МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені В. Н. КАРАЗІНА

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

Методичні рекомендації до практичних занять
і самостійної роботи з дисципліни для здобувачів вищої освіти
першого (бакалаврського) рівня галузі знань 12 «Інформаційні технології» за
спеціальністю 122 «Комп'ютерні науки» за освітньо-професійною програмою
«Комп'ютерні науки та інформаційні технології в бізнесі»

Електронний ресурс

Харків – 2026

УДК 004.4(075.8)
С 79

Рецензенти:

В. О. Гороховатський – д.т.н., професор, професор кафедри інформаційних технологій та математичного моделювання Харківського національного університету імені В. Н. Каразіна;

О. А. Кобилін – к.т.н., доцент, завідувач кафедри інформатики Харківського національного університету радіоелектроніки.

Затверджено до розміщення в мережі Інтернет рішенням Науково-методичної ради Харківського національного університету імені В. Н. Каразіна (протокол № 9 від 24 квітня 2026 року)

С 79 **Спеціалізовані** мови програмування : методичні рекомендації до практичних занять і самостійної роботи з дисципліни для здобувачів вищої освіти першого (бакалаврського) рівня галузі знань 12 «Інформаційні технології» за спеціальністю 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні науки та інформаційні технології в бізнесі» [Електронний ресурс] / уклад. Д. М. Ковальчук. – Харків : ХНУ імені В. Н. Каразіна, 2026. – (PDF 65 с.)

Методичні рекомендації до практичних занять і самостійної роботи з дисципліни «Спеціалізовані мови програмування» розкривають основні аспекти підготовки здобувачів до практичних занять та виконання самостійної роботи. У виданні подано тематичний план курсу, детальні методичні вказівки до виконання практичних завдань мовою програмування Java, завдання для самостійної роботи, питання для підготовки до підсумкового контролю знань, а також систему оцінювання результатів навчальної діяльності й перелік рекомендованих джерел для поглибленого вивчення дисципліни.

Методичні рекомендації призначені для здобувачів першого (бакалаврського) рівня вищої освіти денної форми навчання за галуззю знань 12 «Інформаційні технології», спеціальністю 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні науки та інформаційні технології в бізнесі».

УДК 004.9:004.415

ЗМІСТ

ВСТУП

Дисципліна «Спеціалізовані мови програмування» є важливою складовою освітньо-професійної програми «Комп’ютерні науки та інформаційні технології в бізнесі» підготовки фахівців першого (бакалаврського) рівня вищої освіти за спеціальністю 122 «Комп’ютерні науки» галузі знань 12 «Інформаційні технології».

У сучасному цифровому світі, де автоматизація процесів, інтеграція інформаційних систем і розробка програмного забезпечення мають ключове значення, володіння сучасними мовами програмування є обов’язковою компетенцією майбутнього фахівця з комп’ютерних наук. Мова Java займає провідні позиції серед інструментів розробки корпоративних, веб- та мобільних застосунків, завдяки своїй надійності, кросплатформеності та широкому спектру бібліотек і фреймворків.

Методичні рекомендації розроблено для здобувачів вищої освіти першого (бакалаврського) рівня денної форми навчання з метою підтримки їхньої самостійної роботи та якісної підготовки до практичних занять. У методичному виданні представлено тематичний план дисципліни, покрокові методичні вказівки до виконання практичних завдань і самостійної роботи здобувачів, орієнтовні питання для залікового контролю знань, а також критерії оцінювання результатів навчальної діяльності.

1. Опис навчальної дисципліни

Навчальна дисципліна «Спеціалізовані мови програмування» викладається згідно з освітньо-професійною програмою «Комп'ютерні науки та інформаційні технології в бізнесі» підготовки здобувачів вищої освіти першого (бакалаврського) рівня за спеціальністю 122 «Комп'ютерні науки» галузі знань 12 «Інформаційні технології».

Метою викладання дисципліни є підготовка фахівців, які володіють фундаментальними теоретичними знаннями та здатні вирішувати практичні завдання розробки програмного забезпечення мовою Java, а також ознайомлені з можливостями мови програмування для створення спеціалізованого програмного забезпечення.

Опанування цієї дисципліни забезпечує теоретичну й практичну підготовку майбутніх фахівців у сфері програмної інженерії та інформаційних технологій, зосереджуючи увагу на ефективному використанні мови Java для створення прикладного, графічного, багатопотокового та об'єктно-орієнтованого програмного забезпечення.

Основні завдання вивчення дисципліни.

Вивчення дисципліни «Спеціалізовані мови програмування» спрямоване на досягнення таких основних завдань:

- ознайомлення з принципами побудови програм мовою Java;
- вільне володіння навичками застосування конструкцій програмування;
- опанування навичок з обробки структур даних (масивів, колекцій, файлів);
- опанування навичок візуального програмування із застосуванням бібліотеки Swing;
- вивчення засобів обробки винятків, роботи з потоками введення-виведення та багатопотоковості;
- набуття практичних умінь створення Java-додатків з графічним інтерфейсом;

- розуміння принципів об'єктно-орієнтованого програмування та реалізація його концепцій у прикладному середовищі.

Кількість кредитів -4

Загальна кількість годин -120

Таблиця 1

Характеристика навчальної дисципліни

Обов’язкова / за вибором	
Денна форма навчання	Заочна (дистанційна) форма навчання
Рік підготовки	
3-й	-й
Семестр	
5-й	-й
Лекції	
32 год.	год.
Практичні, семінарські заняття	
32/22* год.	год.
Лабораторні заняття	
год.	год.
Самостійна робота	
56/66 год.	год.
у тому числі індивідуальні завдання	
год.	

* у разі формування малочисельних груп обсяг аудиторного навчального навантаження відведеного на вивчення навчальної дисципліни зменшується відповідно до Положення про планування й звітування науково-педагогічних працівників Харківського національного університету імені В. Н. Каразіна.

Перелік компетентностей, що формує дана дисципліна.

Вивчення навчальної дисципліни «Спеціалізовані мови програмування» спрямовано на формування таких компетентностей:

Інтегральна компетентність.

Здатність розв’язувати складні спеціалізовані задачі та практичні проблеми у галузі комп’ютерних наук або у процесі навчання, що передбачає застосування теорій та методів інформаційних технологій і характеризується комплексністю та невизначеністю умов.

Загальні компетентності

ЗК 1 Здатність до абстрактного мислення, аналізу та синтезу.

ЗК 2 Здатність застосовувати знання у практичних ситуаціях.

ЗК 3 Знання та розуміння предметної області та розуміння професійної діяльності.

ЗК 4 Здатність спілкуватися державною мовою як усно, так і письмово.

ЗК 5 Здатність спілкуватися іноземною мовою.

ЗК7 Здатність до пошуку, оброблення та аналізу інформації з різних джерел.

ЗК 8 Здатність генерувати нові ідеї (креативність).

ЗК 11 Здатність приймати обґрунтовані рішення.

Фахові компетентності

ФК 1 Здатність до математичного формулювання та досліджування неперервних та дискретних математичних моделей, обґрунтовування вибору методів і підходів для розв’язування теоретичних і прикладних задач у галузі комп’ютерних наук, аналізу та інтерпретування.

ФК 3 Здатність до логічного мислення, побудови логічних висновків, використання формальних мов і моделей алгоритмічних обчислень, проектування, розроблення й аналізу алгоритмів, оцінювання їх ефективності та складності, розв’язності та нерозв’язності алгоритмічних проблем для адекватного моделювання предметних областей і створення програмних та інформаційних систем.

ФК 4 Здатність використовувати сучасні методи математичного моделювання об’єктів, процесів і явищ, розробляти моделі й алгоритми чисельного розв’язування задач математичного моделювання, враховувати похибки наближеного чисельного розв’язування професійних задач.

ФК 5 Здатність здійснювати формалізований опис задач дослідження операцій в організаційно-технічних і соціально-економічних системах різного призначення, визначати їх оптимальні розв’язки, будувати моделі оптимального управління з урахуванням змін економічної ситуації, оптимізувати процеси управління в системах різного призначення та рівня ієрархії.

ФК 8 Здатність проектувати та розробляти програмне забезпечення із застосуванням різних парадигм програмування: узагальненого, об’єктно-орієнтованого, функціонального, логічного, з відповідними моделями, методами й алгоритмами обчислень, структурами даних і механізмами управління.

ФК 13 Здатність до розробки мережевого програмного забезпечення, що функціонує на основі різних топологій структурованих кабельних систем, використовує комп’ютерні системи і мережі передачі даних та аналізує якість роботи комп’ютерних мереж.

ФКД 1 Здатність до креативного підходу в процесі проектування, розробки та використання програмних додатків у фінансових технологіях.

Перелік результатів навчання, що формує дана дисципліна:

РН1 Застосовувати знання основних форм і законів абстрактно-логічного мислення, основ методології наукового пізнання, форм і методів вилучення, аналізу, обробки та синтезу інформації в предметній області комп'ютерних наук.

РН2 Використовувати сучасний математичний апарат неперервного та дискретного аналізу, лінійної алгебри, аналітичної геометрії, в професійній діяльності для розв’язання задач теоретичного та прикладного характеру в процесі проектування та реалізації об’єктів інформатизації.

РН5 Розуміти принципи моделювання організаційнотехнічних систем і операцій; використовувати методи дослідження операцій, розв’язання одно – та багатокритеріальних оптимізаційних задач лінійного, цілочисельного, нелінійного, стохастичного програмування.

РН6 проектувати, розробляти та аналізувати алгоритми розв’язання обчислювальних та логічних задач, оцінювати ефективність та складність алгоритмів на основі застосування формальних моделей алгоритмів та обчислюваних функцій.

РН9 Розробляти програмні моделі предметних середовищ, вибирати парадигму програмування з позицій зручності та якості застосування для реалізації методів та алгоритмів розв’язання задач в галузі комп’ютерних наук.

РН10 Використовувати інструментальні засоби розробки клієнт-серверних застосувань, проектувати концептуальні, логічні та фізичні моделі баз даних, розробляти та оптимізувати запити до них, створювати розподілені бази даних, сховища та вітрини даних, бази знань, у тому числі на хмарних сервісах, із застосуванням мов веб-програмування.

РН13 Володіти мовами системного програмування та методами розробки програм, що взаємодіють з компонентами комп’ютерних систем, знати мережні технології, архітектури комп’ютерних мереж, мати практичні навички технології адміністрування комп’ютерних мереж та їх програмного забезпечення

РН15 Розуміти концепцію інформаційної безпеки, принципи безпечного проектування програмного забезпечення, забезпечувати безпеку комп’ютерних мереж в умовах неповноти та невизначеності вихідних даних.

Пререквізити: навчальна дисципліна базується на знаннях таких дисциплін, як: «Дискретна математика», «Основи алгоритмізації та програмування», «Алгоритми і структури даних», «Об’єктно-орієнтоване програмування».

2. Тематичний план навчальної дисципліни

Розділ 1. *Основи мови програмування Java*

Тема 1. Огляд технологій JAVA і IDE

Технологія Java. Платформи JAVA і Java IDE. IntelliJ IDEA. Eclipse.

NetBeans IDE.

Тема 2. Основи синтаксису Java

Особливості синтаксису Java. Ключові слова (key words). Літерали (константи). Керуючі конструкції. Введення-виведення. Примітивні типи даних. Змінні.

Тема 3. Класи в Java

Опис класу. Успадкування і поліморфізм. Клас Object. Рефлексія і клас Class. Абстрактні класи та інтерфейси. Інтерфейси для порівняння об'єктів.

Тема 4. Класи стандартної бібліотеки.

Бібліотеки і пакети. Основні класи пакета java.lang. Класи-колекції. Виключення.

Тема 5. Робота з файловою системою

Потоки введення – виведення. Серіалізація. Маніпулювання файлами і директоріями.

Розділ 2. *Створення графічних додатків та обробка даних в Java*

Тема 1. Створення GUI. Бібліотека Swing.

Можливості та особливості різних бібліотек GUI. Компонент і контейнер. Верстка форми і класи розміщень (layouts). Поняття і принципи usability. Розташування елементів екранної форми. Робота з GUI редактором IDEA. GUI редактор WindowBuilder для IDE Eclipse. Класи графічних компонентів. Прості компоненти Swing. Події компонентів Swing. Приклад – простий текстовий редактор. Робота з меню. Стандартні діалогові вікна.

Тема 2. Реалізація багатопотоковості в Java.

Клас Thread. Інтерфейс Runnable. Синхронізація потоків synchronized.

Структуру навчальної дисципліни «Теорія інформації та кодування» наведено в таблиці 2.

Таблиця 2

Структура навчальної дисципліни

Назви розділів і тем	Кількість годин					
	денна форма					
	усього	у тому числі				
		л	п	лаб.	інд.	с. р.
1	2	3	4	5	6	7
Розділ 1. Основи мови програмування Java						
Тема 1. Огляд технологій JAVA і IDE	15	4	4/3*			7/8*
Тема 2. Основи синтаксису Java	15	4	4/3*			7/8*
Тема 3. Класи в Java	15	4	4/3*			7/8*
Тема 4. Класи стандартної бібліотеки.	15	4	4/3*			7/8*
Тема 5. Робота з файловою системою	15	4	4/3*			7/8*
Разом за розділом 1	75	20	20/15*			35/40*
Розділ 2. Створення графічних додатків та обробка даних в Java						
Тема 1. Створення GUI. Бібліотека Swing.	30	8	8/4*			14/18*
Тема 2. Реалізація багатопотоковості в Java.	15	4	4/3*			7/8*
Разом за розділом 2	45	12	12/7*			21/26*
Усього годин	120	32	32/22*			56/66*

* у разі формування малочисельних груп обсяг аудиторного навчального навантаження відведеного на вивчення навчальної дисципліни зменшується відповідно до Положення про планування й звітування науково-педагогічних працівників Харківського національного університету імені В. Н. Каразіна.

3. Методичні рекомендації з підготовки до практичних занять

У межах робочої програми навчальної дисципліни «Спеціалізовані мови програмування» передбачено проведення практичних занять за темами, наведеними в таблиці 3.

Таблиця 3

Теми практичних занять

№ з/п	Назва теми	Кількість годин
1	Установка віртуальної машини Java й інтегрованого середовища розробки Eclipse	2
2	Команди керування та масиви в Java	3/2*
3	Обробка рядків. Використання регулярних виразів в Java-додатках	3/2*
4	Класи та об’єкти в Java. Абстрактні класи та інтерфейси	3/2*
5	Винятки. Обробка винятків	3/2*
6	Операції з файлами. Файлові потоки введення-виведення в Java	3/2*
7	Розробка графічного інтерфейсу в Java	3/2*
8	Робота із зображеннями в Java	3/2*
9	Обробка математичних функцій та побудова графіків функцій в Java	3/2*
10	Реалізація багатопотоковості в Java	3/2*
11	Класи-колекції	3/2*
	Разом	32/22*

* у разі формування малочисельних груп обсяг аудиторного навчального навантаження відведеного на вивчення навчальної дисципліни зменшується відповідно до Положення про планування й звітування науково-педагогічних працівників Харківського національного університету імені В. Н. Каразіна.

Нижче представлені завдання до кожної з тем практичних занять.

Завдання до теми №1. Установка віртуальної машини Java й інтегрованого середовища розробки Eclipse

1) Встановити JDK останньої стабільної версії.

2) Налаштувати системні змінні середовища (JAVA_HOME, Path).

3) Завантажити та встановити Eclipse IDE для Java Developers.

4) Створити перший проєкт у Eclipse та перевірити правильність встановлення середовища.

5) Створити консольний додаток HelloWorld.java, скомпілювати і

виконати його.

Порядок виконання завдання:

1) Завантаження JDK:

- Перейдіть на офіційний сайт https://www.oracle.com/java/technologies/javase-downloads.html
- Виберіть відповідну версію JDK під вашу ОС (Windows/Linux/macOS).
- Завантажте та виконайте інсталяцію.

2) Налаштування змінних середовища (Windows):

- Відкрийте «Панель керування» → «Система» → «Додаткові параметри системи» → «Змінні середовища».
- Додайте змінну системи:
- JAVA_HOME = шлях до встановленої JDK (наприклад: `C:\Program Files\Java\jdk-21`)
- У змінну Path додайте `%JAVA_HOME%\bin`

3) Перевірка інсталяції JDK:

- Відкрийте командний рядок (cmd).
- Введіть:

```
java -version
javac -version
```

- У відповідь має з'явитися інформація про версію Java та компілятора.

4). Інсталяція Eclipse IDE:

- Завантажте Eclipse із сайту: https://www.eclipse.org/downloads/
- Виберіть «Eclipse IDE for Java Developers».
- Встановіть програму, виберіть JDK, вказаний у системі.

5. Створення першого Java-проєкту:

- Запустіть Eclipse.
- Створіть новий проєкт: File → New → Java Project.

- Введіть назву проєкту (наприклад, FirstProject) → Finish.
- Створіть клас: New → Class → HelloWorld, встановіть опцію public static void main(String[] args).
- У тілі методу main() введіть код:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

- Запустіть програму (Run → Run As → Java Application).

Теоретичні питання для самоперевірки

1) Що таке JDK, JVM і JRE? У чому їхні відмінності?

2) Для чого потрібні змінні середовища JAVA_HOME і Path?

3) Які IDE використовуються для розробки Java-програм?

4) Які основні особливості Eclipse IDE?

5) Як створити Java-проєкт і додати до нього клас?

6) Що таке метод main() у програмі Java?

7) Як відбувається компіляція та виконання Java-програми?

Завдання до теми №2. Команди керування та масиви в Java

Виконайте практичне завдання відповідно до вашого варіанту.

1) Заповнити матрицю A(n,n) випадковими числами від 1 до n^2 по рядках. Знайти суму елементів побічної діагоналі та поміняти місцями головну та побічну діагоналі.

2) Заповнити матрицю випадковими цілими числами в діапазоні [-50;50]. Порахувати кількість від'ємних чисел у кожному рядку. Якщо їх більше за 2 - обнулити відповідний рядок.

3) Створити матрицю, у якій значення кожного елемента дорівнює добутку його індексів (i*j). Знайти мінімальний елемент матриці та обміняти

його з центральним елементом.

4) Заповнити матрицю A(n,m) так, щоб кожен елемент дорівнював сумі номерів рядка та стовпця. Знайти рядок із найбільшою сумою елементів і поміняти його з першим рядком.

5) Заповнити матрицю парними числами за зростанням. Порахувати кількість чисел, що кратні 4, та поміняти місцями перший і останній стовпець.

6) Заповнити матрицю A(n,m) випадковими значеннями. Знайти середнє значення в кожному стовпці. Якщо середнє менше за заданий поріг, поміняти цей стовпець з останнім.

8) Заповнити матрицю A(n,n) за законом: A[i][j] = i + j. Знайти рядок із найбільшою кількістю парних чисел і обміняти його з останнім рядком.

9) Заповнити матрицю квадратами натуральних чисел. Знайти середнє арифметичне всіх непарних елементів і поміняти місцями стовпці з номерами 1 і 2, якщо воно більше за 100.

10) Заповнити матрицю випадковими значеннями з плаваючою крапкою. Знайти елемент із найбільшим значенням у кожному рядку та поміняти його місцями з першим елементом у цьому рядку.

11) Створити матрицю A(n,n), де всі елементи дорівнюють сумі квадратів їхніх індексів. Знайти елемент із найбільшим значенням на головній діагоналі та обміняти його з елементом у правому нижньому кутку.

12) Заповнити квадратну матрицю A(n, n) значеннями за законом: $A[i][j]=i^2-j^2$. Обчислити кількість рядків, у яких сума елементів перевищує задане число k і обнулити всі елементи, розташовані вище головної діагоналі.

Приклад. Заповнити матрицю A(n, m) випадковими числами Впорядкувати кожен рядок матриці A(n, m) за спаданням, якщо сума його елементів є парною, і за зростанням − якщо непарною.

```
import java.util.Scanner;
import java.util.Random;
```

```
public class Matrix {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Введення розмірів матриці
        System.out.print("Enter number of rows (n): ");
        int n = scanner.nextInt();
        System.out.print("Enter number of columns (m): ");
        int m = scanner.nextInt();
        int[][] matrix = new int[n][m];
        Random rand = new Random();
        // Заповнення матриці випадковими числами від 0 до 20
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                matrix[i][j] = rand.nextInt(21); // 0-20
            }
        }
        System.out.println("\nOriginal matrix:");
        printMatrix(matrix, n, m);
        // Обробка кожного рядка
        for (int i = 0; i < n; i++) {
            int sum = 0;
            for (int j = 0; j < m; j++) {
                sum += matrix[i][j];
            }
            System.out.println("Sum of elements in row " + (i + 1) +
": " + sum);
            // Якщо сума парна – сортуємо за спаданням
            if (sum % 2 == 0) {
                sortDescending(matrix[i], m);
            }
            // Якщо сума непарна – сортуємо за зростанням
            else {
                sortAscending(matrix[i], m);
            }
        }
        System.out.println("\nModified matrix:");
        printMatrix(matrix, n, m);
    }
```

```
    // Метод для виведення матриці
    public static void printMatrix(int[][] matrix, int n, int m) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                System.out.printf("%4d", matrix[i][j]);
            }
            System.out.println();
        }
    }
    // Сортування за зростанням (бульбашкове сортування)
    public static void sortAscending(int[] row, int m) {
        for (int i = 0; i < m - 1; i++) {
            for (int j = 0; j < m - 1 - i; j++) {
                if (row[j] > row[j + 1]) {
                    int temp = row[j];
                    row[j] = row[j + 1];
                    row[j + 1] = temp;
                }
            }
        }
    }
    // Сортування за спаданням (бульбашкове сортування)
    public static void sortDescending(int[] row, int m) {
        for (int i = 0; i < m - 1; i++) {
            for (int j = 0; j < m - 1 - i; j++) {
                if (row[j] < row[j + 1]) {
                    int temp = row[j];
                    row[j] = row[j + 1];
                    row[j + 1] = temp;
                }
            }
        }
    }
}
```

```
Enter number of rows (n): 5
Enter number of columns (m): 7

Original matrix:
  10  19  17   8   5   7  18
  11  18   7   0   5  20  10
   7  11   5   2   7  11  18
  18  17   1  16   2  10   9
   2  13  12  12   8   1  17
Sum of elements in row 1: 84
Sum of elements in row 2: 71
Sum of elements in row 3: 61
Sum of elements in row 4: 73
Sum of elements in row 5: 65

Modified matrix:
  19  18  17  10   8   7   5
   0   5   7  10  11  18  20
   2   5   7   7  11  11  18
   1   2   9  10  16  17  18
   1   2   8  12  12  13  17
```

Рисунок 1 – Результат компіляції програми

Підготуйте звіт за результатами виконання завдання, який має містити:

- алгоритм розв’язання поставленої задачі;

- програмний код з коментарями;

- скріншоти результатів виконання програми.

До звіту обов’язково додайте архів з файлами проекту.

Теоретичні питання для самоперевірки

1) Яке призначення умовного оператора if у мові Java?

2) Яка різниця між операторами if та if-else?

3) Як працює оператор switch у Java? Який тип значення допускається у виразі switch?

4) Яке призначення оператора break?

5) Для чого використовується гілка default в операторі switch? Чи обов’язкова вона?

6) Поясніть різницю між циклами while і do-while. У яких випадках доцільно використовувати кожен із них?

7) Як працює цикл for у Java? З яких частин він складається?

8) Як оголошується масив в Java? Наведіть приклад.

9) Як здійснюється ініціалізація масиву в Java?

10) Для чого використовується властивість .length у масиві?

11) Які правила індексації масиву в Java? Що станеться при спробі звернутися до елементу поза межами масиву?

12) Як описати та ініціалізувати двовимірний (ступінчастий) масив у Java? Чим відрізняється ступінчастий масив (jagged array) від класичного прямокутного двовимірного масиву?

Завдання до теми № 3. Обробка рядків. Використання регулярних виразів в Java-додатках

Виконайте практичне завдання відповідно до вашого варіанту.

Варіант 1

а) Дано речення. Зробити перетворення, щоб перша літера кожного слова була великою.

б) В тексті знайти всі слова, що мають три однакові букви поспіль (наприклад: «меее», «нооо»).

Варіант 2

а) Дано текст, у якому слова розділені випадковою кількістю пробілів. Видалити зайві пробіли (залишити лише один між словами).

б) Визначити всі дати у форматі дд.мм.рррр і перевірити їх на коректність (враховуючи місяць і день).

Варіант 3

а) Дано речення. Вилучити всі голосні букви.

б) Знайти всі електронні адреси у тексті та замінити символ @ на [at].

Варіант 4

а) Дано список прізвищ через кому. Впорядкувати їх за алфавітом.

б) Визначити, чи містить рядок дійсний номер банківської картки (16 цифр без пробілів або з пробілами через кожні 4 цифри).

Варіант 5

а) Дано рядок. Замінити кожну цифру в ньому на відповідну кількість зірочок (наприклад, «3» → «***»).

б) Визначити всі URL-адреси в тексті та вивести їх список.

Варіант 6

а) Дано текст. Замінити всі входження подвійних лапок (" ") на одинарні (‘ ’).

б) Знайти в тексті всі правильні номери паспортів у форматі: AA123456 (2 великі літери + 6 цифр).

Варіант 7

а) У слові знайти кількість повторень кожної літери.

б) Визначити, чи є рядок дійсною URL-адресою (http/https, домен, крапка, зона).

Варіант 8

а) Видалити з рядка всі символи, які не є літерами або цифрами.

б) Перевірити, чи є рядок дійсною MAC-адресою: формат XX:XX:XX:XX:XX:XX, де X – шістнадцяткова цифра.

Варіант 9

а) Дано рядок. Після кожного знаку оклику (!) додати ще два.

б) Знайти всі IP-адреси, які належать до підмережі 192.168.*.*.

Варіант 10

а) Вивести кількість слів у тексті, що починаються та закінчуються

однаковою літерою.

б) Перевірити, чи містить текст хоча б одне речення, яке закінчується на трикрапку (...).

Варіант 11

а) Дано рядок. Після кожного знаку оклику (!) додати ще один.

б) В тексті, визначити кількість слів, що містять принаймні дві великі літери поспіль.

Варіант 12

а) Замінити всі лапки "..." в цитатах на формат: «...» (українські лапки).

б) В тексті, знайти усі українські номери авто у форматі: AA0000AA та порахувати їх кількість.

Приклад. Написати програму перевірки коректності електронної адреси.

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;
public class EmailValidator {
    private static final Pattern EMAIL_PATTERN =
Pattern.compile(
        "[a-zA-Z0-9._]+@([a-zA-Z]+\\.){1,2}(net|com|org)"
    );
    public static void checkEmail(String email) {
        Matcher matcher = EMAIL_PATTERN.matcher(email);
        if (matcher.matches()) {
            System.out.println("Address \"" + email + "\" is
valid.");
        } else {
            System.out.println("Address \"" + email + "\" is
invalid.");
        }
```

```
    }
    public static void main(String[] args) {
        // Коректні адреси
        checkEmail("john.doe@example.com");
        checkEmail("tester.name@alpha.beta.org");
        // Некоректні адреси
        checkEmail("noatsymbol.com");
        checkEmail("user@domain.co.uk");
        checkEmail("space here@domain.com");
    }
}
```

```
Address "john.doe@example.com" is valid.
Address "tester.name@alpha.beta.org" is valid.
Address "noatsymbol.com" is invalid.
Address "user@domain.co.uk" is invalid.
Address "space here@domain.com" is invalid.
```

Рисунок 2 – Результат компіляції програми

Підготуйте звіт за результатами виконання завдання, який має містити:

- алгоритм розв’язання поставленої задачі з використанням рядкових функцій та регулярних виразів;

- програмний код з коментарями;

- скріншоти результатів виконання програми.

До звіту обов’язково додайте архів з файлами проекту.

Теоретичні питання для самоперевірки

1) Які аргументи підтримуються конструкторами класу String при створенні нових рядкових об’єктів?

2) Які методи класу String можна використати для порівняння рядків, і чим вони різняться?

3) Які функції Java дозволяють здійснювати пошук окремих символів або підрядків у рядкових даних?

4) У яких прикладних задачах доцільно використовувати регулярні вирази для обробки текстової інформації?

5) Чим відрізняється клас String від StringBuffer?

6) Для яких операцій доцільно використовувати метод substring() та які його параметри?

7) Як працює метод toString(), і коли він викликається автоматично?

8) Які базові метасимволи використовуються в регулярних виразах Java і яке їхнє призначення?

Завдання до теми № 4. Класи та об’єкти в Java. Абстрактні класи та інтерфейси

Виконайте практичне завдання відповідно до вашого варіанту.

1 варіант. Розробити класи для обчислення основних геометричних характеристик різних типів трикутників – площі, периметра, радіуса вписаного та описаного кола. Передбачити окремі інтерфейси для кожного типу трикутника (рівностороннього, рівнобедреного, різностороннього, прямокутного) з відповідними методами. Забезпечити реалізацію принципів інтерфейсного програмування та наслідування.

2 варіант. Створити інтерфейс, який містить сигнатури методів для роботи з векторами: обчислення довжини, додавання векторів, множення на скаляр, знаходження скалярного добутку. Реалізувати цей інтерфейс у вигляді окремих класів для роботи з двовимірними та тривимірними векторами. Продемонструвати використання поліморфізму та наслідування.

3 варіант. Розробити інтерфейс, що описує базові операції для роботи з алгебраїчними рівняннями: пошук коренів, визначення кількості розв’язків, вивід результату. Створити окремі класи для розв’язання лінійних, квадратних рівнянь та систем з двома змінними, реалізувавши відповідні методи з інтерфейсу. Забезпечити використання механізмів наслідування для узагальнення спільної функціональності.

Приклад. Розробити інтерфейс Shape, який містить метод для

обчислення площі геометричної фігури. Реалізувати цей інтерфейс у класах, що представляють різні фігури: коло, прямокутник, трикутник. Кожен клас повинен реалізувати власну формулу для знаходження площі. У методі main створити масив фігур і продемонструвати виклик методу обчислення площі для кожного об’єкта.

```
// Інтерфейс геометричної фігури
interface Shape {
    double getArea();
    String getName();
}
// Клас для кола
class Circle implements Shape {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    @Override
    public double getArea() {
        return Math.PI * radius * radius;
    }
    @Override
    public String getName() {
        return "Circle";
    }
}
// Клас для прямокутника
class Rectangle implements Shape {
    private double width;
    private double height;
    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }
```

```
    @Override
    public double getArea() {
        return width * height;
    }
    @Override
    public String getName() {
        return "Rectangle";
    }
}
// Клас для трикутника
class Triangle implements Shape {
    private double base;
    private double height;
    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }
    @Override
    public double getArea() {
        return 0.5 * base * height;
    }
    @Override
    public String getName() {
        return "Triangle";
    }
}

public class ShapeTest {
    public static void main(String[] args) {
        Shape[] shapes = {
            new Circle(5),
            new Rectangle(4, 6),
            new Triangle(3, 4)
        };
```

```
        for (Shape shape : shapes) {
            System.out.println("Area of " + shape.getName() + ":
" + shape.getArea());
        }
    }
}
```

```
Area of Circle: 78.53981633974483
Area of Rectangle: 24.0
Area of Triangle: 6.0
```

Рисунок 3 – Результат компіляції програми

Підготуйте звіт за результатами виконання завдання, який має містити:

- алгоритм розв’язання поставленої задачі з використанням принципів об’єктно-оріентованого програмування;

- програмний код з коментарями;

- скріншоти результатів виконання програми.

До звіту обов’язково додайте архів з файлами проекту.

Теоретичні питання для самоперевірки

1) Що таке клас і об’єкт у Java? Яке між ними співвідношення? Яка роль полів і методів у класі?

2) У чому полягає різниця між динамічним і статичним методом?

3) Чи можна звернутися до нестатичних змінних або методів зсередини статичного методу?

4) Які існують модифікатори доступу в Java? Поясніть їх призначення: public, private, protected, default.

5) Що таке спадкування? Яка його мета? Як оголошується клас, що наслідує інший клас?

6) Чи можна перевизначити метод батьківського класу в підкласі?

7) Що таке абстрактний клас? Для чого він використовується?

8) Чи можна створити об’єкт абстрактного класу? Чому?

9) У чому полягає основна ідея інтерфейсів? Чим інтерфейс відрізняється від абстрактного класу?

10) Як реалізується інтерфейс у класі? Чи може клас реалізовувати кілька інтерфейсів одночасно?

11) Які елементи можуть міститися в інтерфейсі?

12) Що таке поліморфізм? Як він реалізується в Java?

Завдання до теми № 5. Винятки. Обробка винятків

Виконайте практичне завдання відповідно до вашого варіанту.

1) Обчислити вираз: $y = \sqrt{\frac{a \cdot (a-b)}{a}}$.

2) Обчислити вираз: $y = a\sqrt{\frac{a-b}{a+b}}$.

3) Обчислити вираз: $y = \sqrt{\frac{a}{a+b}} + \sqrt{\frac{a-b}{b}}$.

4) Обчислити вираз: $y = \frac{\sqrt{a+b}}{c}$.

5) Обчислити значення логарифмічної функції $y = f(x)$, де $y = \ln(x-1)$.

6) Створити метод swapXY(x, y), який міняє місцями значення x і y. Якщо хоча б один аргумент дорівнює null, викликати NullPointerException з власним описом.

7) Обчислити факторіал числа *n*, яке вводиться з консолі. Обробити ситуації: введення від’ємного числа, введення нечислового значення, переповнення типу int.

Приклад. Обчислемо значення виразу $\sqrt{\frac{a}{b}}$ з обробкою виняткових ситуацій.

```
import java.util.Scanner;
public class SqrtDivision {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try {
            System.out.print("Enter value for a: ");
            double a = sc.nextDouble();
            System.out.print("Enter value for b: ");
            double b = sc.nextDouble();
            if (b == 0) {
                throw new ArithmeticException("Division by zero
is not allowed.");
            }
            double result = a / b;
            if (result < 0) {
                throw new ArithmeticException("Cannot calculate
square root of a negative number.");
            }
            double sqrtResult = Math.sqrt(result);
            System.out.println("Square root of (a / b) is: " +
sqrtResult);
        } catch (ArithmeticException e) {
            System.out.println("Calculation error: " +
e.getMessage());
        } catch (Exception e) {
            System.out.println("An unexpected error occurred: "
+ e.getMessage());
        } finally {
            sc.close();
        }
    }
}
```

```
Enter value for a: 5
Enter value for b: -3
ERROR!
Calculation error: Cannot calculate square root of a negative number.
```

```
Enter value for a: 2
Enter value for b: 0
ERROR!
Calculation error: Division by zero is not allowed.
```

Рисунок 4 – Результат компіляції програми

Підготуйте звіт за результатами виконання завдання, який має містити:

- алгоритм розв’язання поставленої задачі із з обробкою виняткових ситуацій. Забезпечити правильне функціонування програми для будь-яких вхідних даних шляхом обробки винятків.;

- програмний код з коментарями;

- скріншоти результатів виконання програми.

До звіту обов’язково додайте архів з файлами проекту.

Теоретичні питання для самоперевірки

1) Що таке виняткова ситуація (виняток) у Java?

2) Які основні класи ієрархії винятків існують у Java?

3) У чому полягає відмінність між контрольованими (checked) і неконтрольованими (unchecked) винятками?

4) Які команди використовуються для обробки винятків у Java?

5) Для чого використовується блок finally і чи є він обов’язковим?

6) Яка помилка виникає при виконанні виразу

```
int a = 4;
System.out.println(a / 0);
```

і як її обробити?

Завдання до теми № 6. Операції з файлами. Файлові потоки введення-виведення в Java

Виконайте практичне завдання відповідно до вашого варіанту.

1) Дано текстовий файл, що містить список студентів (ПІБ, група, середній бал). Реалізувати програму для:

- зчитування списку з файлу;

- сортування студентів за спаданням середнього балу;

- запису відсортованого списку у новий файл.

2) Дано текстовий файл, що містить список продажів у форматі: назва товару, кількість одиниць, виручка. Реалізувати програму для:

- зчитування інформації з файлу;

- сортування за спаданням виручки;

- збереження результатів у новий файл.

3) Дано текстовий файл, що містить інформацію про книги у форматі: назва книги, автор, рік видання, кількість сторінок. Реалізувати програму для:

- зчитування каталогу з файлу;

- сортування книг за зростанням року видання;

- запису результату у файл.

4) Дано текстовий файл, що містить інформацію про фільми у форматі: назва фільму, жанр, рейтинг (0–10). Реалізувати програму для:

- зчитування списку з файлу;

- сортування фільмів за спаданням рейтингу;

- збереження списку фільмів з рейтингом ≥ 8 у новий файл.

5) Дано текстовий файл, що містить щоденні записи про погоду у форматі: дата, середня температура, кількість опадів. Реалізувати програму для:

- зчитування даних про погоду;

- сортування днів за спаданням температури;

- запису результату в новий файл.

6) Дано текстовий файл, що містить дані про транспортні засоби у форматі: марка, рік випуску, пробіг, тип пального. Реалізувати програму для:

- зчитування з файлу;

- сортування за зростанням пробігу;

- запису у новий файл лише автомобілів з бензиновим двигуном.

7) Дано текстовий файл, що містить інформацію про смартфони у форматі: модель, виробник, ціна. Реалізувати програму для:

- зчитування даних з файлу;

- сортування моделей за спаданням ціни;

- запису відсортованого списку у новий файл.

Розглянемо приклад програми яка читає текст із вхідного файлу (input.txt), виводить його на екран у консоль, і записує в інший файл (output.txt).

```
import java.io.*;
public class FileCopyExample {
    public static void main(String[] args) {
        String inputFile = "input.txt";
        String outputFile = "output.txt";
        try (
            BufferedReader reader = new BufferedReader(new
FileReader(inputFile));
            BufferedWriter writer = new BufferedWriter(new
FileWriter(outputFile))
        ) {
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line); // Вивід у консоль
                writer.write(line); // Запис у файл
                writer.newLine();// Перехід на новий рядок
            }
            System.out.println("File copied successfully.");
        } catch (FileNotFoundException e) {
            System.err.println("Input file not found: " +
inputFile);
```

```
        } catch (IOException e) {
            System.err.println("Error during file processing: "
+ e.getMessage());
        }
    }
}
```

Підготуйте звіт за результатами виконання завдання, який має містити:

- алгоритм розв’язання поставленої задачі роботи з файлами;

- програмний код з коментарями;

- скріншоти результатів виконання програми.

До звіту обов’язково додайте архів з файлами проекту.

Теоретичні питання для самоперевірки

1) Які класи Java відповідають за роботу з файлами на рівні байтових потоків? Наведіть приклади їх використання.

2) У чому полягає відмінність між байтовими (FileInputStream, FileOutputStream) і символьними (FileReader, FileWriter) потоками?

3) Для чого використовується клас File? Які операції можна виконувати за допомогою цього класу?

4) Які методи класу File дозволяють перевірити існування файлу, його розмір, доступність для читання та запису?

5) Наведіть приклад створення нового файлу за допомогою класу File і методу createNewFile().

6) Як правильно закрити потоки після завершення роботи з файлами?

7) Як отримати список файлів у каталозі за допомогою методів класу File?

Завдання до теми № 6. Розробка графічного інтерфейсу в Java

Виконайте практичне завдання відповідно до вашого варіанту.

1) Реалізуйте додаток з GUI для введення та форматування тексту у

вікні програми. Реалізуйте графічне вікно, що містить меню (JMenuBar) з двома розділами: «Шрифт» і «Стиль». У меню «Шрифт» користувач може обрати один із трьох варіантів (JRadioButtonMenuItem): Times New Roman (за замовчуванням), Arial або Verdana. У меню «Стиль» доступні опції: звичайний (за замовчуванням), жирний, курсивний, жирний курсив.

У центральній частині (JTextPane) користувач вводить текст, який одразу відображається відповідно до вибраних налаштувань шрифту та стилю.

2) Реалізуйте додаток для виведення тексту з вибором параметрів у графічному інтерфейсі. Програма має два розділи:

- верхня панель містить поля для введення тексту (JTextField), вибору шрифту (JSpinner: Times New Roman, Arial, Verdana), вибору кольору (JSpinner: чорний, червоний, зелений, синій) та кнопку для підтвердження (JButton);

- нижня панель відображає текст у графічному контексті з використанням drawString() відповідно до обраних параметрів.

3). Реалізуйте додаток-таймер у графічному вікні. Створіть GUI-додаток з головним вікном, яке містить меню з пунктами «Пуск» та «Вихід», кнопку «Пуск», напис «Результат:» і числове поле з початковим значенням 10. При кожному натисканні кнопки значення зменшується на 1. Коли лічильник досягає 1, подальше натискання кнопки викликає діалог із повідомленням про завершення. Пункт меню «Пуск» виконує ту ж дію, що й кнопка.

4) Створіть додаток-калькулятор з полями для введення двох чисел, результату і кнопками для основних арифметичних дій (+, −, ×, ÷), а також операцій квадратного кореня та піднесення до квадрату. У верхньому меню розмістіть пункти, які дублюють кнопки дій. При виникненні помилок (наприклад, поділ на нуль), виводиться діалогове вікно з попередженням.

5) Реалізуйте графічний додаток для форматування тексту за стилем та розміром. Програма дозволяє вводити текст у вікні (JTextPane), а у верхній

панелі вибирати стиль (JComboBox: простий, жирний, курсив) та розмір шрифту (JComboBox: 10pt, 12pt, 14pt). Після вибору параметрів текст автоматично набуває відповідного вигляду. Якщо введений текст містить числа, ця інформація має відображатися окремо (наприклад, через повідомлення або спеціальне поле).

Розглянемо приклад програми на Java, яка створює просте графічне головне меню з використанням JFrame, JMenuBar, JMenu та JMenuItem.

```
package mymenu;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class MainMenu implements ActionListener {
    JLabel label;
    public MainMenu() {
        // Створення головного вікна
        JFrame frame = new JFrame("Main Menu Example");
        frame.setLayout(new FlowLayout());
        frame.setSize(300, 250);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        label = new JLabel("Select a menu item");
        frame.add(label);
        // Створення рядка меню
        JMenuBar menuBar = new JMenuBar();
        // Меню File
        JMenu fileMenu = new JMenu("File");
        JMenuItem open = new JMenuItem("Open");
        JMenuItem save = new JMenuItem("Save");
        JMenuItem exit = new JMenuItem("Exit");
        fileMenu.add(open);
        fileMenu.add(save);
        fileMenu.addSeparator();
        fileMenu.add(exit);
        // Меню Settings
```

```
        JMenu settingsMenu = new JMenu("Settings");
        // Підменю Colors
        JMenu colorMenu = new JMenu("Colors");
        JMenuItem red = new JMenuItem("Red");
        JMenuItem green = new JMenuItem("Green");
        JMenuItem blue = new JMenuItem("Blue");
        colorMenu.add(red);
        colorMenu.add(green);
        colorMenu.add(blue);
        settingsMenu.add(colorMenu);
        // Підменю Priority
        JMenu priorityMenu = new JMenu("Priority");
        JMenuItem high = new JMenuItem("High");
        JMenuItem low = new JMenuItem("Low");
        priorityMenu.add(high);
        priorityMenu.add(low);
        settingsMenu.add(priorityMenu);
        // Пункт Reset
        JMenuItem reset = new JMenuItem("Reset");
        settingsMenu.addSeparator();
        settingsMenu.add(reset);
        // Меню Help
        JMenu helpMenu = new JMenu("Help");
        JMenuItem about = new JMenuItem("About");
        helpMenu.add(about);
        // Додавання всіх меню до рядка меню
        menuBar.add(fileMenu);
        menuBar.add(settingsMenu);
        menuBar.add(helpMenu);
        // Додавання обробників подій
        open.addActionListener(this);
        save.addActionListener(this);
        exit.addActionListener(this);
        red.addActionListener(this);
        green.addActionListener(this);
```

```
        blue.addActionListener(this);
        high.addActionListener(this);
        low.addActionListener(this);
        reset.addActionListener(this);
        about.addActionListener(this);
        frame.setJMenuBar(menuBar);
        frame.setVisible(true);
    }
    // Обробка вибору меню
    @Override
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();
        if (command.equals("Exit")) {
            System.exit(0);
        }
        label.setText(command + " selected");
    }
    public static void main(String[] args) {
        // Запуск GUI у потоці диспетчеризації подій
        SwingUtilities.invokeLater(() -> new MainMenu());
    }
}
```

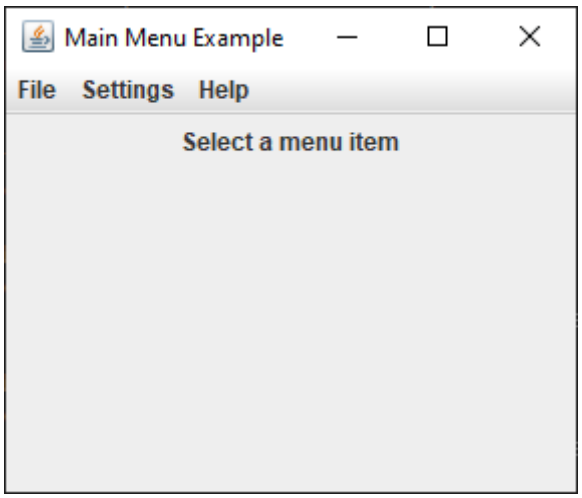

Рисунок 5 – Результат компіляції програми

Підготуйте звіт за результатами виконання завдання, який має містити:

- програмний код з коментарями;

- скріншоти результатів виконання програми.

До звіту обов’язково додайте архів з файлами проекту.

Теоретичні питання для самоперевірки

1) Які існують способи створення діалогових вікон у Java та в чому їх відмінності?

2) Які типи менеджерів компонування підтримуються в Java, і який з них є типовим для JFrame?

3) Що таке JToolBar у бібліотеці Swing, як він створюється та додається до вікна?

4) Яка роль контейнерів (JPanel, JScrollPane, JTabbedPane) у побудові графічного інтерфейсу?

5) Що таке обробник подій (ActionListener, MouseListener) і як він реалізується?

Завдання до теми № 8. Робота із зображеннями в Java

Виконайте практичне завдання відповідно до вашого варіанту.

1) Створити графічне вікно, у якому розміщено еліпс, заповнений певним кольором. При натисканні кнопки «Start» колір еліпса починає змінюватися, а кнопка «Stop» зупиняє цей процес.

2) Створити графічне вікно, у якому розміщено еліпс, що при натисканні кнопки «Start» поступово змінює свій розмір. Зупинка зміни розміру відбувається при досягненні визначених меж.

3) Створити графічне вікно, у якому розміщено прямокутник, що під час натискання кнопки «Start» анімаційно перетворюється у квадрат.

4) Створити графічне вікно, у якому розміщено текст «HelloWorld», який починає рухатись по горизонталі після натискання кнопки «Start», а

кнопка «Stop» зупиняє рух тексту.

5) Створити графічне вікно, у якому за допомогою графічних засобів Java зображено композицію з різноманітних геометричних фігур, кольорів і ліній.

Розглянемо приклад програмbb з графічним інтерфейсом, де в вікні малюється коло, яке змінює свій колір та рухається по діагоналі при натисканні кнопки. Натискання тієї ж кнопки зупиняє анімацію.

```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Ellipse2D;
import javax.swing.*;
public class MovingBallAnimation extends JPanel implements
ActionListener {
    private Timer timer;
    private double x = 10, y = 10;
    private Color color = Color.RED;
    private float hue = 0f;
    public MovingBallAnimation() {
        timer = new Timer(50, this); // Кожні 50 мс оновлення
    }
    public void startAnimation() {
        timer.start();
    }
    public void stopAnimation() {
        timer.stop();
    }
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;
        // Очистка фону
        g2d.setColor(Color.WHITE);
        g2d.fillRect(0, 0, getWidth(), getHeight());
```

```
        // Малюємо коло з поточним кольором
        g2d.setColor(color);
        Ellipse2D ball = new Ellipse2D.Double(x, y, 40, 40);
        g2d.fill(ball);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        // Рух по діагоналі
        x += 5;
        y += 5;
        // Зміна кольору (градієнтна гама)
        hue += 0.01f;
        if (hue > 1f) hue = 0f;
        color = Color.getHSBColor(hue, 1f, 1f);
        // Якщо кулька вийшла за межі панелі, повертаємо назад
        if (x > getWidth() - 40) x = 0;
        if (y > getHeight() - 40) y = 0;
        repaint();
    }
    public static void main(String[] args) {
        JFrame frame = new JFrame("Анімація рухомої кульки");
        MovingBallAnimation      animationPanel      =      new
MovingBallAnimation();
        JButton startStopButton = new JButton("Start");
        startStopButton.addActionListener(e -> {
            if (animationPanel.timer.isRunning()) {
                animationPanel.stopAnimation();
                startStopButton.setText("Start");
            } else {
                animationPanel.startAnimation();
                startStopButton.setText("Stop");
            }
        });
        frame.setLayout(new BorderLayout());
        frame.add(animationPanel, BorderLayout.CENTER);
```

```
        frame.add(startStopButton, BorderLayout.SOUTH);
        frame.setSize(500, 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

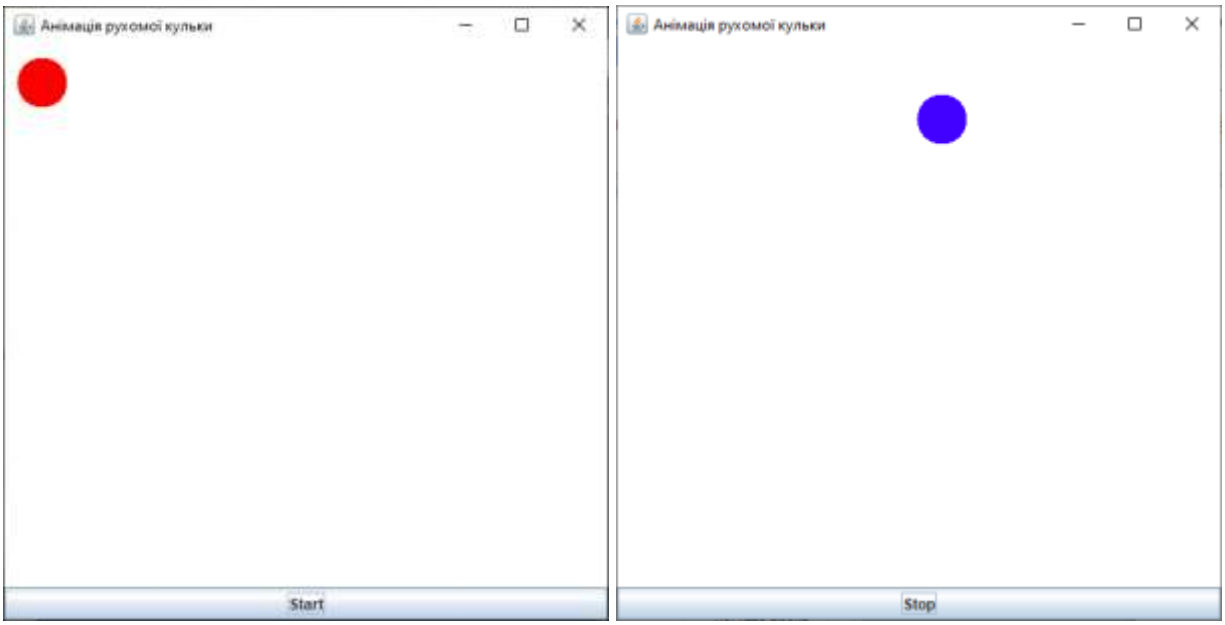

Рисунок 6 – Результат компіляції програми

Підготуйте звіт за результатами виконання завдання, який має містити:

- програмний код з коментарями;

- скріншоти результатів виконання програми.

До звіту обов’язково додайте архів з файлами проекту.

Теоретичні питання для самоперевірки

1) Яке призначення класу Graphics у Java? Які основні методи він надає?

2) Як у Java можна задати колір для малювання? Які існують способи створення об’єкта Color?

3) Як реалізується анімація графічних об’єктів у Java? Яка роль класу Timer у цьому процесі?

4) Яким чином можна змінювати властивості об’єкта (колір, розмір, положення) під час анімації?

5) Як можна реалізувати обробку подій кнопок в графічному інтерфейсі Java?

Завдання до теми № 9. Обробка математичних функцій та побудова графіків функцій в Java

Виконайте практичне завдання відповідно до вашого варіанту.

1) Побудувати графік функції: $y = 150 * \sin(0.2 * x - \pi / 3)$.

2) Побудувати графік функції: $y = 80 * \exp(-0.05 * t) * \sin(5 * t)$.

3) Побудувати графік функції:

$$y = 120 * \exp(-0.01 * t) * \cos(3 * t).$$

4) Побудувати графік функції:

$$y = 50 * \sin(0.4 * x) + 40 * \cos(0.6 * x).$$

5) Побудувати графік функції: $y = 200 * (1 - \exp(-0.03 * t))$.

6) Побудувати графік функції: $y = 100 * \exp(-0.07 * t)$.

7) Побудувати графік функції: $y = 70 * \sinh(0.1 * x)$.

8) Побудувати графік функції: $y = 90 * |\sin(0.3 * x)|$.

9) Побудувати графік функції: $y = 10 * \sin(3 * t) * \exp(-0.02 * t)$.

10) Побудувати графік функції: $y = 30 * \tan(0.1 * x)$.

11) Побудувати графік функції: $y = 100 * \log(t + 1)$.

12) Побудувати графік функції: $y = y = 2 * t^{1.5}$.

13) Побудувати графік функції:

$$y = 10 * \sin(t) + 8 * \sin(3 * t) + 5 * \sin(5 * t).$$

14) Побудувати графік функції: $y = 100 * \cosh(0.05 * x)$.

14) Побудувати графік функції:

$$y = 60 * \exp(-0.04 * t) * \sin(t + \pi / 6).$$

Приклад. Побудуємо графік функції:

$$f(x) = 100 \cdot sin(x \cdot 0.1) + 50 \cdot cos(x \cdot 0.2).$$

```
import javax.swing.*;
import java.awt.*;

public class GraphApp {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Custom Function Graph");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 500);
        frame.add(new GraphPanel());
        frame.setVisible(true);
    }
}

class GraphPanel extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        int width = getWidth();
        int height = getHeight();
        int centerX = width / 2;
        int centerY = height / 2;

        // Малюємо осі X та Y
        g.setColor(Color.GRAY);
        g.drawLine(0, centerY, width, centerY);    // X-axis
        g.drawLine(centerX, 0, centerX, height);   // Y-axis

        // Колір функції
        g.setColor(Color.BLUE);

        // Малюємо графік функції
        for (int i = -centerX + 1; i < centerX - 1; i++) {
```

```
            int x1 = i + centerX;
            int x2 = i + 1 + centerX;

            double y1Val = 100 * Math.sin(i * 0.1) + 50 *
Math.cos(i * 0.2);
            double y2Val = 100 * Math.sin((i + 1) * 0.1) + 50 *
Math.cos((i + 1) * 0.2);

            int y1 = centerY - (int) y1Val;
            int y2 = centerY - (int) y2Val;

            g.drawLine(x1, y1, x2, y2);
        }

        // Підпис
        g.setColor(Color.BLACK);
        g.drawString("f(x) = 100 * sin(x * 0.1) + 50 * cos(x *
0.2)", 10, 20);
    }
}
```

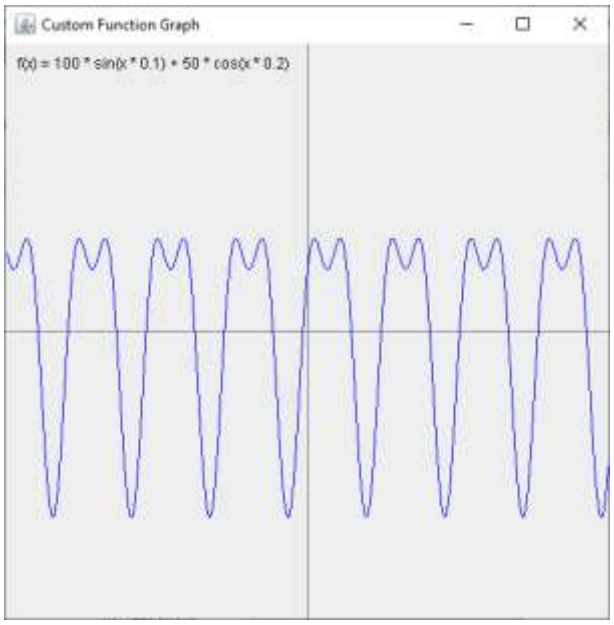

Рисунок 7 – Результат компіляції програми

Підготуйте звіт за результатами виконання завдання, який має містити:

- програмний код з коментарями;

- скріншоти результатів виконання програми.

До звіту обов'язково додайте архів з файлами проекту.

Теоретичні питання для самоперевірки

1) Який клас Java використовується для обчислення математичних функцій (тригонометричних, логарифмічних тощо)?

2) Який пакет Java використовується для роботи з графікою?

3) Для чого потрібні менеджери розташування компонентів?

4) Який клас або метод дозволяє малювати лінії, точки, текст?

5) У чому різниця між використанням Applet, JFrame, JPanel для малювання?

6) Як забезпечити оновлення графіка при зміні параметрів?

7) Як обробити події (наприклад, натискання кнопки) для повторного малювання?

Завдання до теми № 9. Реалізація багатопотоковості в Java

Виконайте практичне завдання відповідно до вашого варіанту.

1) Створіть два потоки. Один потік записує у файл випадкові дробові числа (типу double), другий – читає ці значення, обчислює їх середнє арифметичне та виводить його на екран.

2) Реалізуйте додаток, де окремий потік обчислює значення функції:

$$Y = e^{(-x) * \cos(x) + x^2}$$

у циклі з кроком 0.1 в межах [0, 10] і безперервно виводить його у консоль.

3). Створіть три потоки:

- перший генерує факторіали чисел (n!),

- другий – квадрати натуральних чисел,

- третій – куби.

Усі результати записуються в різні файли. Після зупинки – виведіть

статистику: найбільше значення, середнє, кількість елементів у кожному файлі.

4) Імітація гри «Лото»: п’ять потоків генерують випадкові числа в діапазоні [1–99]. Після зупинки кожного – результати перевіряються на комбінації: всі парні, всі непарні, всі кратні 7, наявність «щасливого» числа (наприклад, 77).

5) Реалізуйте багатопотокову оцінку значення нескінченного ряду:

$$S = 1 - 1/2 + 1/3 - 1/4 + 1/5 - \ldots$$

Один потік обчислює суму непарних членів ряду, другий – парних. Після певної кількості кроків – об'єднати результати і вивести суму.

6) Створіть два потоки:

- перший обчислює добуток елементів заданої матриці,

- другий – мінімальний елемент.

Запишіть результати у файли, після завершення обчислень – виведіть файли і порівняйте обчислення.

7) Реалізуйте гральний автомат з 4 потоками: кожен генерує випадковий символ з набору {'A', 'B', 'C', 'D', '7'}. Виведіть комбінації (4 однакових, 3 однакових, хоча б один «7», комбінація «ABCD», дві пари однакових символів).

8). Створіть візуальну модель «змагання квадратів», де 4 квадрати рухаються вертикально (по осі Y) незалежно. Переможець – той, хто першим дістанеться нижнього краю панелі. Кожен потік повинен мати власну швидкість, змінювану випадково.

Приклад. Реалізуйте програму на Java, в якій два незалежні потоки виконують обчислення математичних послідовностей:

- перший потік обчислює та виводить наближене значення числа π за формулою Лейбніца: $\pi = 4 \cdot (1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \cdots)$. Потік має постійно оновлювати результат кожні 100 ітерацій (виводити у консоль поточне значення π);

- другий потік обчислює та виводить n перших чисел Каталана, використовуючи рекурсивну формулу: $C_n = \frac{(2n)!}{(n+1)!n!}$ або рекурсивно: $C_0 = 1, \quad C_{n+1} = \sum_{i=0}^{n} C_i \cdot C_{n-i}.$

```
import java.io.FileWriter;
import java.io.IOException;

public class MultiThreadPiCatalan {

    public static void main(String[] args) {
        int catalanCount = 10;
        int piIterations = 1_000_000;

        Thread piThread = new Thread(new
PiCalculator(piIterations));
        Thread catalanThread = new Thread(new
CatalanCalculator(catalanCount));

        piThread.start();
        catalanThread.start();

        try {
            piThread.join();
            catalanThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("\n--- Computation completed ---");
    }

    static class PiCalculator implements Runnable {
```

```
        private final int iterations;

        public PiCalculator(int iterations) {
            this.iterations = iterations;
        }

        @Override
        public void run() {
            double pi = 0.0;
            boolean positive = true;

            try (FileWriter writer = new FileWriter("pi.txt")) {
                for (int i = 0; i < iterations; i++) {
                    double term = 4.0 / (2 * i + 1);
                    pi += positive ? term : -term;
                    positive = !positive;

                    if (i % 100_000 == 0) {
                        System.out.printf("pi (iteration %d):
%.10f\n", i, pi);
                    }
                }

                writer.write(String.valueOf(pi));
                System.out.printf("pi (final value): %.10f\n",
pi);
                System.out.printf("Math.PI        : %.10f\n",
Math.PI);
                System.out.printf("Error          : %.10f\n",
Math.abs(Math.PI - pi));

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
```

```
    }

    static class CatalanCalculator implements Runnable {
        private final int n;

        public CatalanCalculator(int n) {
            this.n = n;
        }

        @Override
        public void run() {
            long[] catalans = new long[n];
            catalans[0] = 1;

            for (int i = 1; i < n; i++) {
                catalans[i] = 0;
                for (int j = 0; j < i; j++) {
                    catalans[i] += catalans[j] * catalans[i - 1
- j];
                }
            }

            try (FileWriter writer = new
FileWriter("catalan.txt")) {
                long sum = 0;
                System.out.println("\nCatalan numbers:");
                for (int i = 0; i < n; i++) {
                    System.out.printf("C_%d = %d\n", i,
catalans[i]);
                    writer.write("C_" + i + " = " + catalans[i]
+ "\n");
                    sum += catalans[i];
                    Thread.sleep(500); // simulate delay
                }
                System.out.println("Sum of the first " + n + "
```

```
Catalan numbers: " + sum);
            } catch (IOException | InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

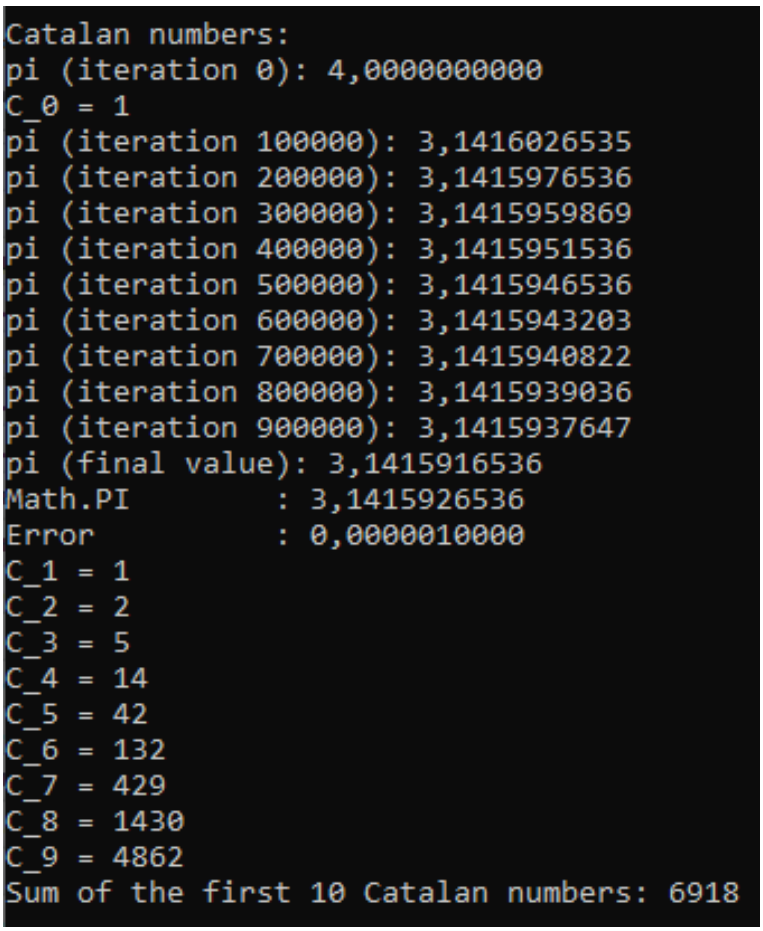

Рисунок 8 – Результат компіляції програми

Підготуйте звіт за результатами виконання завдання, який має містити:

- алгоритм розв’язання поставленої задачі;
- програмний код з коментарями;
- скріншоти результатів виконання програми.

До звіту обов’язково додайте архів з файлами проекту.

Теоретичні питання для самоперевірки

1) Які способи створення потоків передбачені в Java?

2) Як створити потік шляхом реалізації інтерфейсу Runnable?

3) Як створити потік через наслідування від класу Thread?

4) Яку роль виконує метод start() у класі Thread?

5) Чим відрізняється метод run() від методу start()?

6) Для чого використовується метод sleep()?

7) Яке призначення методу join() у потоках?

8) Які проблеми можуть виникати при доступі кількох потоків до спільного ресурсу?

Завдання до теми № 9. Класи-колекції

Виконайте практичне завдання відповідно до вашого варіанту.

Варіант 1. Прочитайте рядки з текстового файлу. Запишіть кожен рядок у стек (Stack<String>). Виведіть рядки у зворотному порядку у новий файл.

Варіант 2. Введіть довільне ціле число. Занесіть кожну цифру цього числа в стек (Stack<Integer>). Сформуйте нове число, у якого цифри йдуть у зворотному порядку.

Варіант 3. Нехай кожен багаточлен заданий як HashMap<Integer, Integer> – де ключ – ступінь, значення – коефіцієнт. Реалізуйте метод для додавання двох багаточленів. Виведіть результат у вигляді звичайного математичного виразу.

Варіант 4. Згенеруйте масив з 20 цілих чисел за допомогою Random(). Знайдіть усі унікальні елементи (які не повторюються) за допомогою Set, Map, або Collections.frequency.

Варіант 5. Згенеруйте масив цілих чисел. Створіть нову колекцію, яка містить лише парні числа з першої колекції.

Варіант 6. Введіть рядок з консолі. Використовуючи власноруч створений клас Stack, виведіть цей рядок у зворотному порядку символів.

Варіант 7. Представте два багаточлени у вигляді списків коефіцієнтів

(List<Integer>). Реалізуйте алгоритм множення багаточленів. Виведіть результат.

Варіант 8. Згенеруйте масив з випадкових цілих чисел (у тому числі й від'ємних). Створіть нову колекцію, яка містить лише додатні числа.

Варіант 9. Реалізуйте власний клас MySet на основі Set<Integer>. Реалізуйте метод, який визначає перетин двох множин. Продемонструйте роботу на прикладі.

Варіант 10. Реалізуйте власний клас MySet на основі Set<Integer>. Реалізуйте метод, який виконує об'єднання двох множин. Виведіть результат.

Приклад. Реалізуйте програму на Java, в якій необхідно:

- зчитати список імен;

-зберегти їх у ArrayList.

- перетворити на HashSet, щоб прибрати дублікати;

- зберегти кожне ім'я в Stack у зворотньому порядку;

- зберегти у HashMap поєднання "Ім'я – номер за порядком";

- вивести дані за допомогою Iterator.

import java.util.*;

```
public class CollectionExample {
    public static void main(String[] args) {
        // Крок 1: Створення списку імен
        List<String> namesList = new ArrayList<>();
        namesList.add("Alice");
        namesList.add("Bob");
        namesList.add("Charlie");
        namesList.add("Alice"); // дублікат
        namesList.add("Diana");

        System.out.println("Initial List:");
        System.out.println(namesList);
```

```
        // Крок 2: Видалення дублікатів через HashSet
        Set<String> uniqueNames = new HashSet<>(namesList);
        System.out.println("\nUnique Names (Set):");
        System.out.println(uniqueNames);

        // Крок 3: Збереження у Stack для виводу у зворотному
порядку
        Stack<String> nameStack = new Stack<>();
        for (String name : uniqueNames) {
            nameStack.push(name);
        }

        System.out.println("\nNames in Reverse Order (Stack):");
        while (!nameStack.empty()) {
            System.out.println(nameStack.pop());
        }

        // Крок 4: Збереження у HashMap (Ім'я -> Порядковий
номер)
        Map<String, Integer> nameMap = new HashMap<>();
        int index = 1;
        for (String name : uniqueNames) {
            nameMap.put(name, index++);
        }

        System.out.println("\nNames with Indexes (HashMap):");
        for (Map.Entry<String, Integer> entry :
nameMap.entrySet()) {
            System.out.println(entry.getKey() + " -> " +
entry.getValue());
        }

        // Крок 5: Обхід ArrayList за допомогою ітератора
        System.out.println("\nIterating over List using
Iterator:");
```

```
        Iterator<String> iterator = namesList.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

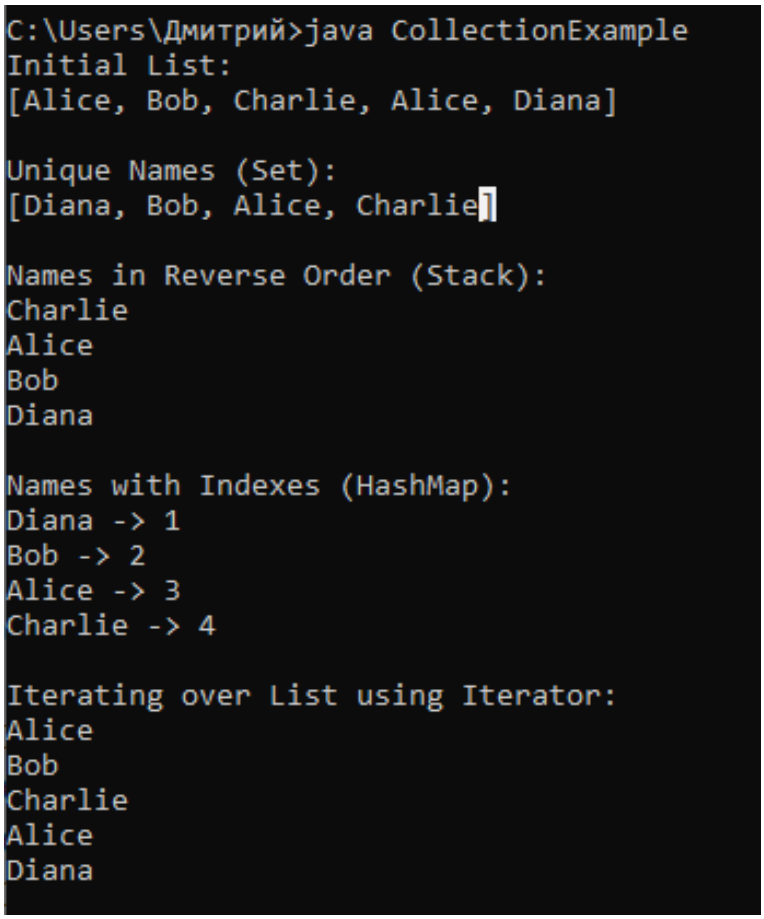

Рисунок 9 – Результат компіляції програми

Підготуйте звіт за результатами виконання завдання, який має містити:

- алгоритм розв’язання поставленої задачі;
- програмний код з коментарями;
- скріншоти результатів виконання програми.

До звіту обов’язково додайте архів з файлами проекту.

Теоретичні питання для самоперевірки

1) Що таке колекції в Java? Яка їх основна мета?

2) Чим відрізняються інтерфейси List, Set і Map? Наведіть приклади реалізацій.

3) Що таке ArrayList, і в чому його відмінність від LinkedList?

4) Які особливості HashSet? Як він зберігає елементи?

5) У чому полягає різниця між HashMap і TreeMap?

6) Для чого використовується клас Stack? Як реалізується принцип LIFO у ньому?

7) Яке призначення класу Collections? Назвіть основні його методи.

8) Що таке дженерики (generics) у колекціях і навіщо вони потрібні?

9) Як видалити всі дублікати зі списку?

10) Які є способи обходу елементів у колекціях?

4. Методичні рекомендації до самостійної роботи

Самостійна робота здобувачів виконується відповідно до тем, визначених у робочій програмі дисципліни «Спеціалізовані мови програмування». Завдання для самостійного опрацювання наведено в таблиці 14.

Таблиця 3

Завдання для самостійної робота

№ з/п	Види, зміст самостійної роботи	Кількість годин
1	Тема 1. Огляд технологій JAVA і IDE. Самостійне опрацювання лекційного матеріалу. Контроль: опитування під час практичного заняття, опитування під час прийому практичних робіт.	7/8*
2	Тема 2. Основи синтаксису Java. Самостійне опрацювання лекційного матеріалу. Контроль: опитування під час практичного заняття, опитування під час прийому практичних робіт.	7/8*
3	Тема 3. Класи в Java. Самостійне опрацювання лекційного матеріалу. Контроль: опитування під час	7/8*

	практичного заняття, опитування під час прийому практичних робіт.	
4	Тема 4. Класи стандартної бібліотеки. Самостійне опрацювання лекційного матеріалу. Контроль: опитування під час практичного заняття, опитування під час прийому практичних робіт.	7/8*
5	Тема 5. Робота з файловою системою. Самостійне опрацювання лекційного матеріалу. Контроль: опитування під час практичного заняття, опитування під час прийому практичних робіт.	7/8*
6	Тема 6. Створення GUI. Бібліотека Swing. Самостійне опрацювання лекційного матеріалу. Контроль: опитування під час практичного заняття, опитування під час прийому практичних робіт.	8/4*
7	Тема 7. Реалізація багатопотоковості в Java. Самостійне опрацювання лекційного матеріалу. Контроль: опитування під час практичного заняття, опитування під час прийому практичних робіт.	4/3*
Разом		56/66*

* у разі формування малочисельних груп обсяг аудиторного навчального навантаження відведеного на вивчення навчальної дисципліни зменшується відповідно до Положення про планування й звітування науково-педагогічних працівників Харківського національного університету імені В. Н. Каразіна.

Індивідуальні завдання

В рамках індивідуального завдання здобувач має виконати контрольну роботу. Контрольна робота у вигляді тестових завдань на платформі Moodle (максимальна кількість балів 12). Тест складається з 24 питань, кожне питання оцінюється по 0,5 балів.

Теми для підготовки до контрольної роботи

1) Основи мови програмування Java. Платформи Java та огляд середовищ розробки (IDE).

2) Базовий синтаксис Java: ключові слова, літерали, оператори керування, змінні, типи даних.

3) Масиви в Java. Ініціалізація, обробка, багатовимірні масиви.

4) Опрацювання рядків у Java. Використання класу String та регулярних виразів.

5) Основи об'єктно-орієнтованого програмування в Java: класи, об'єкти, інкапсуляція, наслідування, поліморфізм.

6) Абстрактні класи та інтерфейси. Використання інтерфейсів для порівняння об’єктів.

7) Винятки в Java. Механізми обробки винятків: try, catch, finally, створення власних винятків.

8) Огляд стандартних бібліотек Java. Пакет java.lang, класи-утиліти, обробка часу та дат.

9) Колекції в Java. Списки, множини, мапи. Інтерфейси List, Set, Map.

10) Робота з файлами у Java. Потоки введення-виведення, серіалізація об'єктів.

11) Основи розробки графічного інтерфейсу користувача в Java за допомогою бібліотеки Swing.

12) Події в Swing, компоненти управління, створення меню та діалогових вікон.

13) Обробка зображень у Java: зчитування, відображення, збереження.

14) Математичні функції в Java. Побудова графіків функцій.

15) Багатопотоковість у Java. Клас Thread, інтерфейс Runnable, синхронізація потоків.

5. Теми для підготовки до заліку

Залікова робота передбачає виконання тестових завдань на платформі Moodle (максимальна кількість балів 40). Тест складається з 40 питань, кожне питання оцінюється по 1 балу.

Теми для підготовки до заліку

1) Архітектура та основні платформи Java (Java SE, EE, ME). Призначення віртуальної машини Java (JVM).

2) Огляд середовищ розробки Java (Eclipse, IntelliJ IDEA, NetBeans). Створення і компіляція проєкту.

3) Синтаксис мови Java: структура програми, правила оголошення класів, методів, змінних.

4) Примітивні типи даних у Java. Автоматичне приведення типів та явне приведення.

5) Ключові слова, оператори, літерали. Оператори керування (if, switch, for, while, do-while, break, continue).

6) Масиви в Java: одновимірні, багатовимірні, динамічні масиви.

7) Обробка рядків: класи String, StringBuilder, StringBuffer. Основні методи. Регулярні вирази (Pattern, Matcher).

8) Основні принципи ООП: інкапсуляція, наслідування, поліморфізм.

9) Створення та використання класів та об’єктів. Модифікатори доступу.

10) Конструктори, статичні члени класу, перевантаження методів.

11) Абстрактні класи та інтерфейси. Переваги інтерфейсного програмування.

12) Клас Object і його методи (toString, equals, hashCode). Клас Class і механізм рефлексії.

13) Внутрішні класи (вкладені, анонімні, локальні).

14) Основні пакети стандартної бібліотеки: java.lang, java.util, java.io, java.nio.

15) Обробка винятків у Java: типи винятків, конструкції try, catch, finally, створення власних винятків.

16) Класи-колекції в Java: інтерфейси List, Set, Map; реалізації ArrayList, HashSet, HashMap.

17) Ітератори та циклічний доступ до колекцій (Iterator, for-each, forEach).

18) Робота з файлами: потоки InputStream, OutputStream, Reader, Writer. Буферизовані потоки.

19) Серіалізація об’єктів: інтерфейс Serializable, класи ObjectInputStream, ObjectOutputStream.

20) Основи побудови графічного інтерфейсу з використанням бібліотеки Swing.

21) Компоненти інтерфейсу: JFrame, JPanel, JLabel, JButton, JTextField, JCheckBox, JComboBox.

22) Обробка подій у Swing. Реалізація слухачів подій.

23) Створення меню та стандартних діалогових вікон (JMenu, JDialog, JOptionPane).

24) Робота з графікою в Java. Малювання фігур, обробка зображень.

25) Побудова математичних графіків у Java: бібліотеки для візуалізації, створення кастомних компонентів.

26) Основи багатопотокового програмування. Порівняння процесів і потоків.

27) Клас Thread і інтерфейс Runnable. Методи start(), run(), sleep(), join(), interrupt().

28) Синхронізація потоків: ключове слово synchronized, блокування, проблеми гонок.

6. Система оцінювання навчальної діяльності здобувача

Оцінювання результатів навчання здобувачів у межах дисципліни «Спеціалізовані мови програмування» здійснюється шляхом:

– прийому та оцінювання звітів з виконання практичних завдань (максимальна кількість балів 48 впродовж семестру);

– виконання контрольної роботи у вигляді тестових завдань на платформі Moodle (максимальна кількість балів 12).

– виконання залікової роботи у вигляді тестових завдань на платформі Moodle (максимальна кількість балів 40).

Система оцінювання за дисципліною відображена у таблиці 16.

Таблиця 16

Схема нарахування балів

	Поточний контроль, самостійна робота, індивідуальні завдання									Залікова робота	Сума
	Розділ 1					Розділ 2		Контрольна робота	Разом		
	Т1	Т2	Т3	Т4	Т5	Т1	Т2				
max	4	4	8	4	4	20	4	12	60	40	100
min	2	2	4	2	2	10	2	6	30	20	50

Т1, Т2 ... – теми розділів.

Додаткові бали можна отримати за наукову роботу в межах навчальної дисципліни:

- написання та публікація тез конференції – 5 балів;

- написання та публікація статті – 10 балів;

- участь в олімпіаді / конкурсі / хакатоні – 15 балів.

Для допуску до складання підсумкового контролю здобувач повинен набрати не менше 25 балів з навчальної дисципліни під час поточного контролю, самостійної роботи.

Критерії оцінювання навчальних досягнень

1. Критерієм успішного проходження здобувачем освіти оцінювання може бути досягнення ним мінімальних порогових рівнів оцінок за кожним запланованим результатом навчання навчальної дисципліни.

2. Мінімальний пороговий рівень оцінки варто визначати за допомогою якісних критеріїв і трансформувати його в мінімальну позитивну оцінку числової (рейтингової) шкали, що використовується.

Критерії оцінювання практичних робіт

Перед виконанням студент вивчає вимоги завдання, відбувається обговорення, щоб завдання було зрозуміле коректно. Дозволяється сумісне обговорення, але програмний код кожен студент пише власноруч. Якщо програмний код робіт різних студентів має суттєву ступінь схожості, то ці роботи дискваліфікуються та не оцінюються.

Захист практичної роботи складається з двох етапів.

Перший етап – представлення результатів: готового програмного коду у відповідності до завдання роботи.

Критерії оцінювання результатів роботи:

• робота була виконана у відповідності з технічним завданням за вказаний час – студент отримує 100% від максимальної кількості балів;

• робота була виконана у відповідності з технічним завданням із запізненням – студент отримує 50 % від максимальної кількості балів;

• технічне завдання виконано не повністю, а на х % – студент отримує х% від балів, що мали б бути зараховані у відповідності до строків виконання.

Другий етап здачі практичної роботи – відповідь на контрольні питання. Кількість питань визначає викладач за результатами представлених результатів, але не менше ніж 2.

Кожне контрольне питання оцінюється таким чином:

• повна розгорнута відповідь з прикладами та додатковим завданням, що було опрацьовано на самостійній роботі – кількість балів, що отримані на першому етапі збільшується до 25 % балів;

• повна, але не розгорнута відповідь без додаткового завдання – кількість балів не змінюється;

• неповна відповідь, або відповідь, що містить незначні та некритичні помилки чи суперечності – кількість балів, що було отримано на попередньому етапі зменшується на 25%;

• відповідь, що містить критичні помилки, або відсутність відповіді – кількість балів, що було отримано на попередньому етапі зменшується на 50 %.

Критерії оцінювання контрольних робіт студентів

Контрольна робота передбачає виконання 24 тестових завдань на платформі Moodle (максимальна кількість балів 12) Кожне питання оцінюється по 0,5 балів.

Критерії оцінювання залікових робіт студентів

Вимоги	Кількість балів
Показані всебічні систематичні знання та розуміння навчального матеріалу; безпомилково виконані завдання.	35-40
Показані повні знання навчального матеріалу; помилки, якщо вони є, не носять принципового характеру.	30-35
Показано повне знання необхідного навчального матеріалу, але допущені помилки.	20-30
Показано повне знання необхідного навчального матеріалу, але допущені суттєві помилки	10-20
Показано недосконале знання навчального матеріалу, допущені суттєві помилки.	5-10
Показано недосконале знання навчального матеріалу, допущені суттєві помилки, які носять принциповий характер; обсяг знань не дозволяє зсвоїти предмет.	1-5

Шкала оцінювання

Сума балів за всі види навчальної діяльності протягом семестру	Оцінка	
	для чотирирівневої шкали оцінювання	для дворівневої шкали оцінювання
90 – 100	відмінно	зараховано
70-89	добре	
50-69	задовільно	
0-49	незадовільно	не зараховано

7. Рекомендована література

1. Татарчук Д. Д., Діденко Ю. В., Свєчніков Г. С. Об’єктно-орієнтоване програмування мовою Java : навч. посіб. [Електронний ресурс] для здобувачів ступеня бакалавра за освіт. програмою «Мікро- та наноелектроніка» спец. 176 «Мікро- та наносистемна техніка» / КПІ ім. Ігоря Сікорського. – Київ : КПІ ім. Ігоря Сікорського, 2024. – 153 с.

2. Галкін О. В., Катеринич Л. О., Шкільняк О. С. Програмування на Java 8 : навч. посібник для студентів факультету комп’ютерних наук та кібернетики. – Київ : ЛОГОС, 2017. – 186 с.

3. Копитко М. Ф., Іванків К. С. Основи програмування мовою Java : тексти лекцій. – Львів : Видавничий центр ЛНУ ім. Івана Франка, 2002. – 83 с.

4. Васильєв А. Н. Програмування мовою Java. – Київ : Bohdan Books, 2022. – 699 с. : іл. – ISBN 966-10-5879-2, ISBN 978-966-10-5879-7.

5. Горбань А. В. Програмування в Java : навч. посібник [Електронний ресурс]. – 2008. – 310 с. – Режим доступу: http://programming.in.ua/programming/basisprogramming/144-programmingJava-book.html

6. Тарнавський Ю. А. Java-програмування : комп’ютерний практикум [Електронний ресурс] : навч. посіб. для студ. спец. 122 «Комп’ютерні науки», освіт.-проф. програми «Комп’ютерний моніторинг та геометричне моделювання процесів і систем» / КПІ ім. Ігоря Сікорського. – Київ : КПІ ім. Ігоря Сікорського, 2021. – 95 с.

7. Брнакевич І. Є., Вагін П. П. Програмування мовою Java : використання фундаментальних класів : тексти лекцій. – Львів : Видавничий центр ЛНУ ім. Івана Франка, 2002. – 75 с. – Режим доступу: http://blues.franko.lviv.ua/ami/books/ami/Java_fundamental.pdf

8. Ратушняк Т. В. Програмування мовою JAVA : практикум : навч. посібник / Державна фіскальна служба України, Ун-т держ. фіскальної служби України. – Ірпінь, 2017. – 212 с.

9. Кадомський К. К., Ніколюк П. К. Java. Теорія і практика : навч. посібник для студентів природничих спеціальностей університетів. – Вінниця : ДонНУ, 2019. – 197 с.

10. Ткаченко К. О., Ткаченко О. А., Ткаченко О. І. Програмування мовою Java. Конспект лекцій [Електрон. ресурс] : навч. посіб. для студ. спец. 121 “Інженерія програмного забезпечення” / К. О. Ткаченко, О. А. Ткаченко, О. І. Ткаченко; КПІ ім. Ігоря Сікорського. – Київ : КПІ ім. Ігоря Сікорського, 2024. – 280 с.

11. Бандура В. В., Храбатин Р. І. Об’єктні технології Java : конспект лекцій / В. В. Бандура, Р. І. Храбатин. – Івано-Франківськ : ІФНТУНГ, 2023. – 194 с.

12. Бандура В. В., Чернишов М. Ю. Об’єктні технології Java : лабораторний практикум / В. В. Бандура, М. Ю. Чернишов. – Івано-Франківськ : ІФНТУНГ, 2023. – 188 с.

13. Каплун В. А., Снігур А. В., Лукічов В. В. Програмування мовою Java. Теорія і практика [електрон. посіб.] / В. А. Каплун, А. В. Снігур, В. В. Лукічов. – Вінниця : ВНТУ, 2023.

14. Шульга В., Замрій І., Шахматов І. Від основ до вебпрограмування на Java. Частина 1. Базові концепції програмування на Java / В. Шульга, І. Замрій, І. Шахматов; Державний ун-т інформ.-комунікац. технологій. – Київ : ДУІКТ, 2025.

15. Тарнавський Ю. А. Java-програмування: комп’ютерний практикум [Електрон. ресурс] : навч. посіб. для студ. спец. 122 «Комп’ютерні науки» / Ю. А. Тарнавський; КПІ ім. Ігоря Сікорського. – Київ : КПІ ім. Ігоря Сікорського, 2021. – 95 с.

16. Schildt H. Java: A Beginner's Guide. 8th ed. – New York : McGraw-Hill Education, 2018. – 684 p.

17. Javaland – програмування на Java [Електронний ресурс]. – Режим доступу: http://Javaland.com.ua

18. Java Tutorial [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/Java/

Електронне навчальне видання комбінованого використання
Можна використовувати в локальному та мережному режимі

Ковальчук Дмитро Миколайович

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

Методичні рекомендації до практичних занять
і самостійної роботи з дисципліни для здобувачів вищої освіти
першого (бакалаврського) рівня галузі знань 12 «Інформаційні технології» за
спеціальністю 122 «Комп'ютерні науки» за освітньо-професійною програмою
«Комп'ютерні науки та інформаційні технології в бізнесі»

В авторській редакції

Підписано до розміщення 24.04.2026. Гарнітура Times New Roman.
Ум. друк. арк. 3,19. Обсяг 0,884 Мб. Зам. № 155/26.

Харківський національний університет імені В. Н. Каразіна,
61022, м. Харків, майдан Свободи, 4.
Свідоцтво суб'єкта видавничої справи ДК № 3367 від 13.01.2009
Видавництво ХНУ імені В. Н. Каразіна