МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Методичні рекомендації до лабораторних робіт

для студентів спеціальності 121 "Інженерія програмного забезпечення" першого (бакалаврського) рівня

Харків
ХНЕУ ім. С. Кузнеця
2021

УДК 004.42(07.034)
О-13

Укладачі: Ю. Е. Парфьонов
О. В. Щербаков

Затверджено на засіданні кафедри інформаційних систем.
Протокол № 9 від 19.03.2021 р.

Самостійне електронне текстове мережеве видання

Об'єктно-орієнтоване програмування [Електронний ресурс] : О-13 методичні рекомендації до лабораторних робіт для студентів спеціальності 121 "Інженерія програмного забезпечення" першого (бакалаврського) рівня / уклад. Ю. Е. Парфьонов, О. В. Щербаков. – Харків : ХНЕУ ім. С. Кузнеця, 2021. – 93 с.

Подано методичні рекомендації до виконання лабораторних робіт, метою яких є закріплення та поглиблення знань теоретичного матеріалу, набуття навичок у розробленні застосунків за допомогою об'єктно-орієнтованих мов програмування. За кожною лабораторною роботою визначено мету, завдання, порядок виконання та контрольні запитання.

Рекомендовано для студентів спеціальності 121 "Інженерія програмного забезпечення" першого (бакалаврського) рівня.

УДК 004.42(07.034)

Вступ

З кожним роком інформаційні технології набувають все більшого розповсюдження, проникаючи в усі сфери людської діяльності. Ефективність застосування сучасних інформаційних технологій визначається, насамперед, ефективністю та якістю програмного забезпечення, яке покладено в їхню основу. Саме тому, галузь розроблення програмного забезпечення набуває все більшого значення, оскільки складність та функціональні можливості комп'ютерної техніки, що швидко зростають, потребують більш досконалих програмних засобів для задоволення потреб користувачів.

Істотною рисою таких програмних систем є рівень складності: для одного розробника практично неможливо охопити всі їхні аспекти. Причому ця складність є неминучою: з нею можливо справитися, але позбавитися від неї неможливо.

У теперішній час найбільш розповсюдженим методом боротьби зі складністю є об'єктно-орієнтований підхід до розроблення програмного забезпечення. З використанням цього підходу створюється більша частина програм у всьому світі. Отже, навчальна дисципліна "Об'єктно-орієнтоване програмування" є надзвичайно важливою для бакалаврів спеціальності 121 "Інженерія програмного забезпечення" в їхній майбутній професійній діяльності.

Базовими мовами програмування для виконання лабораторних робіт є C#, Java та Python, які інтенсивно використовуються компаніями-розробниками програмного забезпечення. В цих мовах також реалізовано більшість концепцій об'єктно-орієнтованого підходу в "класичному" вигляді. Їх також досить легко вивчати, тому що C# та Java є C-подібними мовами програмування, синтаксис яких має багато спільних рис.

Запропонований курс лабораторних робіт спрямовано на формування стійких практичних навичок щодо розроблення застосунків з використанням об'єктно-орієнтованого підходу.

Для розроблення програм студент має можливість вибрати мову програмування C# або Java. Також рекомендується використовувати такі інтегровані середовища розроблення, як: Microsoft Visual Studio, JetBrains Rider, IntelliJ IDEA та Eclipse.

Загальні рекомендації до виконання лабораторних робіт

Необхідним елементом успішного засвоєння матеріалу навчальної дисципліни "Об'єктно-орієнтоване програмування" є самостійна робота студентів. Тому в процесі виконання лабораторних робіт доцільно спочатку опрацювати певні джерела зі списку рекомендованої літератури, зокрема навчальний посібник "Основи об'єктно-орієнтованого програмування".

Звіт з будь-якої лабораторної роботи має містити:

1. Титульний лист (відомості про назву навчальної дисципліни; тему лабораторної роботи; ПІБ студента, курс, спеціальність, шифр навчальної групи; ПІБ та посаду викладача).
2. Лист змісту (нумерований перелік назв структурних елементів роботи (див. пункт 3) із зазначенням номерів сторінок).
3. Опис виконаних завдань:

а) умова завдання;

б) опис архітектури програми (структура класів, зв'язки між ними, алгоритми тощо);

в) вихідний код програми;

г) приклади результатів роботи програми на тестових вихідних даних.

4. Висновки з лабораторної роботі з урахуванням усіх завдань, що були виконані: аналіз результатів, які були одержані за кожним завданням; ступінь відповідності програм, що були розроблені, формулювання завдання; інша інформація.

Оцінювання результатів виконання лабораторної роботи

Результати виконання лабораторної роботи оцінюються за такими параметрами:

1. Якість розробленого програмного продукту.

Оцінюється за ступенем його оригінальності, відповідності вимогам певного завдання роботи та якістю розробленого програмного коду.

2. Виконання календарного плану роботи відповідно до технологічної карти навчальної дисципліни.
3. Якість демонстрації результатів роботи на її захисті.
4. Якість виконання звіту.

Оцінюється за ступенем оригінальності його контенту та відповідності вимогам до змісту й оформлення, викладених у цих методичних рекомендаціях.

Встановлюються таки рівні оцінювання результатів роботи (відповідно до технологічної карти навчальної дисципліни):

Рівень1. До 70 % від максимальної оцінки.

Рівень 2. До 80 % від максимальної оцінки.

Рівень 3. До 90 % від максимальної оцінки.

Рівень 4. До 100 % від максимальної оцінки.

Лабораторна робота 1
Основи використання мов C# та Java

Мета лабораторної роботи:

1. Придбання практичних навичок розробки простих консольних програм з використанням основних елементів мов C# і Java.

2. Удосконалення навичок роботи з інтегрованим середовищем MS Visual Studio і довідковою системою Microsoft Developer Network (MSDN).

3. Придбання навичок роботи з інтегрованим середовищем Eclipse.

Перед виконанням лабораторної роботи студент має:

знати:

загальні відомості про мови C# і Java;

основи синтаксису мов C# і Java;

вміти розробляти прості консольні програми з використанням базових елементів мов C# та Java.

Порядок виконання лабораторної роботи

Загальні вимоги до розроблюваних програм:

• наявність найпростішого текстового інтерфейсу;

• забезпечення коректного оброблення початкових даних з метою запобігання появи помилок під час виконання програми (рівні 3 і 4).

Завдання 1

Використовуючи елементи класу Math (клас для підтримки математичних операцій) розробити програму табулювання значень деякої функції

в межах заданого діапазону зміни аргументу мовою C# або Java. Варіанти завдань наведено в табл. 1.1.

Таблиця 1.1

Вихідні данні

Номер варіанта	Функція, яка табулюється	Оператор циклу для розрахунку таблиці значень функції
1	$a^{1/2}$, де $1 \le a \le 10\,000$	while
2	Перетворення аргументу a (кілометр → миля), де $1 \le a \le 1\,000$	do while
3	exp(a), де $-5 \le a \le 5$	for
4	Перетворення аргументу a (Цельсій → Фаренгейт), де $-273° \le a \le 273°$	while
5	lg (a), де $1 \le a \le 1\,000\,000$	do while
6	Перетворення аргументу a (байт → килобайт), де $1 \le a \le 10^9$	for
7	sin(a), де $0° \le a \le 360°$	while
8	cos(a), де $0° \le a \le 360°$	do while
9	tg(a), де $-90° < a < 90°$	for
10	Перетворення аргументу a (гривна → євро), де $1 \le a \le 100\,000$	while
11	ctg(a), де $-180° < x < 180°$	do while
12	a^3, де $-100 \le a \le 100$	for
13	ln(a), де $1 \le a \le 100\,000$	while
14	Перетворення аргументу a (радіани → градуси), где $0 < a < 6{,}28$	do while
15	Перетворення аргументу a (градуси → радіани), де $0° \le a \le 360°$	for

Примітка: Слід врахувати, що стандартні методи класу Math для обчислення тригонометричних функцій очікують параметр у радіанах, а вихідні дані та результат мають бути наведені в градусах.

Обчислення кожного із значень функції має бути реалізовано у вигляді статичного методу класу (рівні 2 – 4). Вихідні дані ($a_{поч}$ – початкове значення аргументу; $A_{кін}$ – кінцеве значення аргументу; Δa – крок зміни аргументу) мають вводитися з консолі.

Програма має дозволяти:

• одноразово ввести початкові дані, отримати результати обчислень, вивести їх і завершити роботу (рівні 1 і 2);

• багаторазово повторювати процес введення початкових даних, отримання результатів обчислень і їхнє виведення до тих пір, поки користувач не введе певний рядок символів, після чого завершити роботу (рівні 3 і 4).

Кожне значення аргументу і відповідне йому значення функції мають виводитися на консоль з використанням засобів форматного виведення:

• необхідної кількості колонок із заголовками (числові значення з трьома знаками після коми) – рівні 1 і 2;

• заданої таблиці з горизонтальними і вертикальними лініями сітки (числові значення з заданим користувачем кількістю знаків після коми) – рівні 3 і 4.

Контрольні запитання до лабораторної роботи 1

1. Охарактеризуйте структуру програми мовою C# або Java.
2. Перерахуйте вбудовані типи даних C# або Java.
3. Яке призначення функцій класу System.Math або java.lang.Math?
4. Запишіть і охарактеризуйте загальний формат статичної функції.
5. Наведіть порядок створення проєкту консольного застосування в MS Visual Studio або Eclipse.
6. Наведіть порядок відкриття наявного проєкту в MS Visual Studio або Eclipse.
7. Наведіть порядок додавання файла в проєкт MS Visual Studio або Eclipse.
8. Наведіть порядок компіляції і запуску програми в MS Visual Studio або Eclipse.
9. Наведіть порядок документування вихідного коду програми за допомогою документаційних коментарів в C# або Java.

Лабораторна робота 2
Проєктування класів з використанням мови UML

Мета лабораторної роботи:

1. Придбання практичних навичок проєктування класів з використанням елементів мови UML.

Перед виконанням лабораторної роботи студент має:

знати:

загальні відомості про мову UML;

основи використання мови UML щодо створення об'єктних моделей;

вміти розробляти прості об'єктні моделі та створювати відповідні UML-діаграми класів.

Програмні засоби, що рекомендуються для створення UML-діаграм:

1) draw.io – спеціалізований web-застосунок для створення різних діаграм;

2) Microsoft Visio.

Приклад розроблення об'єктної моделі

Опис предметної області:

1. Підприємство веде неавтоматизований облік яблук з використанням відомості в "паперовому" вигляді.

2. Відомість має табличну структуру з чотирма стовпцями, кожний з яких має назву, яка відповідає одному з параметрів будь-якого яблука: сорт (S), колір (C), діаметр (D), об'єм (V). Об'єм будь-якого яблука обчислюється за формулою:

$$V_i = \frac{1}{6} \times \pi \times D^3,$$

де i – номер запису відомості.

Приклад заповненої відомості наведено у табл. 2.1.

Таблиця 2.1

Відомість обліку яблук

Сорт	Колір	Діаметр	Об'єм
S	C	D	V
Антонівка	жовтий	6	113,04
Джонатан	червоний	10	523,33
…	…	…	…
Середній об'єм яблука			120,99

3. Кожен запис відомості призначено для зберігання даних деякого яблука, наприклад:

Антонівка	жовтий	6	113,04

4. Кількість записів відомості не обмежена.

5. Співробітник підприємства може додавати до відомості дані про нове яблуко шляхом внесення до відповідних полів запису відомості початкових даних і даних, які були ним розраховані за формулою 1.

6. Наприкінці звітного періоду співробітник підприємства обчислює середній об'єм яблука за формулою 2 і заносить його в графу "Середній об'єм яблука" відомості:

$$V_{cp} = \frac{\sum V_i}{n},$$

де V_i – діаметр і-го яблука у відомості;

n – кількість яблук у відомості.

Постановка завдання

Розробіть об'єктну модель програмної системи, яка призначена для автоматизації обліку яблук на підприємстві.

Результати розроблення об'єктної моделі

Спосіб 1. Основні класи програмної системи – Apple, MainClass (рис. 2.1).

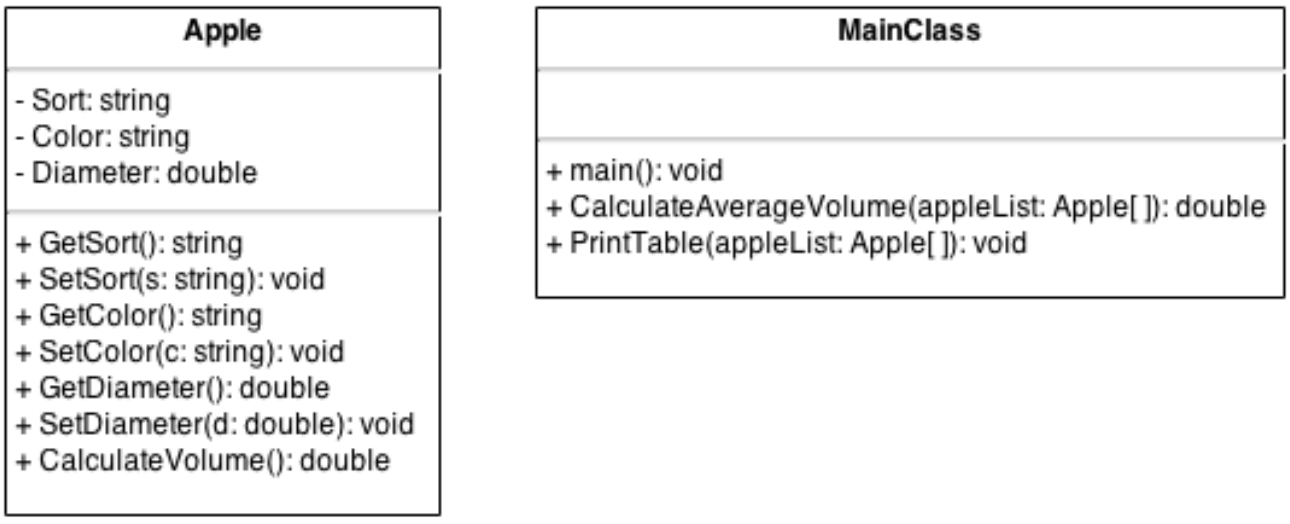

Рис. 2.1. **Рекомендована структура класів (завдання 1)**

Клас Apple призначено для опису будь-якого яблука.

Поле Sort – сорт яблука, поле Color – колір, поле Diameter – діаметр. Метод GetSort() – повертає поточне значення сорту яблука, метод GetDiameter() – повертає поточне значення діаметра, метод GetColor() – повертає поточне значення кольора. Метод SetSort() – встановлює нове значення сорту яблука, метод SetDiameter() – встановлює нове значення діаметра, метод SetColor() – встановлює нове значення кольора. Метод CalculateVolume() обчислює об'єм яблука.

Клас MainClass – головний клас. Метод CalculateAverageVolume() обчислює середній об'єм яблука у відомості. Призначення методу PrintTable() – "друк" значень полів об'єктів-яблук, діаметра яблука та підсумкових даних відомості.

Метод main() – точка входу в програму. Він містить масив об'єктів типу Apple для зберігання об'єктів-яблук. Цей метод призначено для введення бажаної кількості яблук у відомості, параметрів яблук, створення на основі введених параметрів об'єкта класу Apple для кожного з яблук, додавання цих об'єктів у масив, обчислення середнього об'єму яблука у відомості за допомогою виклику методу CalculateAverageVolume(), "друку" значень полів об'єктів-яблук та підсумкових даних відомості за допомогою виклику методу PrintTable().

Спосіб 2. Основні класи програмної системи – Apple, AppleList, MainClass (рис. 2.2).

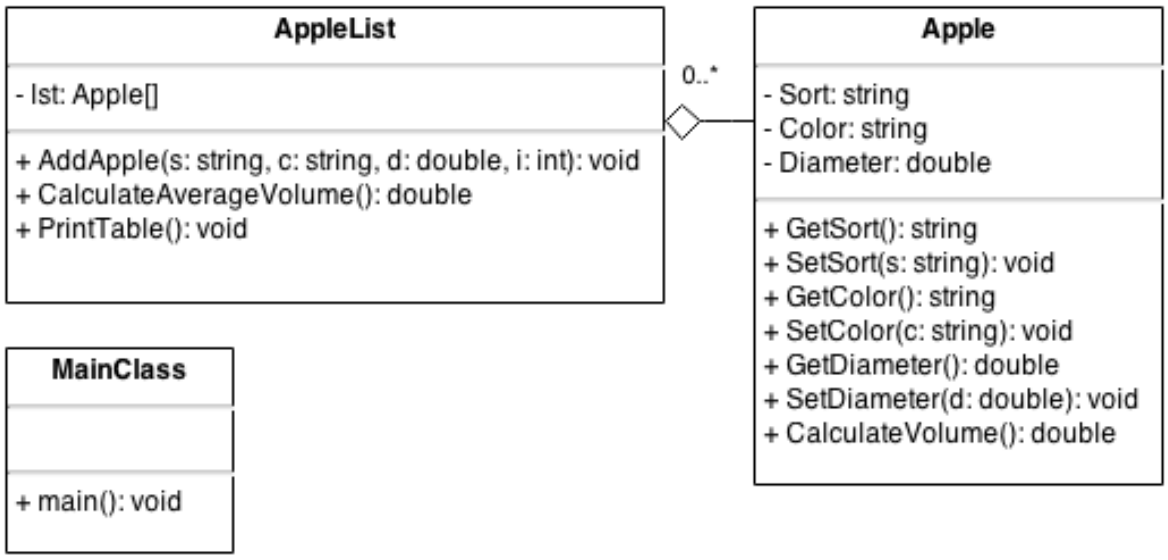

Рис. 2.2. **Рекомендована структура класів (завдання 2)**

Клас Apple призначено для опису будь-якого яблука. Поле Sort – сорт яблука, поле Color – колір, поле Diameter – діаметр. Метод GetSort() – повертає поточне значення сорту яблука, метод GetDiameter() – повертає поточне значення діаметра, метод GetColor() – повертає поточне значення кольора. Метод SetSort() встановлює нове значення сорту яблука, метод SetDiameter() – встановлює нове значення діаметра, метод SetColor() – встановлює нове значення кольора. Метод CalculateVolume() обчислює об'єм яблука.

Клас AppleList призначено для опису відомості обліку яблук. Між класами AppleList і Apple існує відношення агрегації: поле lst класу AppleList є масивом об'єктів типу Apple і призначене для зберігання об'єктів-яблук (тобто один об'єкт класу AppleList може містити кілька об'єктів класу Apple).

Метод AddApple() необхідний для додавання об'єкта-яблука в масив lst. Метод CalculateAverageVolume() обчислює середній об'єм яблука у відомості. Призначення методу PrintTable() – "друк" значень полів об'єктів-яблук, діаметра яблука та підсумкових даних відомості.

Клас MainClass – головний клас. Метод main() – точка входу в програму. Цей метод призначено для: введення бажаної кількості яблук у відомості; створення об'єкта класу AppleList; введення параметрів яблук; створення на основі введених параметрів об'єкта класу Apple для кожного з яблук; додавання цих об'єктів у відомість за допомогою виклику методу AddApple() об'єкта класу AppleList, виведення даних за допомогою виклику методу PrintTable() класу AppleList.

Порядок виконання лабораторної роботи

Завдання 1

Розробіть об'єктну модель програмної системи, яка призначена для автоматизації предметної області, що відноситься до діяльності "підприємства", відповідно до варіанта завдання.

Для цього необхідно визначити основні класи програмної системи, які необхідні для моделювання предметної області, їхні атрибути (найменування, тип даних) і методи (найменування, тип повертаємого значення). Також за допомогою одного з рекомендованих програмних засобів (див. "Програмні засоби, рекомендовані для створення UML-діаграм") потрібно розробити відповідну UML-діаграму класів. Навести цю діаграму та її опис, а саме: призначення кожного класу, його атрибутів та методів.

Завдання 2

Перетворіть об'єктну модель програмної системи, розроблену в завданні 1, у такий спосіб, щоб у ній використовувалося відношення агрегації або композиції.

Варіант 1

Відомість нарахування зарплати співробітникам підприємства:

Прізвище	Зарплата, грн	Утримано, грн	Видано, грн
F	Z	P	S = Z – P
…	…	…	…
Разом	∑	∑	∑

Варіант 2

Відомість витрат палива на автобазах міста:

Автобаза	Витрачено палива, кг	Кількість автомашин, шт.	Середня витрата палива, кг
A	T	K	C = T / K
…	…	…	…
Разом	∑	∑	∑ / n

Варіант 3

Відомість використання машинного часу на обчислювальному центрі:

Кафедра	Використання машинного часу, год		Відхилення від плану	
	за планом	фактично	у годинах	у %
K	P	F	$O_1 = P – F$	$O_2 = O_1 \times 100 / P$
…	…	…	…	…
Разом	∑	∑		

Варіант 4

Відомість споживання електроенергії на заводах міста:

Завод	Споживання електроенергії, кВт·год		Відхилення від плану	
	за планом	фактично	у кВт·год	у %
Z	P	F	$O_1 = P – F$	$O_2 = O_1 \times 100 / P$
…	…	…	…	…
Разом	∑	∑		

Варіант 5

Відомість руху матеріалів на складі підприємства за звітний період:

Склад	Рух матеріалів за період, грн			Залишок на кінець періоду
	залишок на початок періоду	отримано	видано	
C	O_c	P	V	$R = O_c + P - V$
…	…	…	…	…
Разом	∑	∑	∑	∑

Варіант 6

Відомість прибутку підприємства за звітний період за видами продукції:

Продукція	Кількість, шт.	Оптова ціна, грн	Собівартість, грн	Прибуток, грн
Pr	K	Z	C	P = K(Z – C)
…	…	…	…	…
Разом	∑			∑

Варіант 7

Відомість відвідування занять студентами:

Прізвище	Пропущено, год		Пропуски	
	усього	виправдано	у годинах	у %
F	V	O	P1 = V – O	P2 = P1 × 100 / V
…	…	…	…	…
Разом	∑	∑	∑	

Варіант 8

Відомість обсягу поставок продукції в натуральному та вартісному вираженні:

Продукція	Шифр	Обсяг поставки, шт.	Оптова ціна, грн	Обсяг поставки, грн
P	H	V	Z	O = V × Z
…	…	…	…	…
Разом		∑	∑	∑

Варіант 9

Відомість розрахунку середньої вартості перевезення авіапасажирів:

Тип літака	Рейс	Витрати на рейс, грн	Кількість пасажирів	Середня вартість перевезення, грн
T	R	Z	K	S = Z / K
…	…	…	…	…
Разом		∑	∑	∑ / n

Варіант 10

Відомість обліку часу роботи верстатів підприємства:

Тип верстата	Час роботи, год		Відхилення від плану	
	за планом	фактично	у годинах	у %
Z	P	F	O1 = P – F	O2 = O1 × 100 / P
…	…	…	…	…
Разом	∑	∑		

Варіант 11

Відомість випуску деталей робочими цеху:

Прізвище	Кількість деталей, шт.		Брак	
	виготовлено	прийнято	шт.	у %
Z	P	F	O1 = P – F	O2 = O1 × 100 / P
…	…	…	…	…
Разом	∑	∑	∑	

Варіант 12

Відомість наявності та руху основних фондів підприємства:

Назва фонду	Наявність на початок року, шт.	Надійшло, шт.	Вибуло, шт.	Наявність на кінець року, шт.
F	N1	P	V	N2 = N1 + P – V
…	…	…	…	…
Разом	∑	∑	∑	∑

Варіант 13

Відомість обліку запчастин на складі підприємства:

Марка автомобіля	Назва запчастини	Кількість одиниць товару, шт.	Ціна без ПДВ, грн	Ціна з ПДВ, грн
M	N	Q	P	AVP = P + 0,2 × P
…	…	…	…	…
Разом		∑	∑	∑

Варіант 14

Відомість обліку успішності студентів:

Прізвище	Ім'я	Оцінка з першої дисципліни	Оцінка з другої дисципліни	Середня оцінка студента
LN	FN	M1	M2	AM = (M1 + M2) / 2
…	…	…	…	…
Разом		∑ / n	∑ / n	

Варіант 15

Відомість обліку оплати за телефонні розмови:

Прізвище	Нараховано за міські розмови	Нараховано за міжміські розмови	Нараховано за міжнародні розмови	Нараховано всього з ПДВ
LN	C	IC	F	T = (C + IC + F) × 1,2
…	…	…	…	…
Разом	∑	∑	∑	

Контрольні запитання до лабораторної роботи 2

1. Охарактеризуйте прості та складні програмні системи.
2. Як можна боротися зі складністю програм?
3. У чому полягає сутність об'єктно-орієнтованої декомпозиції?
4. Які основні елементи має об'єктно-орієнтована технологія?
5. Яке співвідношення існує між класами й об'єктами?
6. Які характеристики має будь-який об'єкт?
7. Сформулюйте принципи об'єктно-орієнтованого підходу.

Лабораторна робота 3
Розроблення застосунків з використанням базових елементів об'єктно-орієнтованого програмування

Мета лабораторної роботи:

1. Придбання практичних навичок використання основних елементів об'єктно-орієнтованого програмування (ООП) та відносин агрегації і композиції при розробленні програм мовами Java і C#.

2. Удосконалення навичок роботи з інтегрованими середовищами Eclipse і MS Visual Studio.

Перед виконанням лабораторної роботи студент має:

знати:

поняття класу та об'єкта, співвідношення між ними;

принципи об'єктно-орієнтованого програмування;

синтаксис опису класу;

життєвий цикл об'єкта в програмі;

порядок використання конструкторів;

синтаксис агрегації;

синтаксис композиції;

вміти:

самостійно описувати класи мовами Java і C#;

застосовувати відношення та композиції.

Реалізація базових елементів ООП у мовах Java і C#

Парадигма програмування – це набір теорій, методів, стандартів, які використовуються під час розроблення та реалізації програм на комп'ютері. ООП – одна з парадигм, яка розглядає програму як безліч об'єктів, що взаємодіють. Кожен об'єкт здатний отримувати повідомлення, обробляти дані і передавати повідомлення іншим об'єктам. Кожен об'єкт можна розглядати як незалежний автомат з окремим призначенням або відповідальністю.

ООП базується на трьох основних принципах:

1. Інкапсуляція – об'єднання в єдине ціле (клас) даних і алгоритмів обробки цих даних. В ООП дані називаються полями, а алгоритми – методами.

2. Успадкування – можливість створення ієрархії класів, коли класи-нащадки отримують від класу-предка поля та методи.

3. Поліморфізм – це властивість класів однієї ієрархії вирішувати схожі за змістом завдання за допомогою різних алгоритмів.

Класи. Класи – це основний спосіб організації даних в Java і C#. Будь-яка програма, яка розробляється з їхнім використанням, має складатися не менше ніж з одного класу.

Функції в Java і C#. Будь-яка функція має бути членом класу (методом). Методи можуть приймати чи не приймати параметри, повертати чи не повертати значення. Параметри можуть передаватися в методи за значенням або за посиланням. Методи можуть бути статичними або методами примірників класу (об'єктів).

Модифікатори. Одним з важливих принципів об'єктно-орієнтованого підходу до програмування є інкапсуляція даних. Для вираження різних рівнів доступності елементів класу в Java і C# існує ряд модифікаторів. Наприклад, поля і методи можуть мати такі основні модифікатори доступу:

public (цей елемент класу доступний усім зовнішнім споживачам);

private (елемент недоступний за межами опису класу);

Крім того, існують модифікатори, що змінюють поведінку елементів класу, наприклад:

const (тільки в C# – означає, що це поле є константою);

final (тільки в Java – означає, що це поле є константою);

static (вказує, що даний елемент належить класу, а не його конкретному екземпляру).

Ключове слово this:

є посиланням на поточний екземпляр об'єкта;

посилання this є прихованим покажчиком, передається в кожен нестатичний метод класу;

будь-який метод може використовувати this для доступу до інших нестатичних методах і полях цього об'єкта.

Конструктори:

конструктор – метод, який ініціалізує стан об'єкта після його створення;

конструктор має те саме ім'я, що і клас, у якому він використовується;

конструктор не повертає значення (навіть типу void);

конструктор викликається автоматично;

якщо конструктор не заданий у програмі, то він буде автоматично згенерований компілятором для побудови відповідних об'єктів;

конструктор можна перевантажувати.

Приклад програми на Java. У цій програмі є клас Apple з конструктором, а також клас Program с методом main, у якому створюється об'єкт класу Apple, а потім викликаються його методи (аналогічна програма на C# відрізняється головним чином операторами для введення-виведення даних):

```
class Apple {
   // Поля
   private String sort, color;
   private double diameter;
   // Конструктор
   public Apple (String sort, String color, double diameter) {
      this.sort = sort;
      this.color = color;
      this.diameter = diameter;
   }
   // ---------- Методи start ----------
   public String getSort () {
      return sort;
   }

   public String getColor () {
       return color;
   }

   public double getDiameter () {
       return diameter;
    }

   public double calculateVolume () {
        return (Math.PI * Math.pow (diameter, 3) / 6);
   }
   // ---------- Методи end ----------
}

public class Program {
   public static void main (String [] args) {
         Apple apl = new Apple ( "Антонівка", "Жовтий", 8.5);
         System.out.println ( "Сорт:" + apl.getSort ());
```

```
        System.out.println ( "Колір:" + apl.getColor ());
        System.out.println ( "Діаметр:" + apl.getDiameter () + '\ n');
        System.out.format ( "Обсяг:% 1 $ 5.3f куб.см.", apl.calculateVolume ());
    }
}
```

Масиви об'єктів у Java (C#). Далі наведено фрагмент програми на C#, у якому створюється масив для зберігання двох об'єктів класу MyClass і два об'єкти цього класу додаються в масив:

```
MyClass [ ] ArrayOfObject = new MyClass [2];
for (int i = 0; i <ArrayOfObject.Length; i ++)
ArrayOfObject [i] = new MyClass ();
```

де *Length* – властивість, яка містить довжину масиву (в Java необхідно використовувати властивість *length*).

Реалізація відносин агрегації і композиції в Java і C#

Між класами або об'єктами можуть існувати різні логічні відносини.

Логічним відношенням, що часто зустрічається, між об'єктами є асоціація. Асоціація показує, що об'єкти одного класу пов'язані з об'єктами іншого класу. Якщо між двома класами встановлено зв'язок, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Асоціація, що зв'язує два класи, називається бінарною. Часто під час моделювання буває важливо вказати, скільки об'єктів може бути пов'язано за допомогою одного екземпляра асоціації. Це число називається кратністю ролі асоціації. Кратність, зазначена на одному з кінців асоціації, позначає, що на цьому кінці саме стільки об'єктів має відповідати кожному об'єкту на протилежному кінці. Можна задати кратність, яка дорівнює одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *) і т. д. Асоціації може бути присвоєно ім'я, яке описує природу відносини.

Приклад відносини асоціації наведено на рис. 3.1.

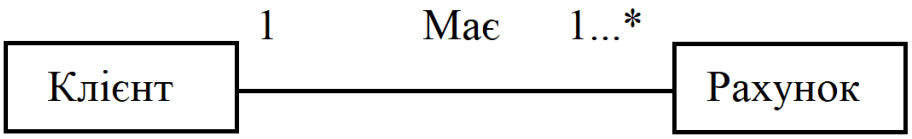

Рис. 3.1. **Відношення асоціації**

Проста асоціація відображає структурне відношення між рівноправними сутностями, коли обидва класи знаходяться на одному концептуальному рівні та жоден з них не є більш важливим, ніж інший.

Однак дуже часто доводиться моделювати відношення типу "частина/ціле", в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин). Відношення такого типу називають агрегацією. Воно є окремим випадком асоціації та зображується у вигляді простої асоціації з незафарбованим ромбом з боку "цілого".

Агрегація може означати фізичне входження одного об'єкта в інший, коли об'єкт-ціле не може існувати без об'єктів-частин. Наприклад, літак складається з крил, двигунів, шасі та інших частин. Тоді це сильніший варіант відносини агрегації, який називається композицією. Композиція зображується у вигляді простої асоціації з зафарбованим ромбом з боку "цілого".

Відмінності між композицією і агрегацією

Наприклад, відносини між об'єктами класів "Університет", "Студент" і "Факультет" злегка відрізняються один від одного, хоча обидва є відносинами агрегування. Відношення між об'єктами класів "Університет" і "Факультет" – це композиція, так як під час знищення об'єкта "Університет" об'єкти факультетів, що належать цьому університету, також мають бути знищені. Об'єкти класів "Студент" і "Університет" перебувають у відношенні агрегації тому, що об'єкт класу "Студент" може існувати після знищення цього університету, так як студент може вчитися в декількох навчальних закладах одночасно.

Агрегація (композиція) в Java і C#

Об'єкт, який є атрибутом іншого об'єкта (агрегату), має зв'язок зі своїм агрегатом. Через цей зв'язок агрегат може посилати йому повідомлення.

Синтаксично агрегація (композиція) – це використання посилання на об'єкт одного класу як поле другого класу.

Приклад реалізації відносини агрегації (один об'єкт класу AppleList може містити кілька об'єктів класу Apple) наведено далі (Java). У цьому прикладі об'єкти класу Apple створюються за межами класу AppleList, тобто об'єкти класу Apple можуть існувати і після знищення об'єкта класу AppleList.

```
class AppleList
{
	private Apple[ ] lst;

	//Конструктор
	public AppleList(int n)
	{
	   lst = new Apple[n];
	}

	public void addApple(Apple apl, int index)
	{
	   lst[index] = apl;
	}
	private double calculateAverageVolume()
	{
	   double sum = 0;
	   for (int i = 0; i < lst.length; i++)
	      sum += lst[i].calculateVolume();
	   return (sum/lst.length);
	}

	public void printTable()
	{
	   System.out.println("Цвет\tСорт\tДіаметр\tОб'єм");
	   foreach (Apple A in lst)
	   {
	  System.out.format("%1$\t%2$\t%3$\t%4$10.4f}", A.getSort(), A.getColor(),
A.getDiameter(), A.calculateVolume());
	   }
	  System.out.format("\nРазом:\t\t\t%1$6.2f}",  calculateAverageVolume());
	}
}
class Program
{
	public static void main(String[ ] args)
	{
	   java.util.Scanner sc = new java.util.Scanner(System.in);
	   System.out.print("Введіть кількість яблук: ");
	   int n = sc.nextInt();
	   AppleList vedomost = new AppleList(n);
	   String sort, color;
	   double diameter;
	  Apple apl;
	   for (int i = 0; i < n; i++)
```

```
        {
            System.out.print("Введіть сорт яблука: ");
            sort = sc.next();
            System.out.print ("Введіть колір яблука: ");
            color = sc.next();
            System.out.print("Введіть діаметр яблука: ");
            diameter = sc.nextDouble();
            System.out.println();
    apl = new Apple(sort, color, diameter);
            vedomost.addApple(apl, i);
        }
        sc.close();
        System.out.println();
        vedomost.printTable();
        System.out.println();
    }
}
```

Приклад реалізації відношення композиції (один об'єкт класу AppleList може містити кілька об'єктів класу Apple) наведено нижче (C#). У цьому прикладі об'єкти класу Apple створюються всередині класу AppleList, тобто об'єкти класу Apple можуть існувати лише доти, доки існує об'єкт класу AppleList.

```
class AppleList
{
    private Apple [ ] lst;

    // Конструктор
    public AppleList (int n)
    {
        lst = new Apple [n];
    }

    public void AddApple (string sort, string color, double diameter, int index)
    {
        lst [index] = new Apple (sort, color, diameter);
    }
    public double CalculateAverageVolume ()
    {
        double sum = 0;
        for (int i = 0; i <lst.Length; i ++)
            sum + = lst [i] .CalculateVolume ();
        return (sum / lst.Length);
    }
```

```
    public void PrintTable ()
    {
        Console.WriteLine ( "Колір\tСорт\tДіаметр\tОб'єм");
        foreach (Apple A in lst)
        {
            Console.WriteLine ( "{0}\t{1}\t{2}\t{3,10: f4}", A.GetSort (), A.GetColor (),
A.GetDiameter (), A.CalculateVolume ());
        }
       Console.WriteLine ( "\nРазом:\t\t\t{0,6: f2}", CalculateAverageVolume ());
    }
}

class Program
{
    static void Main (string [ ] args)
    {
        Console.Write ( "Введіть кількість яблук:");
        int n = int.Parse (Console.ReadLine ());
        AppleList Vedomost = new AppleList (n);
        string sort, color;
        double diameter;
        for (int i = 0; i <n; i ++)
        {
            Console.Write ( "Введіть сорт яблука:");
            sort = Console.ReadLine ();
            Console.Write ( "Введіть колір яблука:");
            color = Console.ReadLine ();
            Console.Write ( "Введіть діаметр яблука:");
            diameter = double.Parse (Console.ReadLine ());
            Console.WriteLine ();
            Vedomost.AddApple (sort, color, diameter, i);
        }
        Console.WriteLine ();
        Vedomost.PrintTable ();
        Console.WriteLine ();
    }
}
```

Порядок виконання лабораторної роботи

Використовуючи об'єктні моделі, розроблені в лабораторній роботі 2, виконайте:

- для рівня 1 – завдання 1 мовою Java або C#;
- для рівня 2 – завдання 2 мовою Java або C#;
- для рівнів 3 і 4 – завдання 1 та завдання 2 мовою Java або C#.

Завдання 1

Розробіть застосунок мовою Java або C#, призначений для автоматизації предметної області відповідно до варіанта завдання.

Для цього необхідно створити два класи: Клас 1 і Клас 2 (конкретні імена Класу 1 і Класу 2 залежать від предметної області за варіантом завдання. Варіанти завдання слід взяти з лабораторної роботи 2).

Клас 1 призначений для опису будь-якої із записів відомості. Поля цього класу мають відповідати полям відомості, які призначені для зберігання початкових даних. Отримання поточних значень полів, розрахунки за формулами мають виконуватися за допомогою відповідних нестатичних методів цього класу. Для установки значень полів слід використовувати конструктор з параметрами.

Клас 2 має містити статичні методи:

метод main() – точка входу в програму;

необхідну кількість методів для підрахунку підсумкових даних відомості (рівень 4);

метод для виведення всіх даних відомості на консоль.

У методі main() має бути визначений масив об'єктів типу Клас 1 для зберігання відповідних об'єктів.

Застосунок має забезпечувати:

1. Можливість використання текстового меню для вибору варіантів дій користувача (рівні 2 – 4).

2. Введення з консолі кількості записів, що будуть міститися у відомості.

3. У циклі – введення з консолі значень початкових полів кожного запису відомості; створення об'єктів Класу 1, кількість яких відповідає кількості записів відомості; додавання об'єктів Класу 1 у масив.

4. Виведення на консоль початкових і розрахункових даних кожного запису відомості у вигляді:

• необхідної кількості колонок із заголовками (рівні 1 і 2);

• справжньої таблиці з горизонтальними та вертикальними лініями сітки (числові значення повинні виводитися з певною кількістю знаків після коми з використанням засобів форматного виведення) – рівні 3 і 4.

5. Обчислення і виведення на консоль підсумкових даних стовпців відомості (рівень 4).

6. Запобігання появі помилок при виконанні програми (рівень 4).

Завдання 2

Перетворіть застосунок для оброблення відомості (див. завдання 1) так, щоб в ньому використовувалося відношення агрегації або композиції, а його функціональні можливості залишилися тими самими.

Він має містити три класи:

1. Клас – "ціле", що описує відомість з безліччю записів.

2. Клас – "частина", який описує будь-який запис відомості.

3. Клас, що містить головний метод програми, в якому має створюватися один об'єкт класу – "ціле" і з використанням посилання на цей об'єкт викликаються методи класу "ціле".

Контрольні запитання до лабораторної роботи 3

1. Що таке клас в ООП?
2. Які є принципи об'єктно-орієнтованого підходу?
3. Який синтаксис опису класу?
4. В чому полягають особливості статичних елементів класу?
5. Які є специфікатори доступу до елементів класу у C#?
6. Який порядок ініціалізації об'єкта класу?
7. Яке призначення та використання посилання this?
8. Що таке агрегація та композиція?

Лабораторна робота 4
Застосування успадкування та поліморфізму

Мета лабораторної роботи:

1. Удосконалення практичних навичок з розроблення класів мовами C# і Java.

2. Набуття практичних навичок застосування успадкування та поліморфізму в мовах C# і Java.

3. Удосконалення навичок роботи з інтегрованими середовищами MS Visual Studio і Eclipse.

Перед виконанням лабораторної роботи студент має:

знати:

синтаксис опису класу;

принципи перевизначення методів;

синтаксис успадкування в C# і Java;

порядок виклику конструкторів у разі успадкування;

принципи реалізації поліморфізму в C# і Java;

основи використання абстрактних класів та інтерфейсів у C# і Java;

вміти застосовувати механізми успадкування та поліморфізму при розробленні програм мовами C# і Java.

Необхідно виконати:

для рівня 1 – завдання 1 мовою Java або C#;

для рівня 2 – завдання 2 мовою Java або C#;

для рівнів 3 і 4 – завдання 1 та 2 мовою Java або C#.

Загальні вимоги. Необхідно розробити UML-діаграму класів, яка відповідає варіанту виконаного завдання (рівні 2 – 4).

Завдання 1

Використовуючи принцип успадкування, розробіть програму, яка надає користувачеві функціональність калькулятора, що має два режими роботи: стандартний та інженерний.

У стандартному режимі калькулятор може виконувати будь-яку з чотирьох арифметичних операцій: "+", "–", "×", "/" з двома дійсними числами. У інженерному режимі крім арифметичних операцій він також може виконувати додаткову операцію, яка визначається варіантом завдання (див. табл. 1.1). Поточний режим роботи програми вибирає користувач програми.

Програма має забезпечувати:

1. Створення за одним об'єктом класів 1 і 2.

2. Одноразове введення вихідних даних, отримання результату обчислень у вибраному режимі роботи, виведення його на екран (рівень 1).

3. Багаторазове повторення процесу введення початкових даних, отримання результату обчислень у вибраному режимі роботи та виведення його на екран поки користувач не введе певний рядок символів (рівні 2 – 4).

4. Виконання оброблення наступних помилкових ситуацій: "поділ на нуль", "вихідні дані для виконання додаткової операції не відповідають обмеженням варіанта завдання" (рівень 4).

Функціональність програми має бути реалізована в трьох класах: **Клас 1**, **Клас 2**, **Клас 3**, які повинні знаходитися в різних файлах проєкту.

Клас 1 є базовим і призначений для опису обчислювального модуля "стандартного" режиму. Поля цього класу мають відповідати початковим даним для виконання арифметичних операцій. Для встановлення значень

полів і отримання їхніх поточних значень має використовуватися відповідні властивості (C#) або методи set ... і get ... (Java). Цей клас має **конструктор без параметрів**. Виконання кожної з арифметичних операцій має виконуватися за допомогою відповідного нестатичного методу цього класу.

Клас 2 є спадкоємцем **Класу 1** і призначений для опису обчислювального модуля інженерного режиму. Цей клас за необхідності має містити одне або кілька полів (для встановлення значень полів і отримання їхніх поточних значень мають використовуватися відповідні властивості (C#) або методи set ... і get ... (Java)). В цьому класі **має бути конструктор без параметрів**. Додаткова операція інженерного калькулятора має бути реалізована у вигляді нестатичного методу даного класу. Варіанти наведено у табл. 1.1.

Клас 3 містить головний метод програми.

Рекомендовану структуру класів програми (метод Pow() призначений для обчислення функції ab; # – protected) наведено на рис. 4.1.

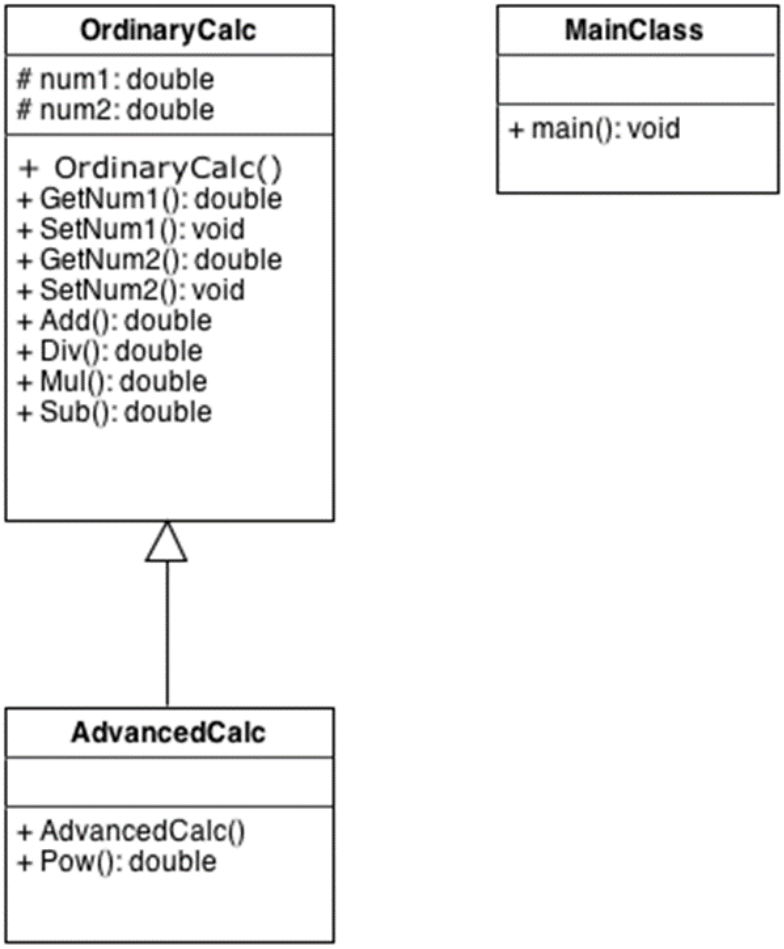

Рис. 4.1. **Рекомендована структура класів (завдання 1)**

Завдання 2

Перетворіть програму з завдання 1 у такий спосіб, щоб у ній використовувалися абстрактний клас і інтерфейс. Нова програма має працювати також, як програма з завдання 1 за відповідним варіантом.

Функціональність програми має бути реалізована в одному абстрактному класі, одному інтерфейсі та трьох звичайних класах.

Рекомендовану структуру класів програми наведено на рис. 4.2.

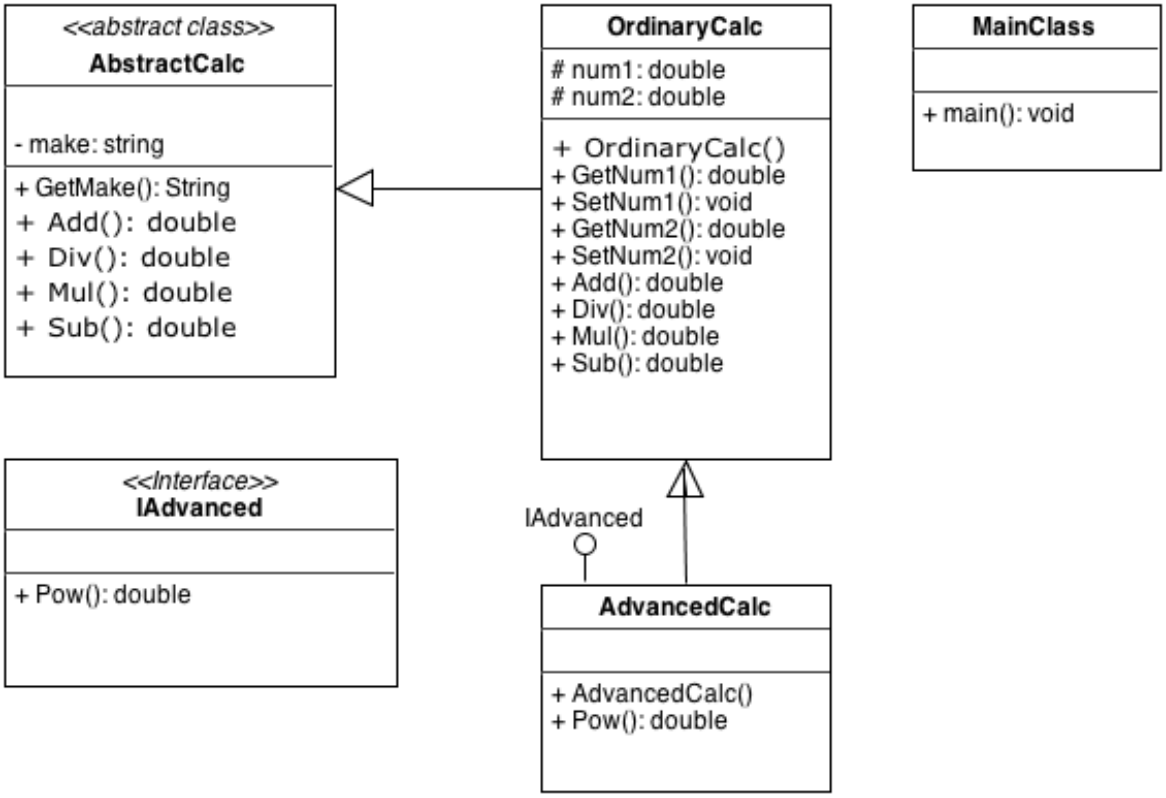

Рис. 4.2. **Рекомендована структура класів (завдання 2)**

Абстрактний клас AbstractCalc має поле – модель калькулятора і метод для отримання поточного значення цього поля, а також абстрактні методи для опису операцій обчислювального модуля стандартного режиму, тобто арифметичних операцій.

Клас OrdinaryCalc обчислювального модуля стандартного режиму є спадкоємцем абстрактного класу AbstractCalc і перевизначає його методи. В іншому його сенс той самий, що і в завданні 1. Методи обчислювального модуля стандартного режиму мають викликатися поліморфним чином, тобто через посилання на базовий клас AbstractCalc.

Інтерфейс IAdvanced призначений для опису додаткової операції обчислювального модуля інженерного режиму.

Клас AdvancedCalc обчислювального модуля інженерного режиму є спадкоємцем класу обчислювального модуля стандартного режиму і реалізує інтерфейс IAdvanced. Додаткова операція обчислювального модуля інженерного режиму має викликатися поліморфним чином, тобто через посилання на інтерфейс IAdvanced.

У застосунку також є клас MainClass, у якому знаходиться головний метод.

Контрольні запитання до лабораторної роботи 4

1. Які приклади відношення успадкування можна навести?
2. Який синтаксис успадкування в C# і Java?
3. Якою є послідовність виклику конструкторів у спадкуванні?
4. Як перевизначити базовий метод у C# і Java?
5. Яким чином реалізовано принцип поліморфізму в C# і Java?
6. Які є особливості використання абстрактних класів у C# і Java?
7. Які є особливості використання інтерфейсів у C# і Java.

Лабораторна робота 5
Розроблення об'єктно-орієнтованих застосунків мовою Python

Мета лабораторної роботи:

1. Набуття практичних навичок з розробки класів мовою Python.
2. Набуття практичних навичок застосування успадкування у мові Python.
3. Удосконалення навичок роботи з інтегрованим середовищем PyCharm.

Перед виконанням лабораторної роботи студент має:

знати:

синтаксис опису класу;

перевизначення методів;

синтаксис успадкування;

принципи реалізації успадкування та поліморфізму;

основи використання абстрактних класів в Python;

вміти застосовувати механізми успадкування при розробленні програм мовою Python.

Завдання 1

Використовуючи принцип успадкування, розробіть програму, яка надає користувачеві функціональність калькулятора, що має два режими роботи: стандартний та інженерний.

У стандартному режимі калькулятор може виконувати будь-яку з чотирьох арифметичних операцій: "+", "–", "×", "/" з двома дійсними числами. У "інженерному" режимі крім арифметичних операцій він також може виконувати додаткову операцію, яка визначається варіантом завдання (див. табл. 1.1). Поточний режим роботи програми вибирає користувач програми.

Функціональність програми має бути реалізована в двох класах: **OrdinaryCalc** та **AdvancedCalc**, а також **головної функції**, які мають знаходитися в різних файлах проєкту.

Клас **OrdinaryCalc** є базовим і призначений для опису обчислювального модуля стандартного режиму. Поля цього класу мають відповідати початковим даним для виконання арифметичних операцій. Для встановлення значень полів і отримання їхніх поточних значень мають використовуватися відповідні властивості або методи set ... і get ... . Цей клас має **конструктор без параметрів**. Виконання кожної з арифметичних операцій має виконуватися за допомогою відповідного нестатичного методу цього класу.

Клас **AdvancedCalc** є спадкоємцем класу **OrdinaryCalc** і призначений для опису обчислювального модуля інженерного режиму. В цьому класі **має бути конструктор без параметрів**. Додаткова операція інженерного калькулятора має бути реалізована у вигляді нестатичного методу цього класу. Варіанти завдань наведено у табл. 1.1.

Програма має забезпечувати:

1. Створення в процесі її роботи тільки по одному об'єкту класів **OrdinaryCalc** і **AdvancedCalc**.

2. Одноразове введення початкових даних, отримання результату обчислень у вибраному режимі роботи, виведення його на екран (рівень 1).

3. Багаторазове повторення процесу введення початкових даних, отримання результату обчислень у вибраному режимі роботи і виведення його на екран, поки користувач не введе певний рядок символів (рівень 2).

4. Виконання оброблення таких помилкових ситуацій: "ділення на нуль", "початкові дані для виконання додаткової операції не відповідають обмеженням варіанта завдання" з використанням винятків (рівні 3 і 4).

Контрольні запитання до лабораторної роботи 5

1. У чому полягає принцип успадкування?
2. Який синтаксис успадкування в Python?
3. Як перевизначити базовий метод у Python?
4. Яким чином реалізовано принцип поліморфізму у Python?
5. Які є особливості використання абстрактних класів у Python?

Лабораторна робота 6
Використання шаблонів проєктування та бібліотек класів

Мета лабораторної роботи:

1. Придбання практичних навичок застосування шаблонів проєктування та бібліотек класів під час розроблення програм.

2. Удосконалення навичок роботи з інтегрованими середовищами Eclipse і Microsoft Visual Studio.

Перед виконанням лабораторної роботи студент має:

знати:

загальні відомості про шаблони проєктування та бібліотеки класів;

класифікацію шаблонів проєктування;

основи використання основних шаблонів проєктування;

вміти застосовувати основні шаблони проєктування та бібліотеки класів під час розроблення програм.

Шаблони проєктування

У розробленні програмного забезпечення, шаблон проєктування або патерн (design pattern) – повторювана архітектурна конструкція, що є рішенням проблеми проєктування в рамках деякого часто виникаючого контексту.

Зазвичай шаблон не є закінченим зразком, який може бути прямо перетворений у код. Це лише приклади розв'язання задач, які можна використовувати в різних ситуаціях. Об'єктно-орієнтовані шаблони показують відносини і взаємодії між класами або об'єктами, без визначення того, які кінцеві класи або об'єкти будуть використовуватися.

Класифікація шаблонів проєктування (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 407 p.):

1. Твірні шаблони – відносяться до створення екземплярів класів.

2. Структурні шаблони – відносяться до створення з класів і об'єктів більших структур.

3. Поведінкові шаблони – відносяться до розподілу обов'язків між об'єктами і типовим способам їх взаємодії.

Твірні шаблони, які використовуються найчастіше:

1. Абстрактна фабрика (Abstract Factory) – створює екземпляр декількох сімейств класів.

2. Фабричний метод (Factory Method) – створює екземпляр декількох похідних класів.

3. Одинак (Singleton) – може існувати тільки один екземпляр такого класу.

Приклад використання шаблону "Одинак" наведено на рис. 6.1.

Класи-учасники: Earth.

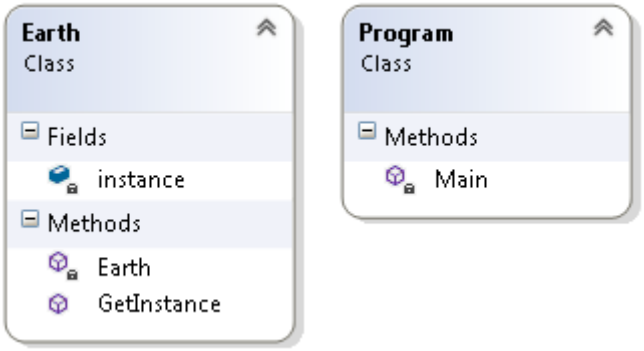

Рис. 6.1. **Діаграма класів програми (шаблон "Одинак")**

Код класів програми мовою C#:

```
class Earth
{
  private static Earth instance;
  private Earth()
  {
    Console.WriteLine("Об'єкт створено...");
  }
```

```
  public static Earth GetInstance()
  {
     if (instance == null)
     {
          instance = new Earth();
       }
       return instance;
  }
}

class Program
{
   static void Main(string[] args)
   {
      Earth e1 = Earth.GetInstance();
   }
}
```

Структурні шаблони, які використовуються найчастіше:

1. Адаптер (Adapter) – "стикує" інтерфейси різних класів.

2. Заступник (Proxy) – об'єкт, який контролює доступ до іншого об'єкта, перехоплюючи всі виклики до нього.

3. Компонувальник (Composite) – деревоподібна структура простих і складних об'єктів.

4. Фасад (Facade) – єдиний клас, який представляє всю підсистему.

Діаграму класів шаблону "Заступник" наведено на рис. 6.2.

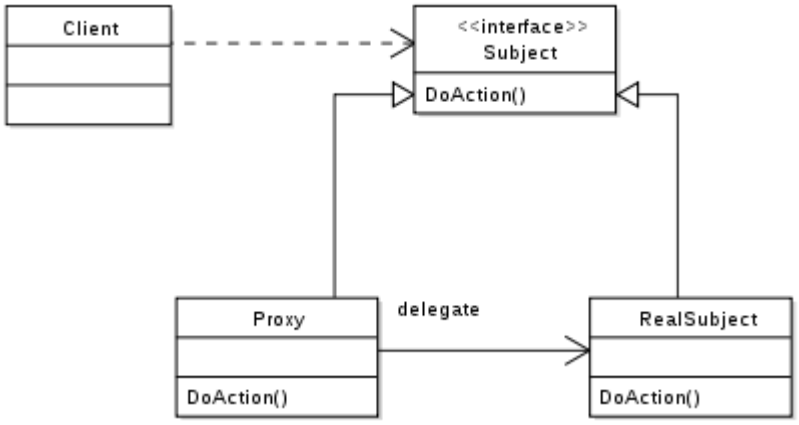

Рис. 6.2. **Шаблон "Заступник"**

Тут Subject – інтерфейс, у якому є метод DoAction().

RealSubject – клас, який реалізує інтерфейс Subject і перевизначає метод DoAction() для виконання деякої конкретної дії.

Proxy – клас, який також реалізує інтерфейс Subject і перевизначає метод DoAction() у такий спосіб, що в ньому відбувається виклик методу DoAction() класу RealSubject.

Client – клас, який використовує посилання на інтерфейс Subject для створення об'єкта класу Proxy.

Приклад використання шаблону "Заступник" наведено на рис. 6.3.

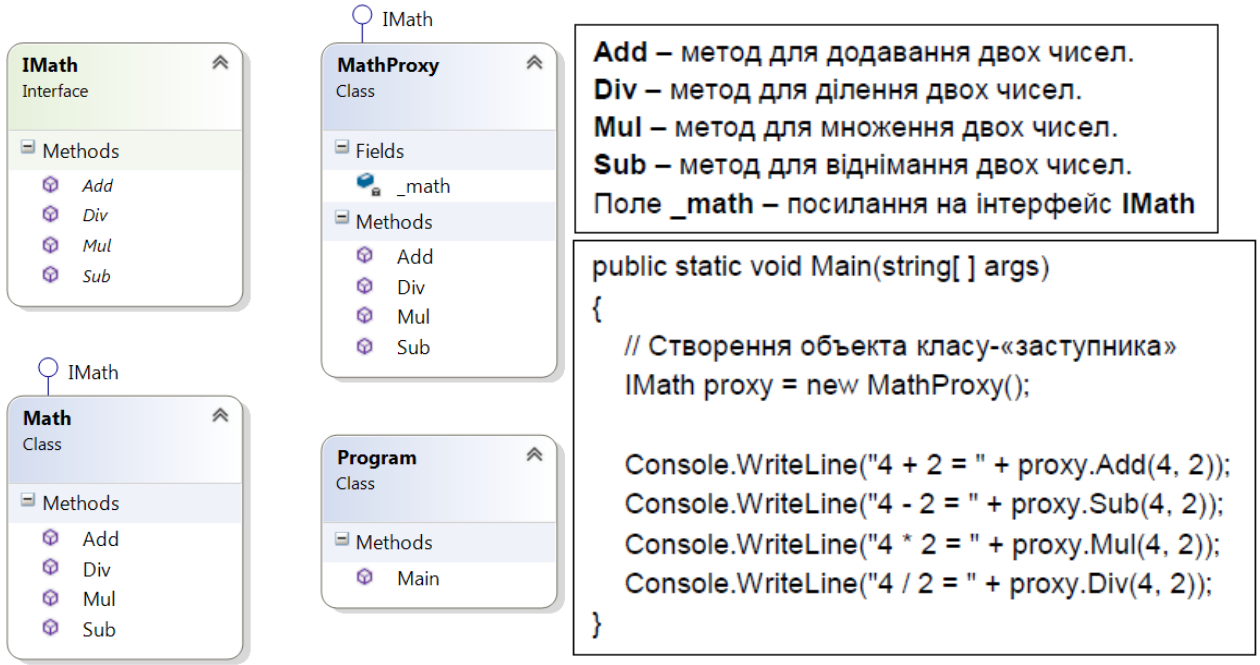

Рис. 6.3. **Реалізація шаблона "Заступник"**

Класи-учасники (програма "Калькулятор"):

1. Інтерфейс IMath – виконує роль Subject.
2. Клас Math – виконує роль RealSubject.
3. Клас MathProxy – виконує роль Proxy.
4. Клас Program – виконує роль Client.

Розроблення бібліотек на платформі Java SE та Microsoft .NET

Бібліотека класів Java – це набір бібліотек, що можуть завантажуватися динамічно та викликатися програмою під час її виконання. Оскільки платформа Java не залежить від конкретної операційної системи, програми не можуть покладатися на платформо-залежні бібліотеки. Натомість

платформа Java надає набір стандартних бібліотек класів, що містять звичайні для сучасних операційних систем функції.

Java-бібліотеки постачаються у вигляді jar-файлів.

JAR (Java ARchive) – це формат архівного файла, який зазвичай використовується для об'єднання багатьох файлів класів Java та пов'язаних метаданих і ресурсів в один файл для поширення прикладного програмного забезпечення або бібліотек на платформі Java. jar-файли базуються на форматі ZIP і мають розширення .jar.

Можна розробляти власні Java-бібліотеки. Приклад створення бібліотеки в інтегрованому середовищі Eclipse наведено далі.

Створимо просту математичну бібліотеку, що складається з одного класу. Клас має чотири метода для виконання арифметичних операцій та два метода для завдання значень операндів. Вихідний код цього класу має такий вигляд:

```
package librarypackage;
public class LibraryClass {
    private double num1, num2;
    public LibraryClass() { }
    public void setNum1(double num1) {
        this.num1 = num1;
    }
    public void setNum2(double num2) {
        this.num2 = num2;
    }
    public double add() {
        return num1 + num2;
    }
    public double sub() {
        return num1 - num2;
    }
    public double mul() {
        return num1 * num2;
    }
    public double div() {
        return num1 / num2;
    }
}
```

1. Створить новий проєкт Java.
2. Додайте до нього клас LibraryClass (рис. 6.4).

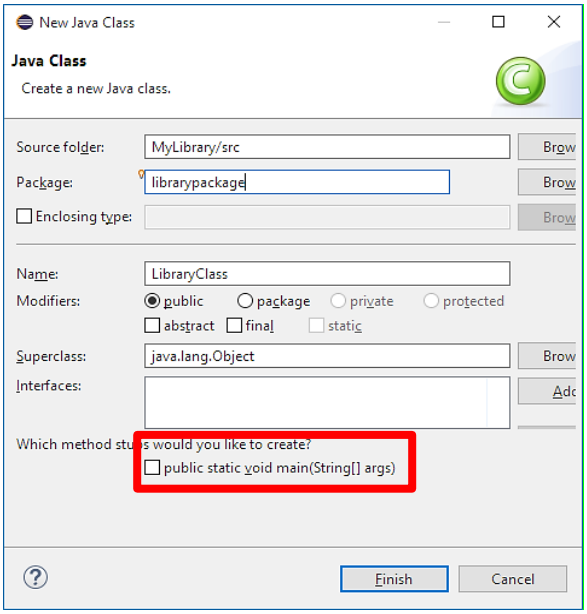

Рис. 6.4. **Створення класу бібліотеки**

3. Експортуйте проєкт бібліотеки в jar-файл (рис. 6.5).

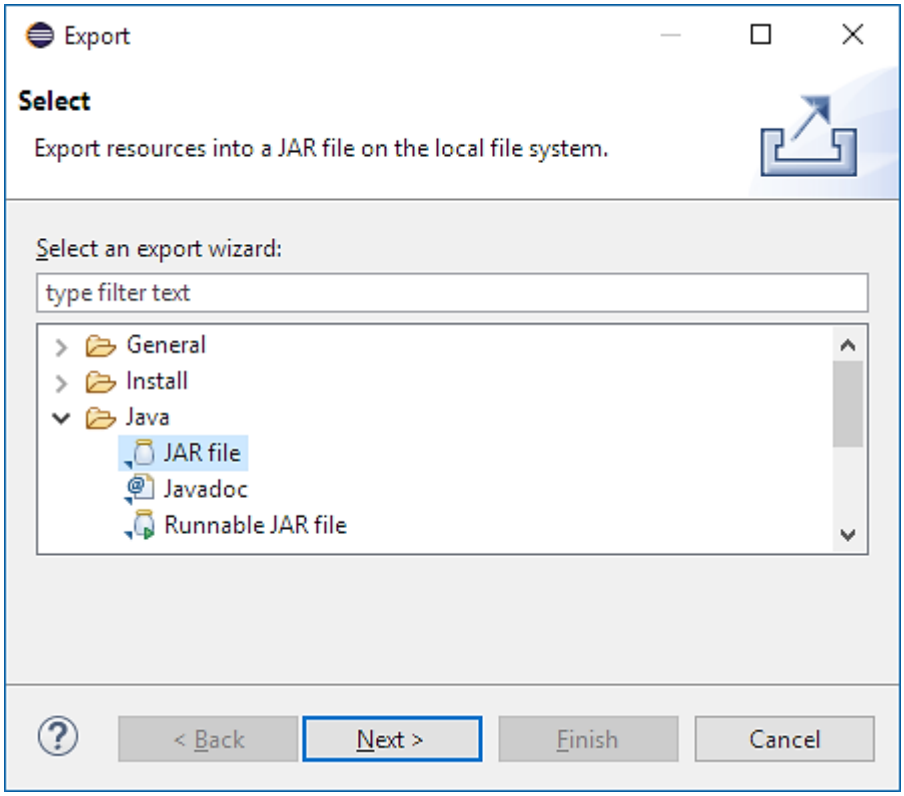

Рис. 6.5. **Вікно експорту**

4. Додайте посилання на jar-файл у цільовому застосунку (рис. 6.6).

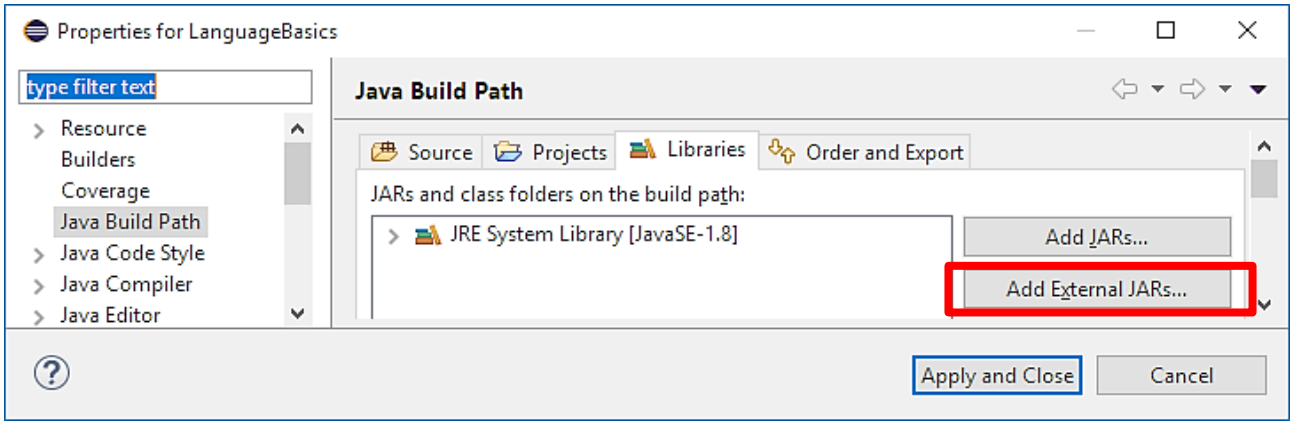

Рис. 6.6. **Вікно додавання посилання на jar-файл**

Вихідний код цільового застосунку наведено далі:

```
package calculator;
import librarypackage.*;
public class Calculator {
    public static void main(String[ ] args) {
        LibraryClass cB = new LibraryClass();
        double n1 = 45.9, n2 = 4.67;
        cB.setNum1(n1);
        cB.setNum2(n2);
        System.out.format("Сума=%1$f5.2", cB.add());
    }
}
```

Розглянемо створення dll-бібліотеки на платформі MS .NET в інтегрованому середовищі Visual Studio 2019.

Для цього необхідно створити новий проєкт типу Class Library (.NET Framework) – рис. 6.7, а потім додати до нього вихідний код класу LibraryClass мовою C#, що є аналогом відповідного вихідного коду мовою Java, наведеного раніше. Після компіляції цього коду отримаємо файл dll-бібліотеки.

У цільовому застосунку додайте посилання на розроблену dll-бібліотеку (рис. 6.8). У вихідний код цільового застосунку, аналогічний коду цільового застосунку мовою Java, потрібно додати інструкцію using для простору імен dll-бібліотеки.

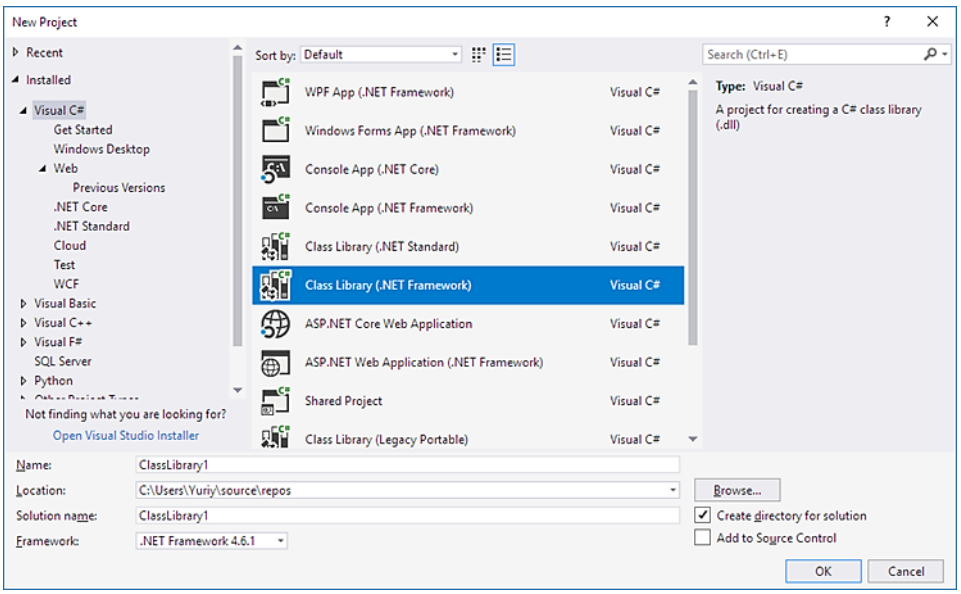

Рис. 6.7. **Вікно створення нового проєкту**

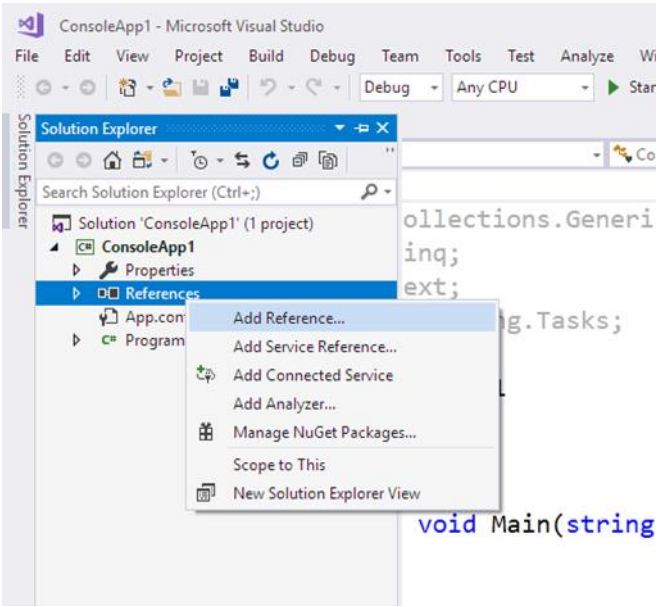

Рис. 6.8. **Вікно додавання посилання на dll-бібліотеку**

Порядок виконання лабораторної роботи

Виконайте одне із завдань мовою Java або C#.

Завдання 1

Розробіть програму, яка надає користувачеві функціональність калькулятора, що виконує будь-яку з чотирьох арифметичних операцій: "+", "–", "×", "/" з двома дійсними числами. Програма має забезпечувати багаторазове повторення процесу введення початкових даних, отримання

результату обчислень і виведення його на екран до введення користувачем певного рядка символів.

Діаграма класів застосунку має відповідати шаблону "Заступник" (див. рис. 6.3).

Обчислювальний модуль, що відповідає за виконання всіх операцій калькулятора та використовується у головному класі програми, розробіть у вигляді бібліотеки класів .jar або .dll.

Завдання 2

Розробіть програму, яка надає користувачеві функціональність калькулятора, що виконує будь-яку з чотирьох арифметичних операцій: "+", "–", "×", "/" з двома дійсними числами, а також додаткову операцію відповідно до варіанта завдання, наведеному в табл. 1.1. Програма має забезпечувати багаторазове повторення процесу введення вихідних даних, отримання результату обчислень і виведення його на екран до введення користувачем певного рядка символів.

Діаграма класів застосунку має відповідати шаблону "Заступник" (див. рис. 6.3). Клас MathProxy має бути "Одинаком" (див. рис. 6.1).

Обчислювальний модуль, що відповідає за виконання всіх операцій калькулятора та використовується у головному класі програми, розробіть у вигляді бібліотеки класів .jar або .dll.

Завдання 3

Розробіть програму, яка надає користувачеві функціональність калькулятора, що виконує будь-яку з чотирьох арифметичних операцій: "+", "–", "×", "/" з двома дійсними числами, а також додаткову операцію відповідно до варіанта завдання, наведеному в табл. 1.1. Програма має забезпечувати багаторазове повторення процесу введення вихідних даних, отримання результату обчислень і виведення його на екран до введення користувачем певного рядка символів. У програмі необхідно передбачити оброблення таких помилкових ситуацій:

1. "Ділення на нуль" – з використанням відповідного бібліотечного класу винятку.

2. "Вихідні дані для виконання додаткової операції не відповідають обмеженням варіанта завдання" – з використанням користувацького класу винятку.

Діаграма класів застосунку повинна відповідати шаблону "Заступник". Вона переважно аналогічна діаграмі класів, яка приведена на рис. 6.3,

за винятком того, що методи для обчислення арифметичних операцій і обчислення додаткової операції необхідно розподілити між двох інтерфейсів, які реалізуються в класах Math і MathProxy. Клас MathProxy повинен бути "Одинаком" (див. рис. 6.1).

Обчислювальний модуль, що відповідає за виконання всіх операцій калькулятора та використовується у головному класі програми, розробіть у вигляді бібліотеки класів .jar або .dll.

Контрольні запитання до лабораторної роботи 6

1. Для чого використовують шаблон "Абстрактна фабрика"?
2. Для чого використовують шаблон "Фабричний метод"?
3. Для чого використовують шаблон "Адаптер"?
4. Для чого використовують шаблон "Компонувальник"?
5. Для чого використовують шаблон "Фасад"?
6. Для чого використовують шаблон "Команда"?
7. Для чого використовують шаблон "Спостерігач"?
8. Для чого використовують шаблон "Стратегія"?

Лабораторна робота 7
Використання колекцій

Мета лабораторної роботи:

1. Вдосконалення практичних навичок з розроблення класів.
2. Придбання практичних навичок використання класів колекцій.
3. Удосконалення навичок роботи з інтегрованим середовищем MS Visual Studio і Eclipse.

Перед виконанням лабораторної роботи студент має:

знати основні структури даних бібліотеки колекцій платформи Microsoft .NET і Java SE і принципи їх використання;

вміти використовувати бібліотеки колекцій Microsoft .NET і Java SE під час розроблення програм.

Порядок виконання лабораторної роботи

Завдання. За основу рекомендується взяти програму зі свого варіанта з лабораторної роботи 3, у якій використовується відношення агрегації або композиції.

Розробіть програму мовою Java або C# для оброблення відомості. Варіанти класів колекцій наведено в табл. 7.1.

Таблиця 7.1

Варіанти завдань

№ з/п	Клас колекції Microsoft .NET	Клас колекції Java SE
1	Hashtable	HashSet<T>
2	ArrayList	HashMap<K, V>
3	SortedList	ArrayList<T>
4	List<T>	TreeSet<T>
5	LinkedList<T>	TreeMap<K, V>
6	HashSet<T>	LinkedList<T>
7	Dictionary<K, V>	LinkedHashSet<T>
8	SortedDictionary <K, V >	ArrayList
9	SortedList<K, V>	HashSet<T>
10	Hashtable	HashMap<K, V>
11	SortedSet<T>	LinkedHashMap<K, V>
12	SortedList	TreeSet<T>
13	HashSet<T>	TreeMap<K, V>
14	Dictionary<K, V>	LinkedList<T>
15	SortedList<K, V>	LinkedHashSet<T>

Інформацію щодо предметної області взяти за варіантами з лабораторної роботи 1.

Для виконання завдання необхідно створити:

А. "Клас 1" – призначений для опису будь-якого із записів відомості. В цьому класі мають бути: поля, відповідні полям відомості, які призначені для зберігання початкових даних; конструктор з параметрами; необхідні нестатичні методи та властивості.

Б. "Клас 2" – містить колекцію для зберігання даних відомості, а також методи, призначені для:

додавання даних у колекцію;

виведення даних відомості з колекції на консоль;

фільтрації даних, що зберігаються в колекції, за певними реквізитами.

В. "Клас 3" – це клас, що містить головний метод програми.

Кожний з класів має знаходитися в окремому файлі (рівні 3 і 4).

Програма має забезпечувати:

1. Вибір варіантів дій користувача за допомогою текстового меню (рівні 2 – 4).

2. Введення з консолі початкових даних кожного із записів відомості в поточному сеансі роботи.

3. Створення для кожного запису відомості об'єкта класу 1; подальше додавання кожного такого об'єкта в колекцію, яка відповідає варіанту завдання.

4. Виведення початкових і розрахункових даних кожного запису відомості з колекції, а також підсумкової інформації відомості на консоль у вигляді необхідної кількості колонок із заголовками (рівні 1 і 2) або у вигляді справжньої таблиці даних, з горизонтальними і вертикальними лініями сітки з використанням засобів форматного виведення – рівні 3 і 4.

5. Можливість фільтрації записів відомості за певними реквізитами – рівні 3 і 4.

Контрольні запитання до лабораторної роботи 7

1. Охарактеризуйте склад бібліотеки контейнерів Java SE та Microsoft .NET.

2. Які основні інтерфейси існують у бібліотеці контейнерів Java SE та Microsoft .NET?

3. Дайте стислу характеристику структури даних "динамічний масив" і її реалізації в Java SE та Microsoft .NET.

4. Дайте стислу характеристику структури даних "двозв'язний список" і її реалізації в Java SE та Microsoft .NET.

5. Що таке "геш-таблиця" і "геш-функція"?

6. Дайте стислу характеристику структури даних "асоціативний масив" і її реалізації в Java SE та Microsoft .NET.

Лабораторна робота 8
Використання лямбда-виразів, LINQ та Stream API

Мета лабораторної роботи:

1. Вдосконалення практичних навичок з використання колекцій.

2. Придбання практичних навичок використання лямбда-виразів, LINQ та Stream API.

3. Удосконалення навичок роботи з інтегрованим середовищем MS Visual Studio і Eclipse.

Перед виконанням лабораторної роботи студент має:

знати основи лямбда-виразів, LINQ та Stream API;

вміти використовувати лямбда-вирази, LINQ та Stream API під час розроблення програм.

Завдання 1

Додайте в програму мовою Java або C# для оброблення відомості (див. завдання лабораторної роботи 7) можливість використання запитів до певної колекції з використанням лямбда-виразів і Stream API (або LINQ to Objects).

Програма має за вимогою користувача виводити результати кожного запиту на консоль у визначеному форматі.

Кожен з класів має знаходитися в окремому файлі (рівні 3 і 4).

Види запитів:

1. Отримати кількість записів відомості.
2. Отримати всі записи відомості, відсортовані за певним реквізитом (за зростанням та спаданням).
3. Отримати записи відомості, де певне текстове поле починається з деякої літери.
4. Отримати записи відомості, де значення одного або декількох полів знаходяться в певному діапазоні.
5. Отримати записи відомості, в полях яких є тільки унікальні значення.
6. Отримати певну кількість записів з початку відомості.
7. Отримати записи відомості, де деяке числове поле має певне значення.
8. Отримати записи відомості, де деяке числове поле має максимальне значення.
9. Отримати записи відомості, де деяке числове поле має мінімальне значення.
10. Отримати значення певної сукупності полів для всіх записів відомості.

Варіанти класів колекцій наведено у табл. 8.1.

Таблиця 8.1

Варіанти завдань

№ з/п	Клас колекції Microsoft .NET	Клас колекції Java SE
1	2	3
1	Hashtable	HashSet<T>
2	ArrayList	HashMap<K, V>
3	SortedList	ArrayList<T>
4	List<T>	TreeSet<T>
5	SortedList <T>	TreeMap<K, V>

Закінчення табл. 8.1

1	2	3
6	HashSet<T>	LinkedList<T>
7	Dictionary<K, V>	LinkedHashSet<T>
8	SortedDictionary <K, V >	ArrayList
9	SortedList<K, V>	HashSet<T>
10	Hashtable	HashMap<K, V>
11	SortedSet<T>	LinkedHashMap <K, V>
12	SortedList	TreeSet<T>
13	HashSet<T>	TreeMap<K, V>
14	Dictionary<K, V>	LinkedList<T>
15	SortedList<K, V>	LinkedHashSet<T>

Інформацію щодо предметної області взяти за варіантами з лабораторної роботи 1.

Контрольні запитання до лабораторної роботи 8

1. Що таке "функціональні інтерфейси" і який їх синтаксис?
2. Що таке "лямбда-вираз"?
3. Для чого використовується мова інтегрованих запитів LINQ?
4. Як LINQ використовує лямбда-вирази?
5. Які є способи створення Stream API?
6. Які існують методи роботи зі Stream API?

Лабораторна робота 9
Використання регулярних виразів

Мета лабораторної роботи:

1. Придбання практичних навичок використання регулярних виразів для обробки текстової інформації в програмах мовами C# і Java.

2. Удосконалення навичок роботи з інтегрованими середовищами MS Visual Studio, Eclipse.

Перед виконанням лабораторної роботи студент має:

знати принципи створення і використання регулярних виразів;

вміти самостійно використовувати механізм регулярних виразів під час розроблення програм розбору текстів і пошуку за зразком у мовах C# і Java.

Регулярні вирази

Регулярні вирази – формальна мова для пошуку та здійснення маніпуляцій з підрядками в тексті, заснований на використанні метасимволів. По суті, це рядок-зразок, що складається з символів і метасимволоів та задає правило пошуку.

Основний клас для оброблення регулярних виразів на платформі Microsoft .NET – це **System.Text.RegularExpressions.Regex**, на платформі Java SE – **java.util.regex.Pattern.**

Пошук у рядках

Напевно, найбільш часто класи Regex і Pattern використовуються для пошуку підрядків, які відповідають деякому шаблону. Вони надають декілька методів пошуку рядків і визначення наявності збігів.

Найпростішим прикладом використання класів Regex і Pattern є перевірка користувальницького вводу, яка просто повідомляє, чи задовольняє вхідна послідовність символів регулярному виразу.

Приклад 1. Фрагмент програми на C# для пошуку телефонних номерів формату **(0XX)XXX-XX-XX**, де X – будь-яка цифра:

```
Console.WriteLine("Введіть номер телефона:");
string str = Console.ReadLine();
 string pattern = @"^\(0\d{2}\)[1-9](\d{2}-){2}\d{2}$";
Regex reg = new Regex(pattern);
if (reg.IsMatch(str))
            Console.WriteLine("Номер записано правильно");
         else
            Console.WriteLine("Неправильний формат номера!!!");
```

Приклад 2. Фрагмент програми на Java для пошуку телефонних номерів формату **(0XX)XXX-XX-XX**, де X – будь-яка цифра:

```
Scanner sc = new Scanner(System.in);
System.out.println("Введіть номер телефона");
String str = sc.next();
String pattern = "^\\(0\\d{2}\\)[1-9](\\d{2}-){2}\\d{2}$";
if (Pattern.matches(pattern, str)) {
   System.out.println("Номер записано правильно\n");
} else {
    System.out.println("Неправильний формат номера!!!\n");
  }
```

Основні символи та послідовності, використовувані в регулярних виразах, наведено в табл. 9.1.

Таблиця 9.1

Основні символи та послідовності, використовувані в регулярних виразах

Символ	Опис
^	Відповідність має виявлятися на початку вхідного рядка
$	Відповідність має виявлятися в кінці вхідного рядка
*	Квантор (відповідає 0 або більше входжень попереднього символу або групи)
+	Квантор (відповідає 1 або більше входжень попереднього символу або групи)
?	Квантор (відповідає 0 або 1 входження попереднього символу або групи)
.	Клас символів (відповідає будь-якому символу, за винятком символу "\n")
[aeiou]	Клас символів (відповідає будь-якому символу з множини, заданої в квадратних дужках)
[^aiou]	Клас символів (відповідає будь-якому символу, за винятком символів, заданих у квадратних дужках)
[0-9] [a-z]	Клас символів (завдання одного символу з діапазону)
\w	Клас символів (один із символів, які використовуються під час завдання ідентифікаторів)
\s	Клас символів (відповідає символу пробілу)
\d	Клас символів (відповідає будь-якому символу з множини цифр)
\W	Клас символів (будь-який з символів, крім використовуваних під час завдання ідентифікаторів)
\S	Клас символів (будь-яке непорожнє місце)
\D	Клас символів (не цифра)
{n}	Квантор (попередній елемент повторюється рівно n раз)
{n,}	Квантор (попередній елемент повторюється мінімум n раз)
{n,m}	Квантор (попередній елемент повторюється мінімум n раз, але не більше ніж m разів)

Загальні відомості про синтаксичні аналізатори

Синтаксичний аналіз – це процес аналізу вхідної послідовності символів з метою розбору її граматичної структури за заданою формальною граматикою. Зокрема, синтаксичний аналіз дозволяє визначити, чи належить вхідна послідовність символів до заданої граматики (наприклад, до граматики деякої мови).

Синтаксичний аналізатор – це програма, яка виконує синтаксичний аналіз.

Під час опису граматик будемо використовувати лінгвістичні формули (розширена форма Бекуса – Наура). Це формальна система визначення синтаксису, в якій одні синтаксичні категорії послідовно визначаються через інші.

Головні метасимволу, використовувані для запису металінгвістичних формул відповідно до розширеної форми Бекуса – Наура:

::= (означає "це");

{} (фігурні дужки використовуються для об'єднання декількох елементів);

{}* (вказує, що вкладений в фігурні дужки елемент формули може бути опущений або повторений довільну кількість раз);

{}+ (вказує, що вкладений в фігурні дужки елемент формули може бути повторений один або довільну кількість раз);

{}? (вказує, що вкладений в фігурні дужки елемент формули може бути повторений один раз або він може бути відсутнім);

| (означає "або").

Завдання 1

Розробіть консольну програму для виконання аналізу вхідної послідовності символів, що вводяться користувачем, з метою визначення її приналежності до граматики, яка описує деяке поняття (згідно з варіантом завдання).

Програма має містити блок синтаксичного аналізу вхідної послідовності символів на базі регулярних виразів.

Якщо вхідна послідовність припустима, то програма має видавати відповідне повідомлення користувачу, інакше – аналогічним чином повідомляти його про неприпустимість вхідної послідовності. У будь-якому випадку далі виконання програми має тривати до введення користувачем рядка-ознаки завершення програми.

Варіант 1

Розробіть синтаксичний аналізатор для поняття **умовний_оператор**:

умовний_оператор::= if (умова) оператор {else оператор}?
оператор::={змінна++;} | {змінна--;}
умова::= змінна {< | >} змінна
змінна::=літера$^{+}$цифра*
літера::a|b|c
цифра ::= 0|1|2|3|4|5|6|7|8|9

Варіант 2

Розробіть синтаксичний аналізатор для поняття **новий_об'єкт**:

новий_об'єкт::= {ім'я_класу змінна=new конструктор;}|{new конструктор;}
конструктор::= ім'я_класу ({змінна {, змінна }*}*)
ім'я_класу::= літера$^{+}$
змінна::= літера$^{+}$цифра*
літера::d|e|f
цифра::= 1|2|3

Варіант 3

Розробіть синтаксичний аналізатор для поняття **просте_логічне**:

просте_логічне::= TRUE | FALSE | {простий_ідентифікатор
знак_операції простий_ідентифікатор }
простий_ідентифікатор::= літера
знак_операції::= AND | OR
літера::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

Варіант 4

Розробіть синтаксичний аналізатор для поняття **відношення**:

відношення::= операнд знак_ відношення операнд
знак_ відношення::= $<|>|=$
операнд::= константа $|$змінна
константа::= цифра$\{$цифра$\}^{*}$
змінна::= літера $\{$ літера $|$цифра$\}^{*}$
цифра::= 0|1|2|3|4|5|6|7|8|9
літера::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

Варіант 5

Розробіть синтаксичний аналізатор для поняття **присвоювання**:

присвоювання::= {ліва_частина=права_частина}
ліва_частина::= змінна
права_частина::= змінна |{ змінна знак_операції змінна }
змінна::= літера { літера | цифра}*
знак_операції::= $+|-|*|/$
цифра::= 0|1|2|3|4|5|6|7|8|9
літера::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

Варіант 6

Розробіть синтаксичний аналізатор для поняття **список_параметрів**:

список_параметрів::= параметр {,параметр}*
параметр::= {ім'я = цифра$^{+}$}| { ім'я = літера $^{+}$}
ім'я::= літера$^{+}$
цифра::= 0|1|2|3|4|5|6|7|8|9
літера::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

Варіант 7

Розробіть синтаксичний аналізатор для поняття **скалярний_тип**:

скалярний_тип::= {ім'я_константи {, ім'я_константи}*}
| {константа . . константа}
Ім'я_константи::= літера {цифра | літера }*
константа::= цифра {цифра}*
цифра::= 0|1|2|3|4|5|6|7|8|9
літера::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

Варіант 8

Розробіть синтаксичний аналізатор для поняття **дійсне_число**:

дійсне_число::= {ціле_число. ціле_без_знака}
| { ціле_число. ціле_без_знака E ціле _число}
| { ціле_число E ціле_число}
ціле_число::= {+ | - }? ціле_без_знака
ціле_без_знака::= цифра {цифра}*
цифра::= 0|1|2|3|4|5|6|7|8|9

Варіант 9

Розробіть синтаксичний аналізатор для поняття **опис_міток**:

опис_міток::= LABEL мітка {, мітка}*;
мітка::= ідентифікатор | послідовність_цифр
послідовність_цифр::= 0 . . 9999
ідентифікатор::= літера { літера | цифра}*
цифра::= 0|1|2|3|4|5|6|7|8|9
літера::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

Варіант 10

Розробіть синтаксичний аналізатор для поняття **арифметична дія**:

арифметична дія::= ціле {знак_операції ціле }$^{+}$
ціле::= цифра{цифра}*
знак_операції::= * | / | + | -
цифра::= 0|1|2|3|4|5|6|7|8|9

Варіант 11

Розробіть синтаксичний аналізатор для поняття **оператор_ циклу**:

оператор_циклу::= while (умова) оператор_присвоювання
умова::= змінна {< | >} цифра$^{+}$
оператор_ присвоювання::= {змінна=цифра$^{+}$;}
змінна::= літера$^{+}$
цифра::= 1|2|3|4|5
літера::= a|b|c|d

Варіант 12

Розробіть синтаксичний аналізатор для поняття **функція**:

функція::= тип_повертання ім'я список_параметрів тіло_функції
тип_ повертання::= void
ім'я::= літера { літера | цифра}*
список_параметрів::= {()} | {(параметр{, параметр}*)}
тіло_функції::= ліва_дужка пустий_оператор права_дужка
пустий_оператор::= ;
ліва_ дужка::= {
права_ дужка::= }
літера::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
цифра::= 0|1|2|3|4|5|6|7|8|9

Варіант 13

Розробіть синтаксичний аналізатор для поняття **текст**:

текст::= слово{,слово}*
слово::= корень{суфікс}*
корень::= сад | мед
суфікс::= ик | ок

Варіант 14

Розробіть синтаксичний аналізатор для поняття **список_списків**:

список_списків::= список {; список}*
список::= елемент{, елемент }*
елемент::= літера
літера::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

Варіант 15

Розробіть синтаксичний аналізатор для поняття **послідовність_символів**:

послідовність_символів::= елемент {роздільник елемент}*
елемент::= {символ символ*}
символ::= літера | цифра
літера::= а|б|в|…|я
цифра::= 0|1|2|3|4|5|6|7|8|9
роздільник::= {, | ;}

Завдання 2

Розробіть програму, яка знаходить усі номери телефонів припустимих форматів у довільній послідовності символів.

Програма має містити блок синтаксичного аналізу вхідної послідовності на базі регулярних виразів.

Послідовність символів для аналізу необхідно ввести з консолі. Виведіть на консоль початкову послідовність символів, усі знайдені номери телефонів та їхню загальну кількість. Якщо жоден номер телефону не знайдений, необхідно вивести на консоль відповідне повідомлення.

Програма має виконуватися циклічно до введення користувачем рядка-ознаки завершення програми. Варіанти цього завдання наведено далі:

Номер варіанта	Припустимі формати номерів, де x – цифра від 1 до 9	
1	(+81xx)xx-xxx-xxx	(xxx)xx-xxx-xxx
2	(+380xx)xxx-xx-xx	xxx-xx-xx
3	(0572)x-xx-xxx	x-xx-xxx
4	(+1xxx) xx-xxx-xx	(xxx) xx-xxx-xx
5	(+86) xxxx-xxxx-xxx	xxxx-xxxx-xxx
6	(xxx)xx-xx-xx	xx-xx-xx

7	(+55xx) xx-xxx-xxx	(xx) xx-xxx-xxx
8	(+244) xxx-xxx-xxx	xxx-xxx-xxx
9	(044)xxxxx-xx	xxxxx-xx
10	(+61) xx-xxx-xxxx	xx-xxx-xxxx
11	(+226) xx-xx-xx-xx	xx-xx-xx-xx
12	(061)xxxxxxx	xx-xxx-xx
13	(+850) xxxx-xxx-xxx	xxxx-xxx-xxxx
14	(+27) xx-xxx-xxxx	xxx-xxx-xxxx
15	(031)x-xxxx-xx	x-xxxx-xx

Контрольні запитання до лабораторної роботи 9

1. Коли доцільно використовувати регулярні вирази?
2. Який простір імен або пакет необхідно підключити для можливості використання регулярних виразів у Microsoft .NET або Java SE відповідно?
3. Який клас Microsoft .NET або Java SE представляє компільований регулярний вираз?
4. Які основні операції підтримуються класом регулярних виразів?
5. Для чого використовуються символи ^ та $ під час визначення регулярного виразу?

Лабораторна робота 10
Використання введення-виведення даних

Мета лабораторної роботи – вдосконалення практичних навичок з використання введення-виведення даних і роботи з інтегрованим середовищем MS Visual Studio і Eclipse.

Перед виконанням лабораторної роботи студент має:

знати основи використання введення-виведення даних;

вміти використовувати бібліотеки введення-виведення даних під час розроблення програм.

Завдання

Додайте до програми мовою Java або C# для оброблення відомості (див. завдання лабораторних робіт 7 і 8) можливість введення-виведення даних певній колекції.

Нова функціональність має забезпечувати за запитом користувача:

1. Виведення даних з колекції до текстового CSV-файла (рівні 1 і 2) або бінарну серіалізацію колекції до файла (рівні 3 і 4) з використанням окремого методу.

2. Введення даних у колекцію (див. табл. 10.1) з текстового CSV-файла (рівні 1 і 2) або бінарну десеріалізацію колекції з файла (рівні 3 і 4) з використанням окремого методу.

Структура CSV-файла. Кожен запис має розташовуватися в окремому рядку. Поля кожного запису мають бути розділені роздільником відповідно до варіанта завдання, наприклад комою:

Іван,Петров,8,7,11

Галина,Бланка,10,23,4

Варіанти класів колекцій та символів-роздільників наведено у табл. 10.1.

Таблиця 10.1

Варіанти завдань

№ з/п	Символ-роздільник полів записів CSV-файла	Клас колекції Microsoft .NET	Клас колекції Java SE
1	;	Hashtable	HashSet<T>
2	:	ArrayList	HashMap<K, V>
3	|	SortedList	ArrayList<T>
4	/	List<T>	TreeSet<T>
5	\	LinkedList<T>	TreeMap<K, V>
6	$	HashSet<T>	LinkedList<T>
7	&	Dictionary<K, V>	LinkedHashSet<T>
8	!	SortedDictionary <K, V >	ArrayList
9	#	SortedList<K, V>	HashSet<T>
10	%	Hashtable	HashMap<K, V>
11	+	SortedSet<T>	LinkedHashMap <K, V>
12	пробіл	SortedList	TreeSet<T>
13	-	HashSet<T>	TreeMap<K, V>
14	*	Dictionary<K, V>	LinkedList<T>
15	@	SortedList<K, V>	LinkedHashSet<T>

Інформацію щодо предметної області слід взяти за варіантами з лабораторної роботи 1.

Контрольні запитання до лабораторної роботи 10

1. До яких елементів програми можливе застосування атрибутів?
2. Для чого використовують атрибути умовної компіляції?
3. Як створити клас, об'єкти якого можна серіалізувати?
4. Які ви знаєте формати серіалізації?
5. Укажіть базові класи байтових і символьних потоків введення – виведення Java SE та Microsoft .NET.
6. Для чого призначено основні класи символьних потоків введення – виведення Java SE та Microsoft .NET?
7. Для чого призначено основні класи байтових потоків введення – виведення Java SE та Microsoft .NET?

Лабораторна робота 11
Основи JavaFX

Мета лабораторної роботи:

1. Придбання практичного досвіду з розроблення простих застосунків з графічним інтерфейсом на платформі Java SE.
2. Удосконалення навичок роботи в інтегрованому середовищі JetBrains Idea або Eclipse.

Перед виконанням лабораторної роботи студент має:

знати:

основи використання технології JavaFX для розроблення застосунків з графічним інтерфейсом на платформі Java SE;

основи оброблення подій на платформі Java SE;

вміти:

розробляти прості програми з графічним інтерфейсом користувача на базі JavaFX;

використовувати електронну документацію розробника для пошуку інформації, необхідної під час розроблення застосунків із графічним інтерфейсом.

Налаштування головних параметрів форми в Java-коді наведено в табл. 11.1

Таблиця 11.1

Налаштування головних параметрів форми в Java-коді

Назва параметра	Необхідні дії
Заголовок вікна	Викликати метод **setTitle** для сцени (об'єкта класу Stage**)**
Піктограма системного меню	**primaryStage**.getIcons().add(new Image("images/test.png")), де primaryStage – посилання на об'єкт сцени; images – папка з зображенням формату png, gif або jpg у src проєкту Eclipse
Курсор миші	Викликати метод **setCursor** для кореневого елемента епізоду (об'єкта класу AnchorPane, BorderPane, …)
Колір клієнтської області вікна	Зробити кореневий елемент епізоду повністю прозорим шляхом виклику для нього методу **setOpacity(0)**, а потім викликати метод **setFill** для епізоду (об'єкта класу Scene)
Координати лівого верхнього кута вікна (x, y)	Викликати **setX** і **setY** для сцени
Ширина вікна	Задати значення параметра **width** у конструкторі епізоду
Висота вікна	Задати значення параметра **height** у конструкторі епізоду
Початковий вигляд вікна	За стандартом ширина і висота сцени відповідає заданим розмірам. Можна викликати один з методів сцени з параметром **true**: **setMaximized** – сцена розгорнута; **setIconfied** – сцена згорнута

Порядок виконання лабораторної роботи

Завдання 1

Розробіть застосунок JavaFX з використанням лише мови Java.

Загальні вимоги до застосунку

1. Застосунок має одне вікно (об'єкт класу **Stage**) з кореневим елементом **BorderPane** (об'єкт класу **BorderPane**).

2. Заголовок головного вікна – прізвище, ім'я, по батькові студента.

3. Колір клієнтської області вікна – встановлений "за замовченням" (рівень 1) або один з стандартних кольорів (поле класу javafx.scene.paint.Color) відповідно до варіанта завдання (рівні 2 – 4).

4. Піктограма системного меню – встановлена "за замовченням" (рівні 1 і 2) або власна піктограма (файл *.gif, *.jpg, *.png) – рівні 3 і 4.

5. Курсор миші – встановлений "за замовченням" (рівень 1) або один із стандартних курсорів (поле класу javafx.scene.Cursor) відповідно

до варіанта завдання – рівні 2 і 3 або власний курсор (завантажується з файла) – рівень 4.

6. Застосунок має обробляти подію клацання кнопки миші **(MouseEvent.MOUSE_CLICKED)** або натискання клавіш на клавіатурі комп'ютера відповідно до варіанта завдання **(KeyEvent.KEY_PRESSED** або **KeyEvent.KEY_TYPED)**. У обробнику події клацання кнопки миші необхідно визначити, чи відповідає натиснута кнопка (ліва або права), а також кількість клацань (одне або два) зазначеним у варіанті завдання, і, якщо це так, показати:

- стандартне діалогове вікно з кнопкою OK і повідомленням щодо натиснутої кнопки миші (ліва або права) – рівень 1;
- стандартне діалогове вікно з повідомленням, в якому відображаються поточні координати курсору миші в момент клацання, а також кнопки або піктограма відповідно до варіанта завдання – рівні 2 – 4.

У обробнику події натискання клавіш на клавіатурі необхідно показати:

- стандартне діалогове вікно з кнопкою OK і повідомленням про натискання клавіші – рівень1;
- стандартне діалогове вікно з повідомленням, в якому відображається символьний код натиснутої клавіші або символ, відповідний натиснутій алфавітно-цифровій клавіші, а також кнопки або піктограми відповідно до варіанта завдання – рівні 2 – 4.

Варіанти завдань наведені у табл. 11.2 – 11.16.

Таблиця 11.2

Варіант 1

Атрибут	Значення
Курсор миші	TEXT
Колір клієнтської області вікна	Червоний
Координати лівого верхнього кута вікна (x, y)	10, 10
Ширина вікна	100
Висота вікна	100
Початковий вигляд вікна	Розгорнуто на весь екран
Подія	Одинарне клацання лівою кнопкою миші
Кнопки вікна повідомлення	Yes, No, Cancel

Таблиця 11.3

Варіант 2

Атрибут	Значення
Курсор миші	CROSSHAIR
Колір клієнтської області вікна	Жовтогарячий
Координати лівого верхнього кута вікна (x, y)	50, 100
Ширина вікна	100
Висота вікна	150
Початковий вигляд вікна	Згорнуто
Подія	Подвійне клацання лівою кнопкою миші
Піктограма вікна повідомлення	Warning

Таблиця 11.4

Варіант 3

Атрибут	Значення
Курсор миші	CLOSED_HAND
Колір клієнтської області вікна	Жовтий
Координати лівого верхнього кута вікна (x, y)	100, 10
Ширина вікна	200
Висота вікна	100
Початковий вигляд вікна	Відповідає заданим розмірам
Подія	Одинарне клацання правою кнопкою миші
Кнопки вікна повідомлення	OK

Таблиця 11.5

Варіант 4

Атрибут	Значення
Курсор миші	WAIT
Колір клієнтської області вікна	Зелений
Координати лівого верхнього кута вікна (x, y)	300, 200
Ширина вікна	400
Висота вікна	400
Початковий вигляд вікна	Розгорнуто на весь екран
Подія	Подвійне клацання правою кнопкою миші
Кнопки вікна повідомлення	OK, Close

Таблиця 11.6

Варіант 5

Атрибут	Значення
Курсор миші	OPEN_HAND
Колір клієнтської області вікна	Синій
Координати лівого верхнього кута вікна (x, y)	0, 0
Ширина вікна	799
Висота вікна	599
Початковий вигляд вікна	Згорнуто
Подія	Одинарне клацання лівою кнопкою миші
Кнопки вікна повідомлення	Yes, No

Таблиця 11.7

Варіант 6

Атрибут	Значення
Курсор миші	MOVE
Колір клієнтської області вікна	Фіолетовий
Координати лівого верхнього кута вікна (x, y)	555, 666
Ширина вікна	100
Висота вікна	300
Початковий вигляд вікна	Відповідає заданим розмірам
Подія	Подвійне клацання лівою кнопкою миші
Кнопки вікна повідомлення	Close

Таблиця 11.8

Варіант 7

Атрибут	Значення
Курсор миші	V_RESIZE
Колір клієнтської області вікна	Коричневий
Координати лівого верхнього кута вікна (x, y)	400, 300
Ширина вікна	200
Висота вікна	300
Початковий вигляд вікна	Розгорнуто на весь екран
Подія	Одинарне клацання правою кнопкою миші
Кнопки вікна повідомлення	Yes, Cancel

Таблиця 11.9

Варіант 8

Атрибут	Значення
Курсор миші	H_RESIZE
Колір клієнтської області вікна	Бузковий
Координати лівого верхнього кута вікна (x, y)	20, 10
Ширина вікна	800
Висота вікна	500
Початковий вигляд вікна	Згорнуто
Подія	Подвійне клацання правою кнопкою миші
Піктограма вікна повідомлення	Знак оклику

Таблиця 11.10

Варіант 9

Атрибут	Значення
Курсор миші	N_RESIZE
Колір клієнтської області вікна	Ліловий
Координати лівого верхнього кута вікна (x, y)	333, 444
Ширина вікна	500
Висота вікна	500
Початковий вигляд вікна	Відповідає заданим розмірам
Подія	Натискання клавіші на клавіатурі – **KEY_PRESSED**
Піктограма вікна повідомлення	Error

Таблиця 11.11

Варіант 10

Атрибут	Значення
Курсор миші	NE_RESIZE
Колір клієнтської області вікна	Бірюзовий
Координати лівого верхнього кута вікна (x, y)	333, 77
Ширина вікна	333
Висота вікна	144
Початковий вигляд вікна	Розгорнуто на весь екран
Подія	Натискання алфавітно-цифрової клавіші на клавіатурі – **KEY_TYPED**
Кнопки вікна повідомлення	Yes, No

Таблиця 11.12

Варіант 11

Атрибут	Значення
Курсор миші	E_RESIZE
Колір клієнтської області вікна	Сірий
Координати лівого верхнього кута вікна (x, y)	80, 180
Ширина вікна	156
Висота вікна	312
Початковий вигляд вікна	Згорнуто
Подія	Натискання клавіші на клавіатурі – **KEY_PRESSED**
Кнопки вікна повідомлення	No

Таблиця 11.13

Варіант 12

Атрибут	Значення
Курсор миші	SE_RESIZE
Колір клієнтської області вікна	Рожевий
Координати лівого верхнього кута вікна (x, y)	69, 96
Ширина вікна	400
Висота вікна	700
Початковий вигляд вікна	Відповідає заданим розмірам
Подія	Натискання алфавітно-цифрової клавіші на клавіатурі, натискання клавіші на клавіатурі – **KEY_TYPED**
Кнопки вікна повідомлення	Yes

Таблиця 11.14

Варіант 13

Атрибут	Значення
Курсор миші	S_RESIZE
Колір клієнтської області вікна	Червоний
Координати лівого верхнього кута вікна (x, y)	55, 0
Ширина вікна	200
Висота вікна	200
Початковий вигляд вікна	Знаходиться в центрі екрану, відповідає заданим розмірам
Подія	Натискання клавіші на клавіатурі – **KEY_PRESSED**
Кнопки вікна повідомлення	OK

Таблиця 11.15

Варіант 14

Атрибут	Значення
Двостороння стрілка	SW_RESIZE
Колір клієнтської області вікна	Білий
Координати лівого верхнього кута вікна (x, y)	0, 0
Ширина вікна	300 (фіксований розмір)
Висота вікна	500 (фіксований розмір)
Початковий вигляд вікна	Відповідає заданим розмірам
Подія	Натискання алфавітно-цифрової клавіші на клавіатурі – **KEY_TYPED**
Кнопки вікна повідомлення	OK, Cancel

Таблиця 11.16

Варіант 15

Атрибут	Значення
Курсор миші	HAND
Колір клієнтської області вікна	Чорний
Координати лівого верхнього кута вікна (x, y)	10, 30
Ширина вікна	100
Висота вікна	100
Початковий вигляд вікна	Згорнуто
Подія	Натискання клавіші на клавіатурі – **KEY_PRESSED**
Кнопки вікна повідомлення	Close

Завдання 2

Розробіть JavaFX-застосунок з графічним інтерфейсом користувача на базі FXML, який надає функціональність "Калькулятора". Головне вікно програми (об'єкт класу **Stage**) повинно мати фіксований розмір.

Графічний інтерфейс "Калькулятора" для рівнів 1, 2 має виглядати як на рис. 11.1.

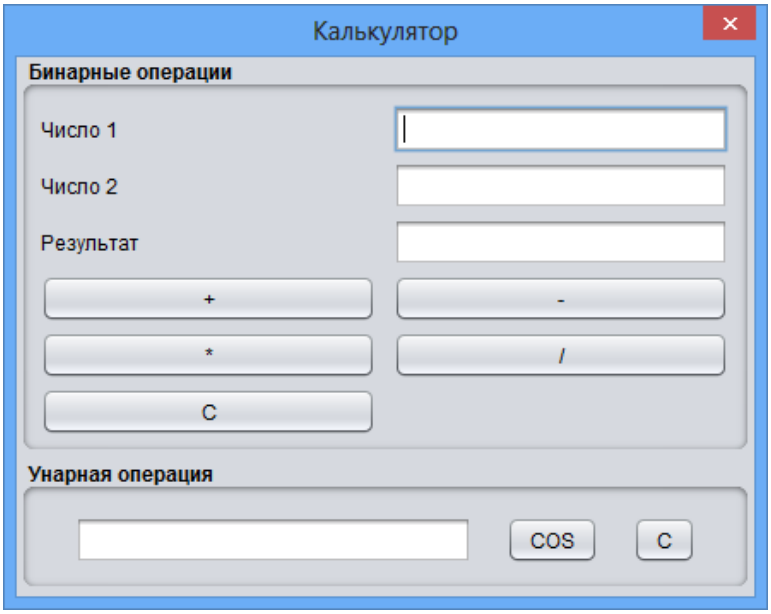

Рис. 11.1. **Графічний інтерфейс для рівнів 1 і 2**

Для цього рекомендується компоновка елементів графічного інтерфейсу "Калькулятора", наведена на рис. 11.2.

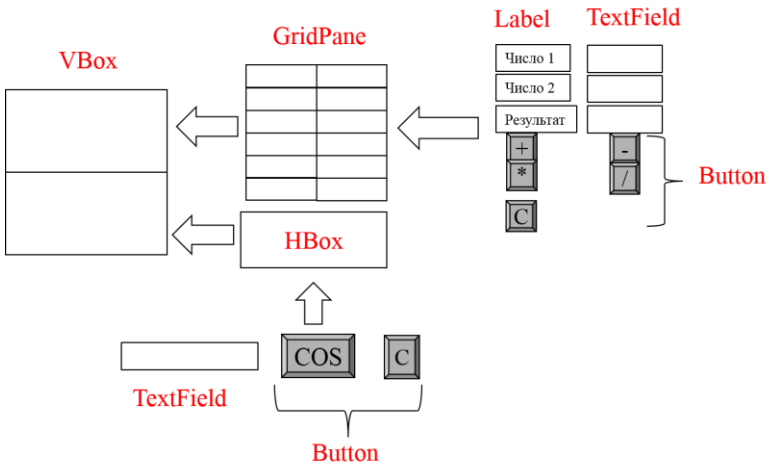

Рис. 11.2. **Компоновка елементів графічного інтерфейсу для рівнів 1 і 2**

Тут TextField, Label, Button – елементи управління; VBox, HBox, GridPane – панелі компонування.

Графічний інтерфейс "Калькулятора" для рівнів 3 і 4 має виглядати, як наведено на рис. 11.3.

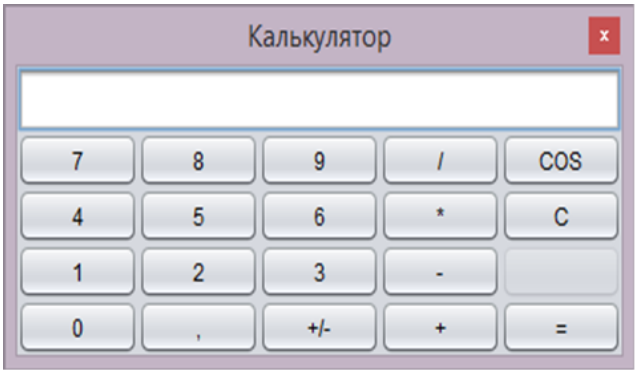

Рис. 11.3. **Графічний інтерфейс для рівнів 3 і 4**

Для цього рекомендується компоновка елементів графічного інтерфейсу "Калькулятора", наведена на рис. 11.4.

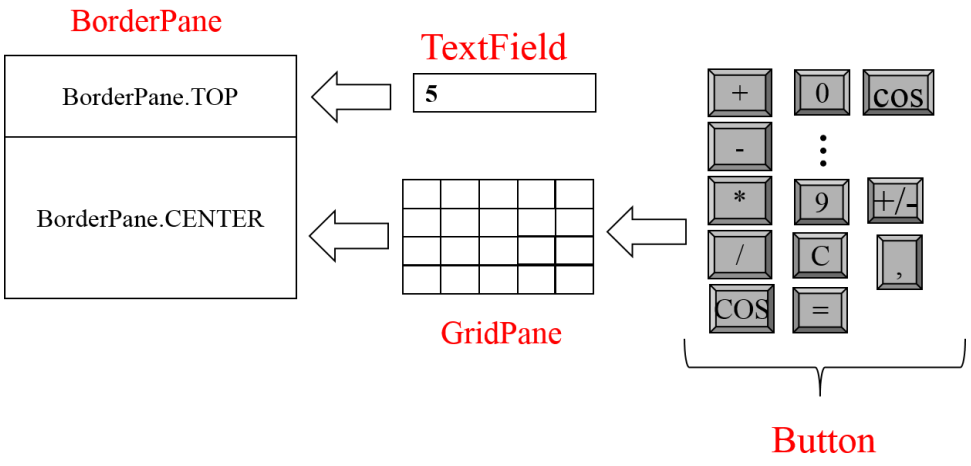

Рис. 11.4. **Компоновка елементів графічного інтерфейсу для рівнів 3 і 4**

Тут TextField, Button – елементи управління; BorderPane, GridPane – панелі компонування.

"Калькулятор" має виконувати будь-яку з чотирьох арифметичних дій: "+", "–", "×", "/" над двома дійсними числами. Він також має виконувати додаткову операцію, яка визначається варіантом завдання (рівні 2 – 4).

"Калькулятор" не має дозволяти користувачеві вводити значення вихідних даних, які не відповідають обмеженням, зазначеним у варіантах завдань, наведених в табл. 1.1 (рівень 4).

Контрольні запитання до лабораторної роботи 11

1. Як використовуються "колекції" візуальних елементів управління формою?
2. Який порядок застосування обробників подій рівня форми?
3. Як розміщуються візуальні елементи управління на формі?
4. Які особливості використання SceneBuilder для розроблення графічних інтерфейсів користувача?
5. Які є основні елементи синтаксису мови FXML?

Лабораторна робота 12
Використання елементів управління та графічних можливостей JavaFX

Мета лабораторної роботи – удосконалення навичок з розроблення графічних інтерфейсів користувача на платформі Java SE та використання інтегрованого середовища програмування JetBrains Idea або Eclipse.

Перед виконанням лабораторної роботи студент має:

знати:

основи використання елементів управління та стилів у JavaFX;

базові графічні можливості JavaFX;

вміти:

самостійно розробляти програми з графічним інтерфейсом з використанням JavaFX;

використовувати основні класи елементів управління та графічні можливості JavaFX.

Порядок виконання лабораторної роботи

Завдання 1

Розробіть програму з графічним інтерфейсом користувача, в головному вікні якої знаходяться як мінімум елемент управління Canvas

(клас javafx.scene.canvas.Canvas) та панель компонування з пакету javafx.scene.layout. Наприклад, панель GridPane.

Canvas – це "полотно" для малювання на ньому певної геометричної фігури.

Будь-яка із зазначених панелей компонування призначена для розташування в ній елементів управління для зміни кольору контуру і внутрішньої області цієї геометричної фігури.

Для створення геометричної фігури необхідно використовувати певний стандартний графічний примітив, а в разі його відсутності – графічний примітив Polygon ("Багатокутник"). Геометрична фігура має створюватися по одинарному (або подвійному) клацанню лівою (або правою) кнопкою миші згідно з варіантом завдання. Всі намальовані фігури мають відображатися, а кольори їхніх контурів і внутрішніх областей – визначатися відповідними елементами управління.

Варіанти завдань наведено у табл. 12.1 – 12.15.

Таблиця 12.1

Варіант 1

Геометрична фігура	Коло (координати її центру мають збігатися з координатами курсору миші в момент клацання)
Керування створенням фігури	Одинарне клацання лівою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	RadioButton
Елемент управління для зміни кольору внутрішньої області фігури (рівні 3 і 4)	ComboBox

Таблиця 12.2

Варіант 2

Геометрична фігура	Прямокутник (координати його лівого верхнього кута мають збігатися з координатами курсору миші в момент клацання)
Керування створенням фігури	Одинарне клацання правою кнопкою миші
Елемент управління для зміни кольору внутрішньої області фігури (рівні 3 і 4)	Slider (3 шт.)

Таблиця 12.3

Варіант 3

Геометрична фігура	Еліпс (координати його центру мають збігатися з координатами курсору миші в момент клацання)
Керування створенням фігури	Подвійне клацання лівою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	Slider (3 шт.)

Таблиця 12.4

Варіант 4

Геометрична фігура	"Пісочний годинник" (координати його центру мають збігатися з координатами курсору миші в момент клацання)
Керування створенням фігури	Подвійне клацання правою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	ColorPicker
Елемент управління для зміни кольору внутрішньої області фігури (рівні 3 і 4)	ColorPicker

Таблиця 12.5

Варіант 5

Геометрична фігура	Прямокутник з округленими кутами (координати мають збігатися з координатами курсору миші)
Керування створенням фігури	Одинарне клацання лівою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	RadioButton
Елемент управління для зміни кольору внутрішньої області фігури (рівні 3 і 4)	RadioButton

Таблиця 12.6

Варіант 6

Геометрична фігура	Двонаправлена горизонтальна стрілка (координати мають збігатися з координатами курсору миші в момент клацання)
Керування створенням фігури	Одинарне клацання правою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	RadioButton
Елемент управління для зміни кольору внутрішньої області фігури (рівні 3 і 4)	ComboBox

Таблиця 12.7

Варіант 7

Геометрична фігура	Велика буква Z (координати її лівого нижнього кута мають збігатися з координатами курсору миші в момент клацання)
Керування створенням фігури	Подвійне клацання лівою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	Slider (3 шт.)

Таблиця 12.8

Варіант 8

Геометрична фігура	Рівносторонній трикутник (координати вершини його верхнього кута мають збігатися з координатами курсору миші в момент клацання)
Керування створенням фігури	Подвійне клацання правою кнопкою миші
Елемент управління для зміни кольору внутрішньої області фігури (рівні 3 і 4)	Slider (3 шт.)

Таблиця 12.9

Варіант 9

Геометрична фігура	Восьмикутна зірка (координати вершини її верхнього кута мають збігатися з координатами курсору миші в момент клацання)
Керування створенням фігури	Одинарне клацання лівою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	ComboBox
Елемент управління для зміни кольору внутрішньої області фігури (рівні 3 і 4)	ComboBox

Таблиця 12.10

Варіант 10

Геометрична фігура	Правильний восьмикутник (координати вершини мають збігатися з координатами курсору миші)
Керування створенням фігури	Одинарне клацання правою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	RadioButton
Елемент управління для зміни кольору внутрішньої області фігури (рівні 3 і 4)	RadioButton

Таблиця 12.11

Варіант 11

Геометрична фігура	Ромб (координати вершини його верхнього кута мають збігатися з координатами курсору миші)
Керування створенням фігури	Подвійне клацання лівою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	ComboBox

Таблиця 12.12

Варіант 12

Геометрична фігура	Правильний шестикутник (координати вершини мають збігатися з координатами курсору миші)
Керування створенням фігури	Подвійне клацання правою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	Slider (3 шт.)

Таблиця 12.13

Варіант 13

Геометрична фігура	"Прямий кут" (координати вершини його лівого верхнього кута мають збігатися з координатами курсору миші в момент клацання)
Керування створенням фігури	Одинарне клацання лівою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	RadioButton

Таблиця 12.14

Варіант 14

Геометрична фігура	Паралелограм (координати однієї з вершин паралелограма мають збігатися з координатами курсору миші в момент клацання)
Керування створенням фігури	Одинарне клацання правою кнопкою миші
Елемент управління для зміни кольору контуру фігури (рівні 3 і 4)	ColorPicker

Таблиця 12.15

Варіант 15

Геометрична фігура	"Хрестоподібний знак множення" (координати його нижнього лівого кута мають збігатися з координатами курсору миші в момент клацання)
Керування створенням фігури	Подвійне клацання лівою кнопкою миші
Елемент управління для зміни кольору внутрішньої області фігури (рівні 3 і 4)	Slider (3 шт.)

Завдання 2

Розробіть застосунок JavaFX з графічним інтерфейсом користувача, який будує і відображає графік певної функції y(x) з використанням компонента LineChart.

Головне вікно програми має містити сам графік функції, його назву, координатну сітку, координатні осі OX і OY, їхнє найменування та градацію, умовні позначення. Зображення графіка має займати всю клієнтську область вікна та масштабуватися при зміні розмірів вікна.

Для рівня 3: забезпечте можливість установки визначеного кольору лінії графіка, кольору фону області діаграми, товщини лінії графіка за допомогою стилів CSS під час запуску програми на виконання.

Для рівня 4: забезпечте для користувача застосунку можливість зміни кольору лінії графіка, кольору фону області діаграми, товщини лінії графіка з використанням стилів CSS і необхідних елементів управління.

Варіанти завдань наведені у табл. 12.16.

Таблиця 12.16

Варіанти завдань

№ з/п	Функція
1	2
1	$2\sin(x)\cos(x)$, де $0° \le x \le 360°$
2	$\sin(x)\cos(2x)$, де $0° \le x \le 360°$
3	$3\cos(x)$, де $0° \le x \le 360°$
4	$2e^{-x}$, де $0 \le x \le 5$
5	$\sin(x) / x$, де $-2\pi \le x \le 2\pi$; $\sin(0) / 0=1$

Закінчення табл. 12.16

1	2
6	cos(x) / x, де $0 < x \le 2\pi$
7	lg(x), де $x > 0$
8	$\sin(x)e^{-x}$, де $-2\pi \le x \le 2\pi$;
9	tg(x), де $-90° < x < 90°$
10	$\cos(x)e^{-x}$, де $-2\pi \le x \le 2\pi$;
11	ln(x) , де $x > 0$
12	x^2, де $-10 \le x \le 10$
13	x^3, де $-5 \le x \le 5$
14	ctg(x), де $0° < x < 180°$
15	1 / x, де $0 < x \le 50$

Контрольні запитання до лабораторної роботи 12

1. Які графічні можливості має JavaFX?
2. Який порядок побудови графічних зображень?
3. Як здійснюється робота зі стандартними графічними примітивами?
4. Як здійснюється робота з діаграмами та графіками у JavaFX?

Лабораторна робота 13
Використання головних елементів PyQt

Мета лабораторної роботи:

1. Придбання практичного досвіду з розроблення простих застосунків з графічним інтерфейсом користувача з використанням мови Python.

2. Удосконалення навичок роботи в інтегрованому середовищі PyCharm.

Перед виконанням лабораторної роботи студент має:

знати:

основи використання Python для розроблення застосунків з графічним інтерфейсом користувача;

основи оброблення подій в PyQt;

вміти розробляти прості програми з графічним інтерфейсом користувача на базі PyQt.

Порядок виконання лабораторної роботи

Завдання 1

Розробіть PyQt-застосунок з графічним інтерфейсом користувача на базі XML, який надає функціональність "Калькулятора". Головне вікно програми повинно мати фіксований розмір.

Графічний інтерфейс "Калькулятора" для рівнів 1 і 2 має виглядати приблизно, як на рис. 13.1.

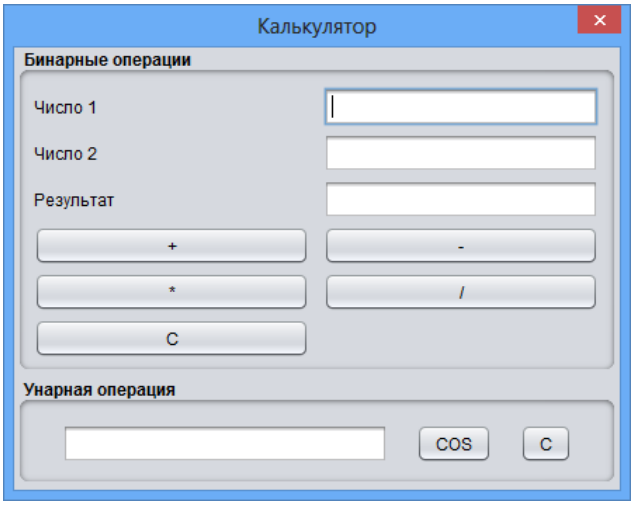

Рис. 13.1. **Графічний інтерфейс для рівнів 1 і 2**

Для цього рекомендується компоновка елементів графічного інтерфейсу "Калькулятора" , наведена на рис. 13.2.

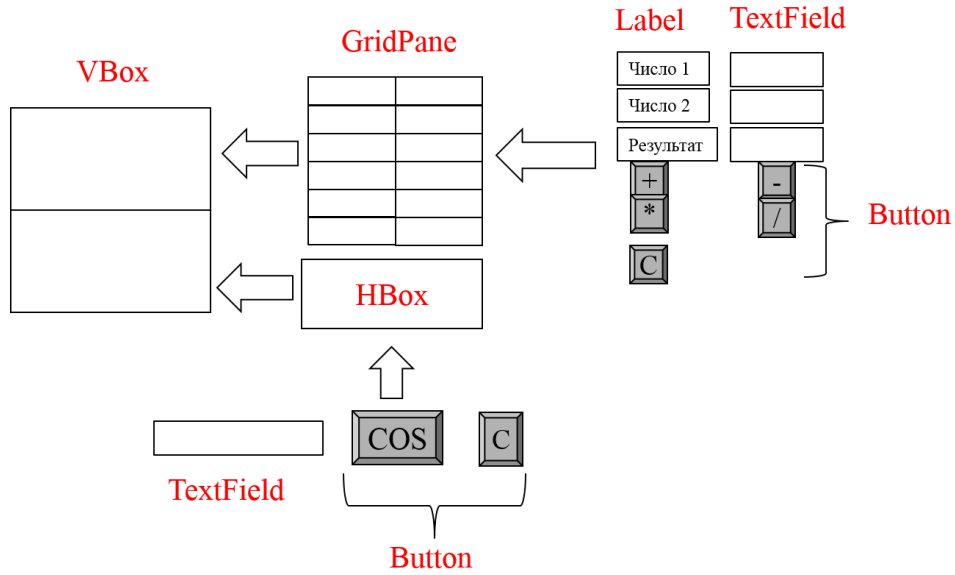

Рис. 13.2. **Компоновка елементів графічного інтерфейсу для рівнів 1 і 2**

Тут TextField, Label, Button – PyQt-віджети QLineEdit, QLabel і QPushButton відповідно; VBox, HBox, GridPane – панелі компонування PyQt: QVBoxLayout, QHBoxLayout та QGridLayout відповідно.

Графічний інтерфейс "Калькулятора" для рівнів 3 і 4 має виглядати приблизно, як наведено на рис. 13.3.

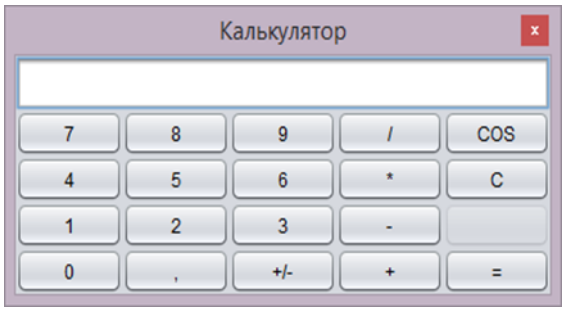

Рис. 13.3. **Графічний інтерфейс для рівнів 3 і 4**

Для цього рекомендується компоновка елементів графічного інтерфейсу "Калькулятора", наведена на рис. 13.4.

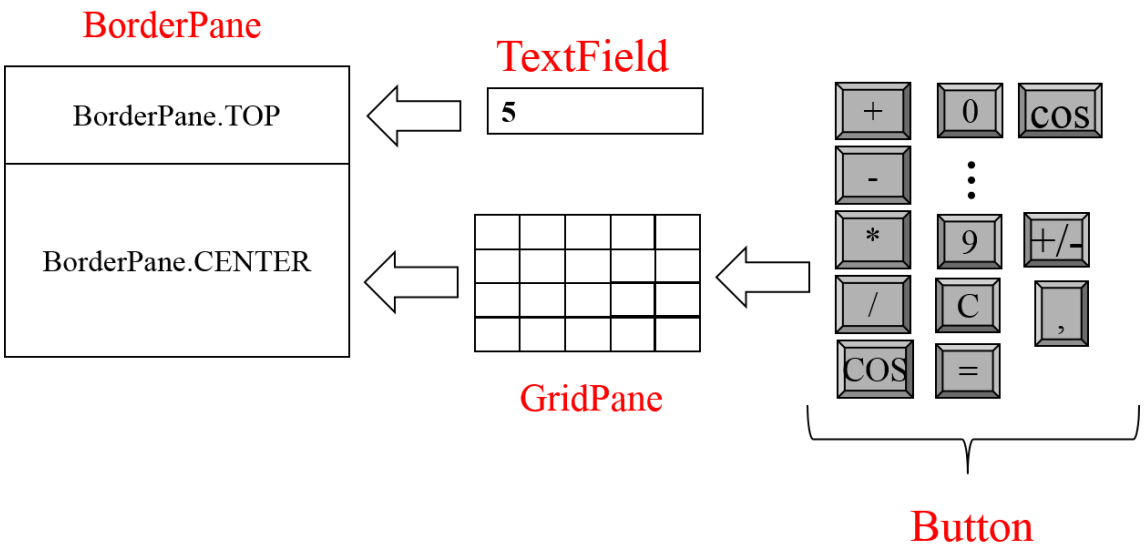

Рис. 13.4. **Компоновка елементів графічного інтерфейсу для рівнів 3 і 4**

Тут TextField, Button – PyQt-віджеті QLineEdit та QPushButton відповідно; BorderPane, GridPane – панелі компонування PyQt: QVBoxLayout та QGridLayout відповідно.

"Калькулятор" має виконувати будь-яку з чотирьох арифметичних дій: "+", "–", "×", "/" над двома дійсними числами. Він також має виконувати

додаткову операцію, яка визначається варіантом завдання, наведеному в табл. 1.1 (рівні 2 – 4).

"Калькулятор" не має дозволяти користувачеві вводити значення вихідних даних, які не відповідають обмеженням, зазначеним у варіантах завдань (рівень 4).

Контрольні запитання до лабораторної роботи 13

1. Як використовується Python для розроблення застосунків з графічним інтерфейсом користувача?
2. Які модулі для створення графічних зображень є в Python?
3. Який порядок оброблення подій у PyQt?

Лабораторна робота 14
Основи використання WPF

Мета лабораторної роботи:

1. Придбання практичного досвіду з розроблення простих застосунків з графічним інтерфейсом з використанням технології WPF.
2. Удосконалення навичок роботи в інтегрованому середовищі Microsoft Visual Studio.

Перед виконанням лабораторної роботи студент має:

знати:

основні елементи WPF та мови XAML;

основи оброблення подій у WPF;

вміти розробляти прості програми з графічним інтерфейсом користувача на базі WPF.

Використання середовища Microsoft Visual Studio під час розроблення застосунків WPF

Створення проєкту в інтегрованому середовищі Microsoft Visual Studio:

1. Оберіть меню File → New Project.
2. У вікні New Project оберіть тип проєкту WPF Application.
3. Введіть назву проєкту в поле Name і вкажіть шлях до місця розташування проєкту в поле Location.

4. Натисніть кнопку ОК.

Після виконання цих дій має з'явитися вікно, наведене на рис. 14.1, в якому має бути порожня форма та область з відповідним XAML-кодом.

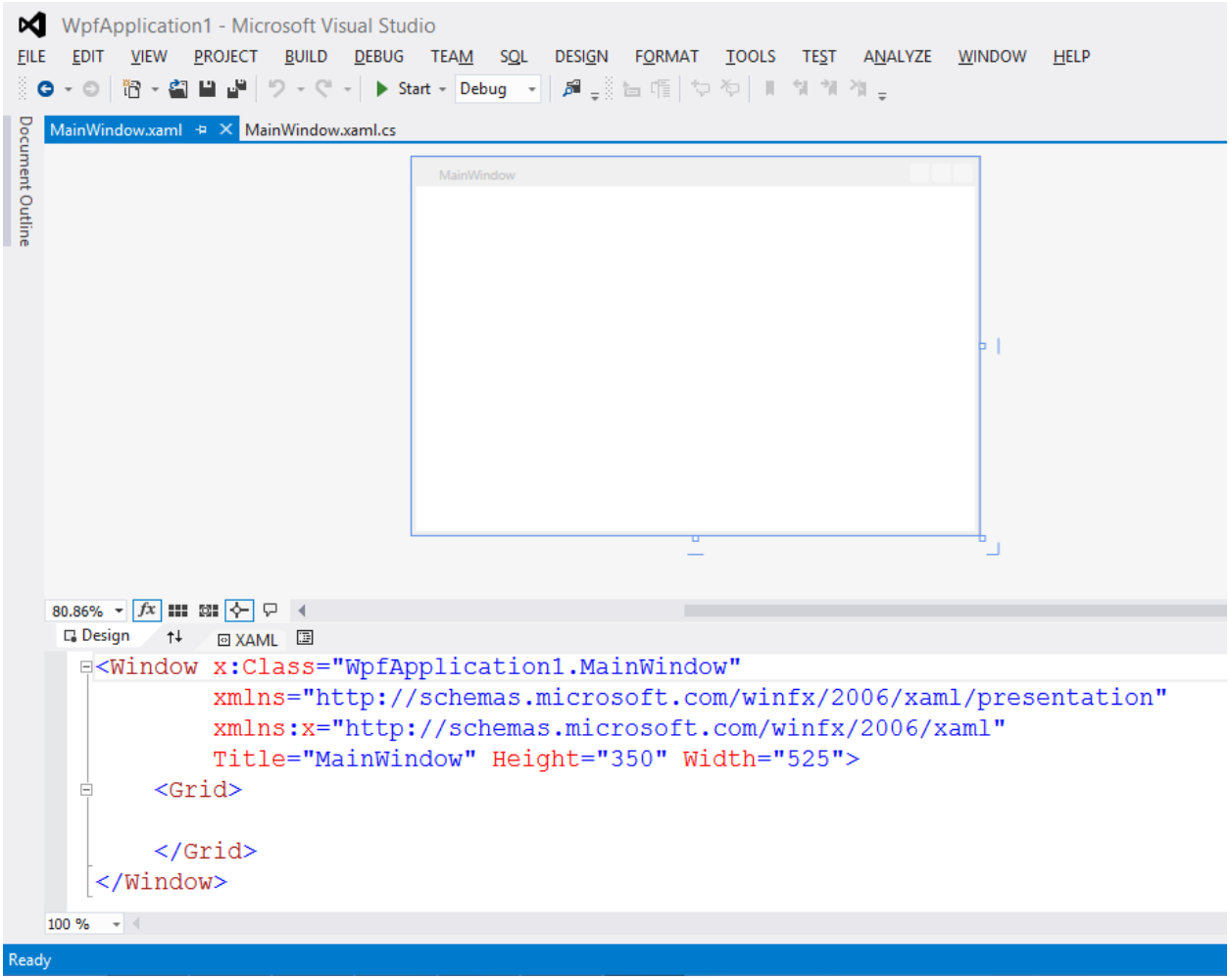

Рис. 14.1. **Вид вікна MS Visual Studio після створення проєкту WPF Application**

Додавання нової форми до проєкту:

1. У вікні Solution Explorer натисніть правою кнопкою миші вузол проєкту і виберіть Add → New Item ... . У вікні Add New Item виберіть категорію WPF і в списку, що з'явився – Window.

2. Введіть ім'я форми в поле Name.

3. Натисніть кнопку Add.

Як здійснити оброблення події для форми або елемента управління?

1. Відкрийте в редакторі файл з вихідним кодом форми.

2. Виділіть форму або елемент управління.

3. Відкрийте вікно властивостей: меню View → Propeties Window.
4. У вікні Properties Window натисніть кнопку "блискавка".
5. Виберіть потрібну подію, введіть ім'я методу-обробника.

Налаштування основних параметрів форми наведено у табл. 14.1.

Таблиця 14.1

Налаштування основних параметрів форми

Назва параметра	Необхідні дії
Заголовок вікна	Задати значення властивості Title у візуальному дизайнері або в XAML
Піктограма системного меню	Задати значення властивості **Icon** у візуальному дизайнері або в XAML
Курсор миші	Задати значення властивості **Cursor** у візуальному дизайнері або в XAML
Колір клієнтської області вікна	Задати значення властивості **Background** кореневого контейнера компонування
Координати лівого верхнього кута вікна (x, y)	Задати значення властивості **Left** та у візуальному дизайнері або в XAML (при цьому значення властивості **WindowStartupLocation** повинно мати значення **Manual**)
Ширина вікна	Задати значення властивості **Width** у візуальному дизайнері або в XAML
Висота вікна	Задати значення властивості **Hight** у візуальному дизайнері або в XAML
Початковий вид вікна	Задати значення властивості **WindowState** у візуальному дизайнері або в XAML

Завдання 1

Розробіть застосунок WPF з графічним інтерфейсом користувача.

Загальні вимоги до застосунка

1. Застосунок має одне вікно (головне вікно – Window).
2. Заголовок головного вікна – прізвище, ім'я, по батькові студента.
3. Колір клієнтської області вікна – встановлений за замовчуванням (рівень 1) або відповідає варіанту завдання (рівні 2 – 4).
4. Піктограма системного меню – встановлена за замовчуванням (рівні 1 і 2) або власна піктограма (файл *.gif, *.jpg, *.png) – рівні 3 і 4.
5. Курсор миші – встановлений за замовчуванням (рівень 1) або один із стандартних курсорів (властивість Cursor) відповідно до варіанта завдання – рівні 2 і 3 або власний курсор (завантажується з файла) – рівень 4.

6. Застосунок має обробляти одну з подій миші або натискання (відпускання) клавіш на клавіатурі комп'ютера відповідно до варіанта завдання.

У обробнику події натискання кнопки миші необхідно показати:

- стандартне діалогове вікно MessageBox з кнопкою OK і повідомленням про натиснену кнопку миші – рівень 1;
- стандартне діалогове вікно MessageBox з кнопкою OK і повідомленням, у якому відображаються поточні координати курсору миші в момент клацання, – рівень 2;
- стандартне діалогове вікно MessageBox з повідомленням, у якому відображаються поточні координати курсору миші в момент клацання і кнопки або піктограма відповідно до варіанта завдання – рівні 3 і 4.

У обробнику події натискання (відпускання) клавіш на клавіатурі необхідно:

- показати стандартне діалогове вікно MessageBox з кнопкою OK і повідомленням про натискання клавіші – рівень 1;
- визначити, яка клавіша була натиснута, і показати стандартне діалогове вікно MessageBox з кнопкою OK і повідомленням, у якому відображається символьний код цієї клавіші, – рівень 2;
- визначити, яка клавіша була натиснута і показати стандартне діалогове вікно MessageBox з повідомленням, у якому відображається символьний код цієї клавіші і кнопки або піктограма згідно з варіантом завдання – рівні 3 і 4.

Варіанти завдань наведено у табл. 14.2 – 14.16.

Таблиця 14.2

Варіант 1

Курсор миші	Стрілка з наконечником
Колір клієнтської області вікна	Червоний
Координати лівого верхнього кута вікна (x, y)	10, 10
Ширина вікна	100
Висота вікна	100
Початковий вигляд вікна	Розгорнуто на весь екран
Подія	Одинарне натискання лівої кнопки миші (подія MouseLeftButtonDown)
Кнопки вікна MessageBox	Yes, No, Cancel

Таблиця 14.3

Варіант 2

Курсор миші	Хрестик
Колір клієнтської області вікна	Помаранчевий
Координати лівого верхнього кута вікна (x, y)	50, 100
Ширина вікна	100
Висота вікна	150
Початковий вигляд вікна	Згорнуто
Подія	Подвійне клацання лівою кнопкою миші (використовуючи подію MouseDoubleClick)
Піктограма вікна MessageBox	Стоп

Таблиця 14.4

Варіант 3

Курсор миші	Гроно руки
Колір клієнтської області вікна	Жовтий
Координати лівого верхнього кута вікна (x, y)	100, 10
Ширина вікна	200
Висота вікна	100
Початковий вигляд вікна	Відповідає заданим розмірам
Подія	Одинарне натискання правої кнопки миші (подія MouseRightButtonDown)
Кнопка вікна MessageBox	OK

Таблиця 14.5

Варіант 4

Курсор миші	"Очікування"
Колір клієнтської області вікна	Зелений
Координати лівого верхнього кута вікна (x, y)	300, 200
Ширина вікна	400
Висота вікна	400
Початковий вигляд вікна	Розгорнуто на весь екран
Подія	Подвійне клацання правою кнопкою миші (використовуючи подію MouseDoubleClick)
Кнопки вікна MessageBox	OK, Cancel

Таблиця 14.6

Варіант 5

Курсор миші	Вертикальна стрілка
Колір клієнтської області вікна	Синій
Координати лівого верхнього кута вікна (x, y)	0, 0
Ширина вікна	799
Висота вікна	599
Початковий вигляд вікна	Згорнуто
Подія	Одинарне натискання лівої кнопки миші (подія MouseLeftButtonDown)
Кнопки вікна MessageBox	OK, Cancel

Таблиця 14.7

Варіант 6

Курсор миші	"Очікування"
Колір клієнтської області вікна	Фіолетовий
Координати лівого верхнього кута вікна (x, y)	555, 666
Ширина вікна	100
Висота вікна	300
Початковий вигляд вікна	Відповідає заданим розмірам
Подія	Подвійне клацання лівою кнопкою миші (використовуючи подію MouseDown)
Кнопки вікна MessageBox	Yes, No

Таблиця 14.8

Варіант 7

Курсор миші	Чотиристоронняя стрілка (північ – південь – захід – схід)
Колір клієнтської області вікна	Коричневий
Координати лівого верхнього кута вікна (x, y)	400, 300
Ширина вікна	200
Висота вікна	300
Початковий вигляд вікна	Розгорнуто на весь екран
Подія	Одинарне клацання правою кнопкою миші (використовуючи подію MouseDown)
Кнопки вікна MessageBox	Yes, No, Cancel

Таблиця 14.9

Варіант 8

Курсор миші	Двостороння стрілка (північний схід – південний захід)
Колір клієнтської області вікна	Бузковий
Координати лівого верхнього кута вікна (x, y)	20, 10
Ширина вікна	800
Висота вікна	500
Початковий вигляд вікна	Згорнуто
Подія	Подвійне клацання правою кнопкою миші (використовуючи подію MouseDoubleClick)
Піктограма вікна MessageBox	Знак оклику

Таблиця 14.10

Варіант 9

Курсор миші	Двостороння стрілка (північ – південь)
Колір клієнтської області вікна	Ліловий
Координати лівого верхнього кута вікна (x, y)	333, 444
Ширина вікна	500
Висота вікна	500
Початковий вигляд вікна	Відповідає заданим розмірам
Подія	Натискання будь-якої клавіші на клавіатурі (подія KeyDown)
Піктограма вікна MessageBox	Знак питання

Таблиця 14.11

Варіант 10

Курсор миші	Двостороння стрілка (північний захід – південний схід)
Колір клієнтської області вікна	Бірюзовий
Координати лівого верхнього кута вікна (x, y)	333, 77
Ширина вікна	333
Висота вікна	144
Початковий вигляд вікна	Розгорнуто на весь екран
Подія	Відпускання будь-якої клавіші на клавіатурі (подія KeyUp)
Кнопки вікна MessageBox	Yes, No

Таблиця 14.12

Варіант 11

Курсор миші	Вертикальна стрілка
Колір клієнтської області вікна	Сірий
Координати лівого верхнього кута вікна (x, y)	80, 180
Ширина вікна	156
Висота вікна	312
Початковий вигляд вікна	Згорнуто
Подія	Натискання будь-якої клавіші на клавіатурі (подія KeyDown)
Кнопки вікна MessageBox	Yes, No, Cancel

Таблиця 14.13

Варіант 12

Курсор миші	Очікування
Колір клієнтської області вікна	Рожевий
Координати лівого верхнього кута вікна (x, y)	69, 96
Ширина вікна	400
Висота вікна	700
Початковий вигляд вікна	Відповідає заданим розмірам
Подія	Одинарне натискання будь-якої кнопки миші (MouseDown)
Кнопки вікна MessageBox	Yes, No, Cancel

Таблиця 14.14

Варіант 13

Курсор миші	Двостороння стрілка (північний схід – південний захід)
Колір клієнтської області вікна	Червоний
Координати лівого верхнього кута вікна (x, y)	55, 0
Ширина вікна	200
Висота вікна	200
Початковий вигляд вікна	Знаходиться в центрі екрану, відповідає заданим розмірам
Подія	Відпускання будь-якої клавіші на клавіатурі (подія KeyUp)
Кнопка вікна MessageBox	OK

Таблиця 14.15

Варіант 14

Курсор миші	Двостороння стрілка (північний захід – південний схід)
Колір клієнтської області вікна	Білий
Координати лівого верхнього кута вікна (x, y)	0, 0
Ширина вікна	300 (фіксований розмір)
Висота вікна	500 (фіксований розмір)
Початковий вигляд вікна	Відповідає заданим розмірам
Подія	Одинарне натискання будь-якої кнопки миші (MouseDown)
Кнопки вікна MessageBox	Retry, Cancel

Таблиця 14.16

Варіант 15

Курсор миші	Гроно руки
Колір клієнтської області вікна	Чорний
Координати лівого верхнього кута вікна (x, y)	10, 30
Ширина вікна	100
Висота вікна	100
Початковий вигляд вікна	Згорнуто
Подія	Відпускання будь-якої клавіші на клавіатурі (подія KeyUp)
Кнопки вікна MessageBox	Yes, No

Завдання 2

Розробіть калькулятор з графічним інтерфейсом користувача (модальне діалогове вікно фіксованого розміру).

Діалогове вікно калькулятора є головним вікном застосунку та має з'являтися після його запуску (рівень 1).

Для відображання діалогового вікна калькулятора у головному вікні застосунку з завдання 1 необхідно додати:

- панель меню з меню "Інструменти" і пунктом "Калькулятор" (рівні 2 – 4);
- панель інструментів з кнопкою та зображенням, після натискання на яку викликається обробник пункту меню "Калькулятор" (рівні 3 і 4);
- панель стану, в якій під час наведення на пункт меню "Калькулятор" відображається назва поточного режиму роботи програми (рівень 4).

Графічний інтерфейс "калькулятора" для рівнів 1 і 2 повинен мати приблизно, як на рис. 14.2. Для цього рекомендується компоновка елементів графічного інтерфейсу "калькулятору", наведена на рис. 14.3.

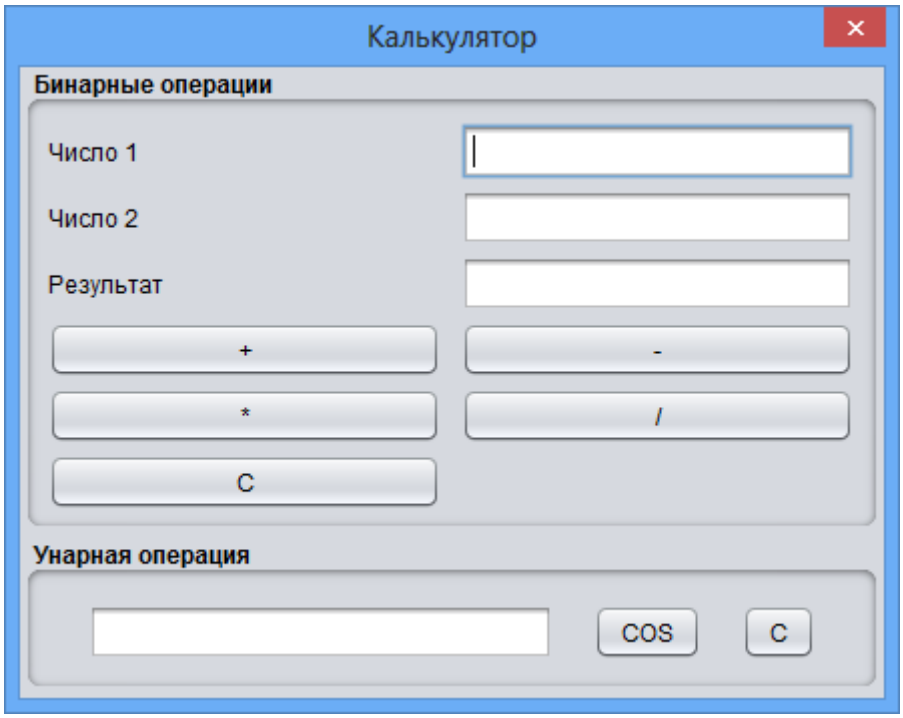

Рис. 14.2. **Графічний інтерфейс для рівнів 1 і 2**

Графічний інтерфейс "калькулятора" для рівнів 3 і 4 має виглядати приблизно, як на рис. 14.4. Для цього рекомендується компоновка елементів графічного інтерфейсу "калькулятору", наведена на рис. 14.5.

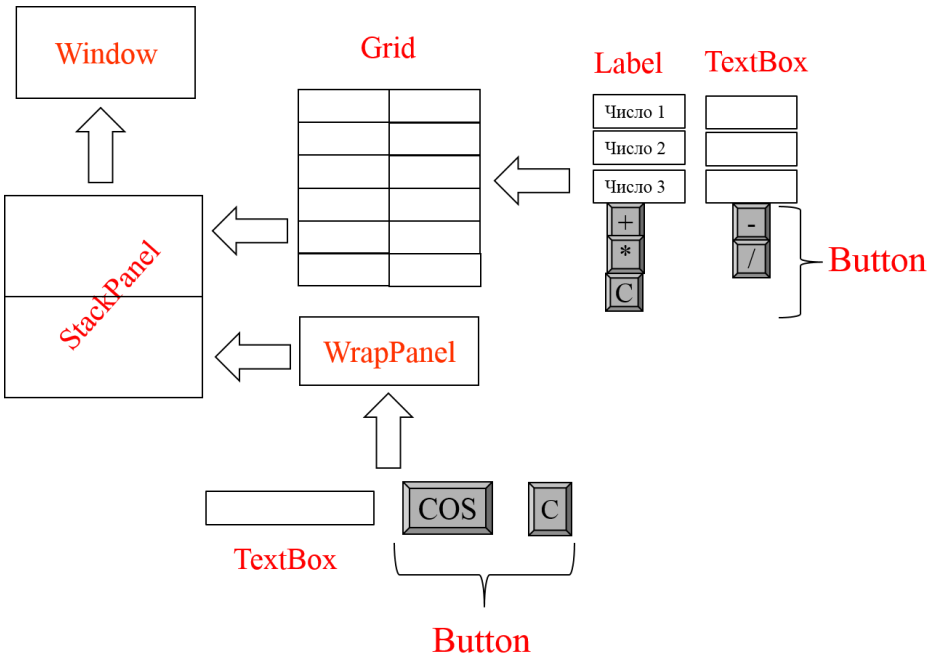

Рис. 14.3. **Компоновка елементів графічного інтерфейсу для рівнів 1 і 2**

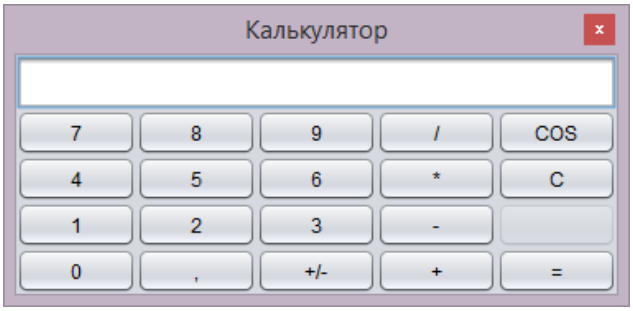

Рис. 14.4. **Графічний інтерфейс для рівнів 3 і 4**

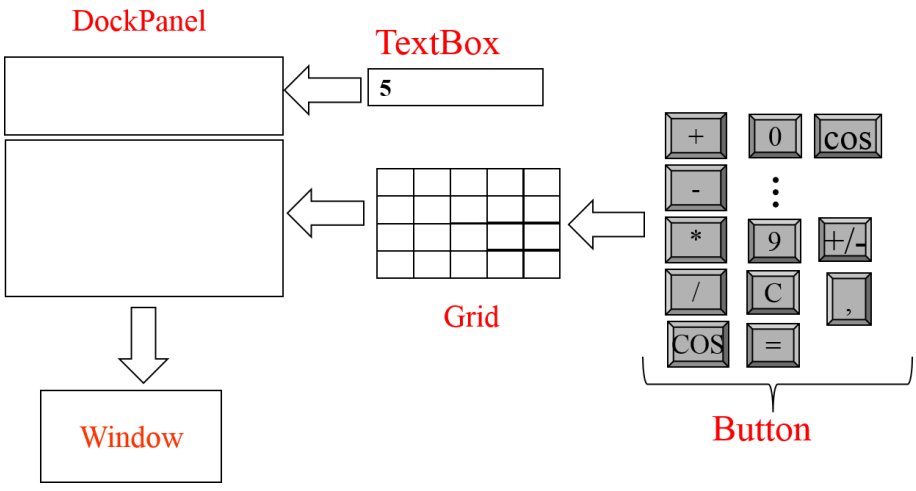

Рис. 14.5. **Компоновка елементів графічного інтерфейсу для рівнів 3 і 4**

На рис. 14.3, 14.5 Window, TextField, Label, Button – елементи управління; DockPanel, WrapPanel, Grid, StackPanel – панелі компонування.

"Калькулятор" має виконувати будь-яку з чотирьох арифметичних дій: "+", "–", "×", "/" над двома дійсними числами (рівні 1 – 4). Він також повинен виконувати додаткову операцію, яка визначається варіантом завдання (рівні 2 – 4).

"Калькулятор" не має дозволяти користувачеві вводити значення вихідних даних, які не відповідають обмеженням, зазначеним у варіантах завдань (рівень 4).

Варіанти додаткової операції наведено в табл. 14.17.

Таблиця 14.17

Варіанти завдань

Номер варіанта	Додаткова операція калькулятора
1	$a^{1/2}$, де $a > 0$
2	Перетворення аргументу a (кілометр → миля), де $1 \le a \le 1\,000$
3	exp(a), де $-5 \le a \le 5$
4	Перетворення аргументу a (Цельсій → Фаренгейт), де $-273 \le a \le 273$
5	lg (a), де $a > 0$
6	Перетворення аргументу a (байт → кілобайт), де $1 \le a \le 10^9$
7	sin(a), де $0° \le a \le 360°$
8	cos(a), де $0° \le a \le 360°$
9	tg(a), де $-90° < a < 90°$
10	Перетворення аргументу a (гривня → євро), де $1 \le a \le 100\,000$
11	ctg(a), де $-180° < x < 180°$
12	a^b, де $a > 0$, $b > 0$
13	ln(a), де $a > 0$
14	Перетворення аргументу a (радіани → градуси), де $0 < a < 6{,}28$
15	Перетворення аргументу a (градуси → радіани), де $0° \le a \le 360°$

Примітка. Потрібно врахувати, що стандартні методи класу Math для обчислення тригонометричних функцій очікують параметр у радіанах, а вихідні дані мають вводитися користувачем у градусах.

Контрольні запитання до лабораторної роботи 14

1. Яка загальна структура застосунку з графічним інтерфейсом користувача на платформі .NET?
2. Який порядок додавання елемента управління на форму?
3. Які класи складають ієрархію класів WPF?
4. Які елементи управління належать до візуальних елементів управління форми?
5. Який порядок використання компоненту MenuStrip?
6. Який порядок використання компоненту ToolStrip?

Рекомендована література

1. Блинов И. Н. Java. Методы программирования : учеб.-метод. пособ. / И. Н. Блинов, В. С. Романчик. – Минск : Изд-во "Четыре четверти", 2013. – 768 с.
2. Объектно-ориентированный анализ и проектирование с примерами приложений / Г. Буч, Р. Максимчук, М. Энгл и др. ; пер. с англ. – Москва : ИД "Вильямс", 2008. – 720 с.
3. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – Санкт-Петербург: Питер, 2007. – 366 с.
4. Щербаков О. В. Основи об'єктно-орієнтованого програмування : навч. посіб. / О. В. Щербаков, Ю. Е. Парфьонов, В. М. Федорченко. – Харків : ХНЕУ ім. С. Кузнеця, 2019. – 237 с.
5. Michaelis M. PowerPoint Slides for Essential C# 6.0 / M. Michaelis, E. Lippert. – Boston : Addison-Wesley, 2016. – 1004 p.

Інформаційні ресурси

6. Парфьонов Ю. Е. Методичні рекомендації до виконання лабораторних робіт з навчальної дисципліни "Основи об'єктно-орієнтованого програмування" [Електронний ресурс] / Ю. Е. Парфьонов. – Режим доступу : http://www.ikt.hneu.edu.ua/course/view.php?id=879.
7. Парфьонов Ю. Е. Основи об'єктно-орієнтованого програмування: презентації лекцій [Електронний ресурс] / Ю. Е. Парфьонов. – Режим доступу : http://www.ikt.hneu.edu.ua/course/view.php?id=879.

8. Уроки Java для начинающих [Электронный ресурс]. – Режим доступа : http://cybern.ru/category/java/begin-java.

9. Programming Tutorials and Source Code Examples [Electronic resource]. – Access mode : http://www.java2s.com.

10. Design Patterns [Electronic resource]. – Access mode : www.oodesign.com.

11. Design Patterns [Electronic resource]. – Access mode : http://sourcemaking.com/design_patterns.

12. .NET Design Patterns [Electronic resource]. – Access mode : https://www.dofactory.com/net/design-patterns.

13. The premier Website about regular expressions [Electronic resource]. – Access mode : http://www.regular-expressions.info.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Методичні рекомендації до лабораторних робіт

для студентів спеціальності 121 "Інженерія програмного забезпечення" першого (бакалаврського) рівня

Самостійне електронне текстове мережеве видання

Укладачі: **Парфьонов** Юрій Едуардович
Щербаков Олександр Всеволодович

Відповідальний за видання *І. О. Ушакова*

Редактор *А. С. Ширініна*

Коректор *Н. В. Завгородня*

План 2021 р. Поз. № 105 ЕВ. Обсяг 93 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру
ДК № 4853 від 20.02.2015 р.