Method): якщо є чіткий план дій, але деякі кроки плану можна міняти в дочірніх класах.

Відвідувач (Visitor): якщо треба додати нову логіку в купу різних класів, не міняючи код цих класів.

Рекомендована література

1. Сайт https://refactoring.guru/uk/design-patterns:
   - https://refactoring.guru/uk/design-patterns/catalog
   - https://refactoring.guru/uk/design-patterns/csharp
2. Книга https://files.znu.edu.ua/files/Bibliobooks/Inshi80/0059779.pdf.

Контрольні запитання

1. Для яких задач ідеально підходить патерн «Одинак»?
2. Як «Фасад» допомагає спростити взаємодію зі складною системою?
3. Яка основна функція патерна «Хранитель» (Memento)?
4. Як поєднати «Хранитель» з «Одинаком» для відміни операцій?
5. Чим відрізняється «Будівельник» від «Фабричного методу»?
6. Яку проблему вирішує патерн «Легковаговик»?
7. Як «Компонувальник» працює з деревоподібними структурами?
8. Коли варто використовувати «Декоратор» замість звичайного наслідування?
9. Яка роль «Адаптера» при інтеграції старого коду?
10. Що таке «Міст» і як він розділяє абстракцію та реалізацію?

Лабораторна робота №8-9
Вступ до патернів
(максимум 8 балів)

Мета: Набуття практичних навичок роботи з деякими складнішими у реалізації патернами в C# на прикладі модифікації вже розробленого консольного додатку, який імітує роботу того чи іншого підприємства або установи.

Завдання

1. Для лабораторної роботи 6-7 вдало реалізувати:
   а) (**4 бали**) породжуючий патерн проєктування “Фабричний метод”;
   б) (**2,5 бали**) поведінковий патерн проєктування “Спостерігач”.
2. (**1,5 бали**) Побудувати відповідні UML-діаграми.

Зрозуміло, що студент повинен повністю розуміти код. Щодо UML-діаграм, то оцінка виставлятиметься, в першу чергу, за старання, а не за правильність побудови.

Рекомендована література

1. Сайт https://metanit.com/sharp/tutorial/.
2. Сайт https://refactoring.guru/uk/design-patterns:
   - https://refactoring.guru/uk/design-patterns/catalog
   - https://refactoring.guru/uk/design-patterns/csharp
3. Книга https://learn.ztu.edu.ua/pluginfile.php/632/mod_resource/content/1/DesignPatterns_AndriyBuday.pdf.
4. Текстовий лекційний матеріал.

Додаткова література

1. https://www.youtube.com/watch?v=dhnsegiPXoo&list=PLLWMQd6PeGY3ob0Ga6vn1czFZfW6e-FLr

2. https://www.youtube.com/watch?v=VqgXn7wsPsc&list=PLuGqgO5WmeGOGGbaBwJvvQ7_Su6cIBc8C

Контрольні запитання

1. Як «Фабричний метод» дозволяє підкласам вибирати тип об'єкта?
2. Опишіть механізм підписки у патерні «Спостерігач» (Observer).
3. У чому різниця між «Абстрактною фабрикою» та «Фабричним методом»?
4. Як патерн «Команда» допомагає реалізувати чергу операцій?
5. Що таке «Ланцюг обов'язків» і як він шукає обробника запиту?
6. Коли патерн «Посередник» (Mediator) стає необхідним?
7. Як патерн «Стан» змінює поведінку об'єкта?
8. Чим «Стратегія» відрізняється від «Шаблонного методу»?
9. Навіщо потрібен «Ітератор» при роботі з колекціями?
10. Яка головна мета патерна «Відвідувач» (Visitor)?

Лабораторна робота №10-11
Робота з чужими проєктами
(максимум 24 балів)

Мета:

- більш глибоке дослідження структурних патернів проєктування;
- розвиток навичок міжособистісної взаємодії (загальна компетентність);
- розвиток вміння виявляти, ставити та вирішувати проблеми (загальна компетентність);
- невеличкий розвиток здатності працювати в команді (загальна компетентність);
- розвиток здатності аргументувати вибір методів розв'язання спеціалізованих задач, критично оцінювати отримані результати, обґрунтовувати та захищати прийняті рішення (спеціальна компетентність);
- розвиток вміння застосовувати знання для ідентифікації, формулювання і розв'язування технічних задач спеціальності, використовуючи методи, що є найбільш придатними для досягнення поставлених цілей (знання);
- розвиток навичок публічних виступів;
- тощо.

Завдання

1. Коректно добудувати до проєктів трьох одногрупників по одному структурному патерну: "Декоратор" (**1 бали**), "Адаптер" (**1,5 бали**), "Проксі" (**1,5 бали**); представити відповідні діаграми класів. Наприклад, якщо мова йшла би про електрокар, то:

- за допомогою патерна "Декоратор" можна було б до існуючого класу ElectricCar "прикрутити" мигалки;

- за допомогою патерна “Адаптер” можна було б щодо існуючого методу підзарядки автомобіля “прикрутити” можливість заряджатись від звичайної розетка 220В;
- за допомогою патерна “Проксі” можна було б добавити якесь додаткове сигналювання про те, що хтось відімкнув авто, зробити, щоб включалась музика одразу при заведенні авто тощо.

Варто зазначити, що через умовний рік після того, як в ІТ-компанії створили програмне забезпечення, протестували його, виправили знайдені баги, знову протестували і т.д., в певних випадках знову доведеться втрутитись в код, щоб щось відкоригувати. Якщо щось потрібно точково добавити, то краще застосувати один з трьох перелічених патернів.

2. (**6 балів**) Коректне компонування всіх раніше виконаних завдань у цілісний додаток засобами Windows Forms.

3. (**14 балів**) Представлення виконаного командного проєкту.

Рекомендована література

1. Сайт https://metanit.com/sharp/tutorial/.
2. Сайт https://refactoring.guru/uk/design-patterns:
   - https://refactoring.guru/uk/design-patterns/catalog
   - https://refactoring.guru/uk/design-patterns/csharp
3. Книга https://learn.ztu.edu.ua/pluginfile.php/632/mod_resource/content/1/DesignPatterns_AndriyBuday.pdf.

Додаткова література

1. https://www.youtube.com/watch?v=dhnsegiPXoo&list=PLLWMQd6PeGY3ob0Ga6vn1czFZfW6e-FLr

2. https://www.youtube.com/watch?v=VqgXn7wsPsc&list=PLuGqgO5WmeGOGGbaBwJvvQ7_Su6cIBc8C

Контрольні запитання

1. Яке основне призначення патерна «Декоратор» при модифікації існуючого класу?
2. У чому полягає відмінність між патернами «Адаптер» та «Проксі»?
3. Як патерн «Проксі» допомагає контролювати доступ до об'єкта в чужому коді?
4. Чому при допрацюванні існуючого проєкту краще використовувати структурні патерни замість прямого редагування базового коду?
5. Наведіть приклад ситуації, коли «Адаптер» є критично необхідним для взаємодії двох несумісних класів.
6. Як використання «Декоратора» сприяє дотриманню принципу відкритості/закритості (Open/Closed Principle)?
7. Які технічні складнощі можуть виникнути при спробі інтегрувати свій код у проєкт іншого розробника?
8. Чи може один об'єкт бути одночасно загорнутий у «Декоратор» та мати «Адаптер»?
9. Яку роль відіграє діаграма класів у процесі аналізу структури проєкту одногрупника?
10. Як патерн «Декоратор» допомагає уникнути створення надмірної кількості підкласів?
11. Як ви аргументуєте вибір конкретного структурного патерна для вирішення поставленої задачі під час презентації?
12. Які критерії ви використовували для оцінки якості інтеграції вашого патерна в проєкт одногрупника?
13. Чому важливо вміти критично оцінювати архітектурні рішення, прийняті іншими розробниками?

14. Яким чином використання патернів полегшує підтримку програмного продукту через значний час після його релізу?

15. Як правильно структурувати презентацію технічного рішення, щоб обґрунтувати вибір методів розв'язання задачі?

16. Що на практиці означає «розвиток навичок міжособистісної взаємодії» під час спільної роботи над кодом?

17. Як ви вирішували конфлікти в структурі класів, що виникали при поєднанні вашої частини з кодом іншого студента?

18. Які переваги дає використання інтерфейсів при реалізації патерна «Адаптер» у великих системах?

19. Як патерн «Проксі» може бути використаний для оптимізації роботи (наприклад, через відкладену ініціалізацію)?

20. Які soft skills (м'які навички) є найбільш критичними для успішного захисту архітектури вашого проєкту?

Рекомендована література

Основна

1. Будай А. Дизайн-патерни — просто, як двері. 2012. 90 с. [Електронний ресурс]. URL: https://drive.google.com/file/d/1pLQL5y_q8p69uDBH5dcC4wlENku_rHy-/view (Last access: 14.01.2026).

2. Robert M. Functional Design: Principles, Patterns, and Practices. 1st ed. Boston : Addison-Wesley Professional, 2023. 384 p.

3. Фрімен Е., Робсон Е. Head First. Патерни проєктування. Харків : Фабула, 2020. 672 с.

4. Yaroshenko T. Design Patterns in .NET: Mastering design patterns to write dynamic and effective .NET Code. 1st ed. Delhi : BPB Publications, 2024. 489 p.

5. Коноваленко І. В. Платформа .NET та мова програмування C# 8.0 : навчальний посібник. Тернопіль : ФОП Паляниця В. А., 2020. 320 с.

Додаткова

6. Gamma E., Vlissides J., Johnson R., Helm R. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

7. Refactoring and Design Patterns. URL: https://refactoring.guru/ (Last access: 14.01.2026).

8. C# Coding Conventions | Microsoft Learn: https://learn.microsoft.com/en-us/dotNET/csharp/fundamentals/coding-style/coding-conventions (Last access: 14.01.2026).

9. Стандарт ECMA по C#: https://www.ecma-international.org/wp-content/uploads/ECMA-334_6th_edition_june_2022.pdf (Last access: 14.01.2026).

Корисні посилання

10. Михайло Володимирович Бойчура - YouTube. URL: https://www.youtube.com/@mvboichura (Last accessed: 14.01.2026).

11. Getting Started with C# - YouTube. URL: https://www.youtube.com/playlist?list=PLLWMQd6PeGY2GVsQZ-u3DPXqwwKW8MkiP (Last accessed: 14.01.2026).

12. Design Patterns in Object Oriented Programming. URL: https://www.youtube.com/playlist?list=PLrhzvIcii6GNjpARdnO4ueTUAVR9eMBpc (Last accessed: 14.01.2026).

13. 10 Design Patterns Explained in 10 Minutes. URL: https://www.youtube.com/watch?v=tv-_1er1mWI (Last accessed: 14.01.2026).

14. SMD /OOAD | Software Design & Modeling | Object Oriented Analysis & Design | UML | Unified Approach. URL: https://www.youtube.com/playlist?list=PL5neT6krvZKY2aq1pMF6V9ycr-DTu7arp (Last accessed: 14.01.2026).

15. Design Patterns & Principles. URL: https://www.youtube.com/watch?v=dhnsegiPXoo&list=PLLWMQd6PeGY3ob0Ga6vn1czFZfW6e-FLr (Last accessed: 14.01.2026).