Полтавський університет економіки і торгівлі
Навчально-науковий інститут денної освіти (заочно-дистанційного навчання)
Форма навчання денна (заочна)
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту

Завідувач кафедри

_____Олена ОЛЬХОВСЬКА
(підпис)

«___»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

РОЗРОБКА НАВЧАЛЬНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА ТЕМИ «COLLECTIONS. HASHSET» ДИСЦИПЛІНИ «ПРОГРАМУВАННЯ»

зі спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерні науки»
ступеня бакалавр

Виконавець роботи **Пазечко Ярослав Олександрович**

_____«__»_____2025 р.
(підпис)

Науковий керівник к.ф.-м.н.,доц., _____

_____«__»_____2025 р.
(підпис)

Рецензент

ПОЛТАВА 2025

РЕФЕРАТ

Дипломна робота містить 71 сторінку, 11 рисунків, 2 таблиці, список літератури з 70 найменувань.

Розробка навчального програмного забезпечення за теми «Collections. HashSet» дисципліни «Програмування»

Дослідження зосереджене на підвищенні ефективності викладання теми колекцій у мові Java, з акцентом на структуру HashSet. Основним завданням є створення навчального засобу, що систематизує теоретичний матеріал і забезпечує зручний інтерактивний доступ до нього в освітньому середовищі.

Об’єктом дослідження є процес вивчення структур даних у курсі програмування, зокрема теми використання колекцій типу HashSet у мові Java, а також можливості покращення цього процесу за рахунок застосування спеціалізованого програмного забезпечення. У центрі уваги — ефективність засвоєння теоретичних знань та практичних навичок шляхом інтерактивної взаємодії з навчальними матеріалами.

Метою дипломної роботи є створення електронного навчального посібника, який дозволяє студентам у зручній формі опрацьовувати теми, пов’язані з Java Collections Framework, а саме HashSet, та закріплювати отримані знання через приклади, візуалізації та функціональні можливості інтерфейсу. Розроблене програмне забезпечення надає доступ до навчального контенту у форматі HTML, забезпечує перегляд PDF-документів, підтримує масштабування, пошук інформації, друк вибраних сторінок та зручну навігацію за допомогою дерева тем.

Результат роботи може використовуватись у навчальному процесі як допоміжний інструмент для викладачів та як самостійне джерело для студентів, які опановують основи колекцій у Java. Програма побудована з використанням бібліотек Java Swing та PDFBox, що дозволяє забезпечити сумісність із різними платформами та простоту використання. Такий формат подачі матеріалу робить навчання більш доступним, наочним і продуктивним, особливо у темах, де важливо зрозуміти логіку структури даних, їх поведінку та особливості реалізації.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

OCPJP – Oracle Certified Professional Java Programmer.

ПЗ – Програмне забезпечення.

ВНЗ – Вищий навчальний заклад.

ID – унікальний ідентифікатор для ідентифікації об’єкта.

JCF – Java Collections Framework.

JDK – Java Development Kit, набір інструментів для розробки програм.

ОС – Операційна система.

ООП – Об'єктно-орієнтоване програмування.

IDE – Інтегроване середовище розробки.

GUI – Graphical user interface.

JVM – Java Virtual Machine.

PDF – Portable Document Format.

HTML – HyperText Markup Language.

ВСТУП

Актуальність теми. У сучасному освітньому просторі, де цифрові технології відіграють ключову роль, особливої ваги набувають інструменти, що забезпечують ефективне вивчення програмування. Тема колекцій у Java, зокрема структура HashSet, є важливою складовою базової алгоритмічної підготовки студентів. Вона потребує не лише розуміння теоретичних засад, а й засвоєння практичних принципів використання колекцій у типових задачах. Проте саме абстрактність цих понять часто ускладнює їхнє сприйняття, особливо на етапі початкового навчання.

У відповідь на це постає потреба у створенні спеціалізованого навчального програмного забезпечення, що поєднує в собі наочність, доступність і функціональність. Використання електронних посібників із підтримкою навігації, масштабування, друку й пошуку дозволяє зробити вивчення HashSet більш зрозумілим і наближеним до реальної практики програмування. Крім того, інтеграція таких матеріалів у графічний інтерфейс надає можливість викладачеві організовувати навчальний процес більш гнучко, а студентові — вивчати теми самостійно, у зручному темпі.

Об’єкт дослідження — навчальне програмне забезпечення, призначене для підтримки вивчення структури HashSet у межах дисципліни «Програмування».

Предмет дослідження — процес створення електронного посібника, який забезпечує структуровану подачу матеріалів, інтерактивну взаємодію з навчальним контентом та реалізацію прикладних функцій за допомогою мови програмування Java.

Мета та завдання роботи. Метою дипломної роботи є розробка навчального програмного забезпечення, що охоплює тематику колекцій у Java, з акцентом на структуру HashSet, і слугує засобом підтримки навчального процесу в закладах вищої освіти. Для реалізації поставленої мети необхідно вирішити низку завдань:

- здійснити аналіз існуючих електронних засобів навчання, що стосуються тематики Java Collections;
- обґрунтувати вибір інструментів для реалізації програмного забезпечення, зокрема бібліотек Swing та PDFBox;
- розробити архітектуру додатку, що забезпечує зручну навігацію, пошук, перегляд і друк навчального матеріалу;
- створити графічний інтерфейс, орієнтований на потреби студентів та викладачів;
- протестувати програмний продукт на відповідність функціональним вимогам і стабільність його роботи в освітньому середовищі.

Методи дослідження. Під час розробки використано методи аналізу предметної області, порівняльного вивчення аналогічних розробок, об’єктно-орієнтованого проєктування, реалізації функціональних модулів у Java, а також емпіричного тестування з метою перевірки працездатності програмного забезпечення.

Практичне значення отриманих результатів. Розроблене програмне забезпечення може бути безпосередньо використане у навчальному процесі при вивченні тематики Java Collections, зокрема HashSet, на лекціях, практичних заняттях або в межах самостійної роботи. Продукт дозволяє не лише вивчати теоретичні аспекти структури HashSet, а й отримати практичні навички взаємодії з цим типом колекцій у реальному середовищі програмування. Завдяки доступності та інтуїтивному інтерфейсу, електронний посібник може бути корисним як студентам молодших курсів, так і викладачам, які прагнуть підвищити ефективність засвоєння матеріалу студентами.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Олена ОЛЬХОВСЬКА

« ___ » вересня 202_ р.

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка навчального програмного забезпечення за теми «Collections. HashSet» дисципліни «Програмування»

зі спеціальності 122 «Комп’ютерні науки»

освітня програма «Комп’ютерні науки»

ступеня бакалавр

Прізвище, ім'я, по батькові _____

Затверджена наказом ректора № ____-Н від « » _____ 202_ р.

Термін подання студентом роботи «__ » _____ 202_ р.

Вихідні дані до кваліфікаційно роботи: офіційні матеріали з теми Java Collections, документація щодо використання HashSet, приклади інтерактивних навчальних посібників у форматі HTML та PDF.

Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ
ВСТУП
1. ПОСТАНОВКА ЗАДАЧІ
1.1 Змістовне формулювання проблеми
1.2 Цілі та очікувані результати проєкту
Висновки до першого розділу
2. ІНФОРМАЦІЙНИЙ ОГЛЯД
2.1 Огляд сучасних електронних засобів навчання програмуванню
2.2 Переваги проаналізованих навчальних рішень
2.3 Недоліки та обмеження існуючих програмних засобів
2.4 Актуальність теми HashSet у системі Java Collections
Висновки до другого розділу
3. ТЕОРЕТИЧНА ЧАСТИНА
3.1 Концепція колекцій у Java та місце HashSet
3.2 Структура HashSet. Логіка, властивості, основні операції
3.3 Обґрунтування вибору Java, Swing та PDFBox для реалізації

Висновки до третього розділу
4. ПРАКТИЧНА ЧАСТИНА
4.1 Побудова архітектури навчального програмного забезпечення
4.2 Алгоритми обробки PDF-контенту та навігації по темах
4.3 Тестування функціональності, перевірка стабільності
4.4 Інструкція для кінцевого користувача
Висновки до четвертого розділу
ВИСНОВКИ
РЕКОМЕНДАЦІЇ
СПИСОК ЛІТЕРАТУРИ
ДОДАТОК А. Вихідний код програми

Консультанти розділів кваліфікаційної роботи

Розділ	ПІП, посада консультанта	Підпис, дата	
		завдання видав	Завдання прийняв
Постановка задачі			
Інформаційний огляд			
Теоретична частина			
Практична реалізація			

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій		
та стандартів та звіт керівнику		
3. Постановка задачі		
4. Інформаційний огляд джерел		
бібліотек та інтернету		
5. Теоретична частина		
6. Практична частина		
7. Закінчення оформлення		
8. Доповідь студента на кафедрі		

Дата видачі завдання «__» _____ 202_ р.

Здобувач вищої освіти _____.

Науковий керівник _____.

Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на _____

(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 2025 р.

Секретар ЕК_____ _____

(підпис) (ініціал та прізвище)

1.1 Змістовне формулювання проблеми

У сучасному освітньому процесі вивчення програмування займає ключове місце в підготовці майбутніх фахівців галузі інформаційних технологій. Однією з основ програмування на мові Java є робота з колекціями даних, зокрема з такою структурою, як HashSet. Ця структура є важливою не лише через її практичне застосування, а й через концептуальну значущість: вона наочно демонструє принципи зберігання унікальних елементів, механізми хешування, а також операції доступу до даних. Проте досвід викладання показує, що для багатьох студентів тематика колекцій — абстрактна і складна для швидкого засвоєння, особливо в контексті самостійного опанування або дистанційного навчання.

Типова проблема полягає в тому, що теоретичні пояснення, які подаються у вигляді текстових лекцій або PDF-документів, не завжди є достатніми для глибокого розуміння структури HashSet — її поведінки, обмежень, внутрішньої логіки. Без можливості візуалізувати ці механізми, студенту важко сформувати інтуїцію щодо того, як працює ця структура в реальному середовищі [4]. Більше того, більшість друкованих або навіть цифрових навчальних матеріалів рідко містять приклади інтеграції HashSet у практичні завдання, що ускладнює процес формування прикладних навичок.

Ще одним аспектом проблеми є фрагментарність навчального контенту. Студент змушений звертатися одночасно до різних джерел — документації, відеоуроків, прикладів коду, форумів — що створює зайве когнітивне навантаження та не сприяє системності знань. Крім того, при відсутності єдиної платформи або інтерактивного посібника, де було б зібрано в одному місці пояснення, приклади, код і засоби перевірки знань, студенти втрачають мотивацію до самостійного опрацювання матеріалу. Особливо це проявляється на ранніх етапах вивчення Java, коли рівень абстракції вищий, ніж у знайомих їм мовах типу Python чи навіть C.

Не менш важливою є проблема ефективності викладання з боку викладача. Без додаткових засобів візуалізації та демонстрації, без

інструментів, які дозволяють миттєво показати дію того чи іншого методу HashSet на конкретному прикладі, проведення занять обмежується сухим теоретичним матеріалом або демонстрацією коду в інтегрованому середовищі розробки. Такі формати не завжди відповідають дидактичним цілям, особливо у випадку студентів, які потребують більше прикладних прикладів, інтерактиву та індивідуального темпу роботи.

Ще один важливий момент полягає у відсутності зручного, платформонезалежного інструменту, який би міг працювати без потреби складного налаштування. Як наслідок — замість того, щоб вивчати навчальні матеріали, студенти витрачають час на подолання інструментальних бар'єрів. Це ще раз підкреслює потребу у простому, інтуїтивно зрозумілому програмному рішенні, орієнтованому саме на навчання.

Окрему увагу слід приділити тому, що більшість наявних рішень зосереджені на подачі матеріалу англійською мовою, що може створювати додаткові труднощі для частини студентів, які ще не мають достатнього рівня володіння фаховою термінологією іноземною мовою. Відсутність локалізованих інструментів ускладнює доступ до знань, робить навчання менш інклюзивним, що суперечить сучасним освітнім принципам рівного доступу.

Проблема полягає не лише в абстрактності теми, а в комплексі чинників: недостатній інтерактивності, фрагментарності джерел, технічній складності середовищ, відсутності локалізації, низькій візуальній підтримці, відсутності можливості самоперевірки та нерівномірному навантаженні на викладачів. Усе це створює потребу в розробці електронного навчального програмного забезпечення, яке б охоплювало зазначену тематику, забезпечувало доступність і функціональність, адаптувалося до різних платформ, містило приклади, пояснення, а також елементи контролю знань.

У зв'язку з цим особливу цінність набуває ідея створення електронного посібника, що поєднує переваги HTML-контенту, функціонал перегляду PDF-документів, інтерактивність інтерфейсу та підтримку друку й пошуку. Такий підхід не лише знімає описані проблеми, а й відкриває нові можливості для

викладання програмування — через сучасне, гнучке, зручне середовище навчання, здатне адаптуватися до потреб і викладача, і студента.

1.2 Цілі та очікувані результати проєкту

Метою цього проєкту є створення сучасного навчального програмного забезпечення, орієнтованого на викладання й самостійне вивчення теми HashSet у контексті Java Collections Framework. Програмне рішення має стати універсальним інструментом, який поєднує в собі зручний доступ до навчального матеріалу, інтерактивні елементи взаємодії, підтримку перегляду й навігації, а також засоби візуалізації та пояснення ключових понять.

Стратегічна мета — підвищення якості засвоєння теми, зменшення когнітивного навантаження на студентів і оптимізація процесу викладання для викладачів. У рамках цієї мети визначено низку конкретних цілей, що охоплюють як технічний, так і педагогічний аспект реалізації програмного продукту:

– створити повноцінний електронний посібник, що охоплює тему HashSet і базові поняття роботи з колекціями у мові Java;

– забезпечити логічну структуру подачі матеріалу з поділом на розділи, що відповідають тематиці занять, з підтримкою ієрархічної навігації через інтерактивне меню JTree;

– реалізувати перегляд навчального контенту у форматі HTML і PDF із можливістю масштабування, зміни шрифтів, пошуку за ключовими словами та виведення на друк;

– забезпечити візуальну простоту й доступність інтерфейсу за допомогою Java Swing, із урахуванням потреб користувача на рівні ергономіки, локалізації та простоти взаємодії;

– інтегрувати практичні приклади використання HashSet, що демонструють типові сценарії застосування, операції над елементами, поняття унікальності, ефективність доступу й хешування;

– протестувати програму на стабільність роботи, коректність функціоналу та відповідність педагогічним завданням.

Очікуваними результатами реалізації цього проєкту є:

– програмно реалізоване навчальне середовище, яке може бути використане як у межах традиційного аудиторного викладання, так і в дистанційній чи змішаній формах навчання;

– полегшення сприйняття складної тематики студентами завдяки візуальній подачі матеріалу та можливості практичної взаємодії з контентом;

– підвищення ефективності роботи викладача через наявність зручного інструменту демонстрації концепцій і організації навчального процесу;

– зменшення потреби у додаткових технічних засобах або складних середовищах розробки для вивчення колекцій у Java, завдяки тому, що розроблена програма працює автономно, без необхідності попереднього налаштування;

– формування у студентів не лише теоретичних знань, а й прикладних навичок роботи з HashSet у реальному середовищі, що має безпосереднє значення для їхньої майбутньої професійної діяльності.

Проєкт має на меті подолання типових проблем, пов’язаних з викладанням та вивченням теми HashSet, шляхом впровадження технологічно простого та методично продуманого програмного продукту.

Висновки до першого розділу

Здійснено ґрунтовне формулювання проблематики, що стала відправною точкою для розробки навчального програмного забезпечення, орієнтованого на тему HashSet у системі колекцій мови Java. Вивчення HashSet становить методичну цінність для формування алгоритмічного мислення студентів і потребує не лише текстового, а й візуального, інтерактивного підходу до викладання.

Було також доведено, що традиційні способи подачі цієї теми — через конспекти, фрагменти коду та статичні презентації — мають низку суттєвих

обмежень. Вони не дозволяють повною мірою продемонструвати поведінку колекції в динаміці, не дають змоги студенту експериментально перевірити гіпотези, змінюючи вхідні дані або властивості об’єктів. Таким чином, виникає потреба у створенні спеціалізованого програмного інструменту, який би реалізовував принцип «навчання через взаємодію», відкриваючи доступ до матеріалу в зручній, структурованій та адаптивній формі.

Сформульована мета дипломної роботи — створення інтерактивного навчального засобу для теми HashSet — була конкретизована через низку очікуваних результатів. Йдеться, зокрема, про реалізацію інтерфейсу, який дає змогу обирати теми, переглядати відповідні PDF-документи, масштабувати текст, здійснювати пошук за ключовими словами, переміщувати вміст і друкувати матеріал. Кінцевий продукт повинен бути інтуїтивно зрозумілим, стійким до помилок, локалізованим українською мовою та доступним для використання на різних операційних системах без складних налаштувань.

У результаті аналізу було чітко окреслено цілі завдання: розробити навчальний застосунок, який поєднує педагогічну цінність теми з інструментальною наочністю, базуючись на Java та сучасних бібліотеках роботи з PDF. Це програмне забезпечення має не лише доповнити існуючі методичні ресурси, а й створити нову модель взаємодії з навчальним контентом, що відповідає запитам сучасної цифрової освіти.

РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Огляд сучасних електронних засобів навчання програмуванню

У світлі активного впровадження цифрових технологій в освітній процес питання формування ефективного навчального середовища для вивчення програмування набуває особливої актуальності. За останнє десятиліття з'явилась велика кількість електронних ресурсів, які використовуються для засвоєння як базових, так і спеціалізованих тем з інформатики, алгоритмів та мов програмування. Електронне навчання (E-learning) відноситься до великої науково-практичної галузі, що носить загальну назву автоматизованого навчання [55]. Особливої популярності набули електронні навчальні платформи, інтерактивні тренажери, віртуальні лабораторії, а також десктопні та мобільні застосунки, орієнтовані на вивчення конкретних мов, бібліотек чи концепцій. Їхнє значення полягає не лише в зручності доступу, а й у зміні парадигми навчання — від пасивного сприйняття до активної практики й самостійного дослідження.

Широке впровадження нових інформаційних технологій веде до розвитку багатьох нових можливостей для управління процесом навчання, навіть у порівнянні з дуже недавнім минулим. Сьогодні у повній мірі може здійснюватись ідея навчання у будь-який час і в будь-якому місці. Сьогодні в Інтернет налічується понад мільйон навчальних курсів, а кількість порталів та віртуальних навчальних закладів сягає 30000 [55].

Сучасні засоби навчання програмуванню можна умовно розподілити за кількома критеріями: за формою подання матеріалу (текст, відео, симуляція), за рівнем інтерактивності (від перегляду лекцій до виконання коду онлайн), за цільовою аудиторією (початківці, студенти технічних спеціальностей, професіонали), а також за ступенем адаптивності до користувача. Найбільш поширеними є такі формати, як інтерактивні онлайн-курси (наприклад, Codecademy, Coursera, Udemy), тренажери (SoloLearn, Replit), локальні електронні посібники та інструменти самонавчання (JetBrains Academy, CS50

IDE), а також інтелектуальні навчальні середовища, що враховують прогрес і типові помилки користувача.

Більшість електронних платформ базуються на принципі гейміфікації навчання, тобто впровадженні елементів ігрового підходу: рівнів, нагород, балів, лідербордів. В основі стратегії гейміфікації лежить винагородження за виконані завдання. Можуть бути різні типи заохочень: бали, відзнаки або рівні, індикатор прогресу і видача виконавцю віртуальної валюти [28]. Це суттєво підвищує мотивацію до проходження матеріалу, особливо у молодіжній аудиторії. При цьому змістове наповнення залишається академічно коректним, а структура — логічно вибудуваною відповідно до педагогічних стандартів.

Наприклад, платформа Codecademy пропонує чітко структуровані курси з Java, де тематика колекцій, включно з HashSet, розглядається поступово, із проміжними завданнями, вбудованими редакторами коду й поясненнями в реальному часі. Такий підхід дозволяє студенту одразу бачити результат виконаного коду, експериментувати, виправляти помилки та засвоювати матеріал практично.

Ще одним прикладом є JetBrains Academy, яка пропонує повноцінні навчальні проєкти в середовищі IntelliJ IDEA. Цей інструмент розрахований на студентів з початковим або середнім рівнем підготовки та дозволяє проходити тематичні модулі в інтегрованому середовищі з автоматизованою перевіркою виконаних завдань. Проте недоліком таких систем є потреба в реєстрації, доступ до певних модулів лише на платній основі, а також часткова англомовність інтерфейсу, що ускладнює процес сприйняття інформації для студентів, які ще не мають достатнього рівня володіння англійською мовою.

Іншим поширеним типом навчального інструменту є десктопні тренажери або візуальні симулятори, що дозволяють вивчати структури даних у графічному форматі. Прикладом може слугувати візуалізатор Visualgo.net, який дає можливість побачити, як саме працюють алгоритми вставки, пошуку, видалення у множинах, списках, хеш-таблицях. Водночас подібні інструменти рідко охоплюють повноцінні курси або не надають можливості перегляду

теоретичних пояснень поряд із практикою. Більшість з них орієнтовані на короткі демонстрації алгоритмів без цілісної навчальної програми.

Ще одним помітним напрямом розвитку електронних навчальних систем є комплексні онлайн-платформи з вбудованими середовищами розробки, як-от Replit, LeetCode, HackerRank. У таких середовищах користувачеві доступні готові інтерфейси для написання, компіляції й тестування коду прямо в браузері. Це значно знижує поріг входу для початківців, оскільки немає потреби завантажувати або налаштовувати додаткові інструменти. Проте варто зауважити, що більшість із зазначених платформ розроблені переважно для просунутої аудиторії, а отже містять завдання підвищеної складності або не передбачають детального теоретичного пояснення, що робить їх менш придатними для первинного ознайомлення з темою.

Для ілюстрації можна розглянути головну сторінку платформи Replit, яка надає змогу створювати середовища для Java-програмування онлайн. Користувач може одразу запускати код, бачити результати виведення, редагувати структуру проєкту та спілкуватися з іншими учасниками. Такий підхід підтримує колаборативне навчання, однак зазвичай не має чіткого освітнього плану. Інтерфейс платформи Replit зображено на рисунку 2.1.

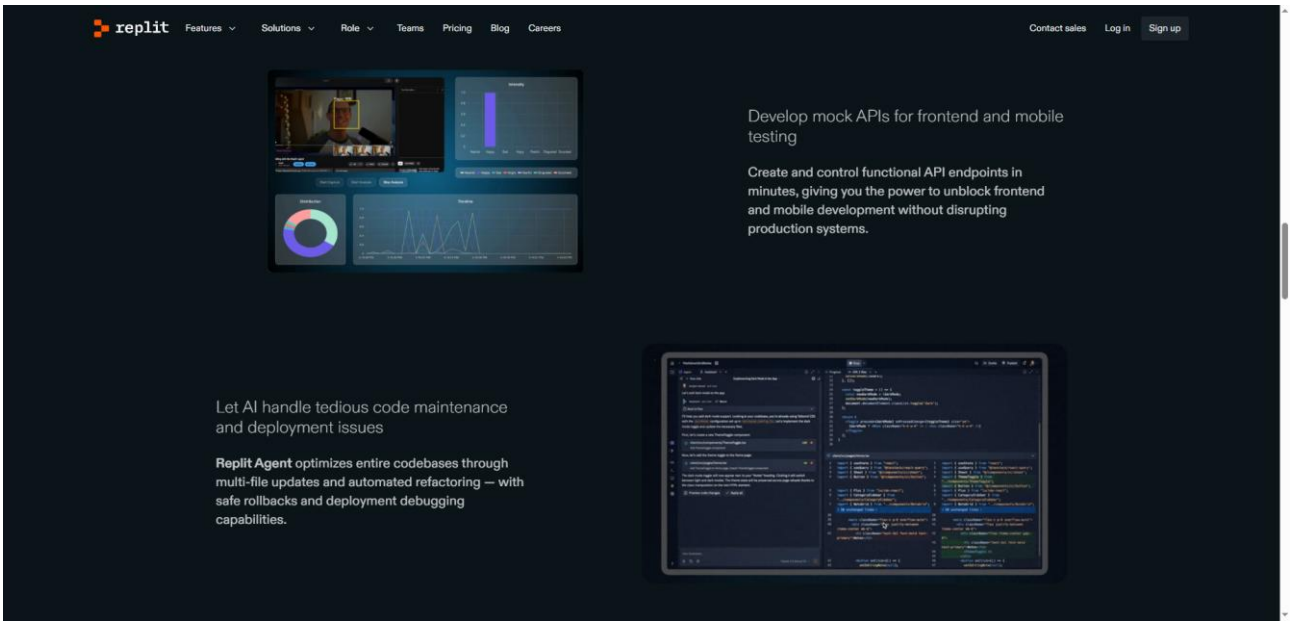

Рис. 2.1. Онлайн-середовище Replit

Подібним чином працює платформа LeetCode, яка дає змогу вирішувати алгоритмічні задачі різного рівня складності з миттєвою перевіркою

коректності розв’язку. Її основна мета — підготовка до технічних інтерв’ю, тому вона не містить глибоких вступних роз’яснень тем, але може бути використана як етап закріплення вже засвоєних знань. Що важливо — функціонал цієї платформи орієнтований на англомовну аудиторію, що знову-таки ускладнює адаптацію її до освітніх реалій українських вишів.

Ще однією категорією інструментів є інтерактивні мобільні застосунки, які призначені для швидкого опанування базових принципів програмування. До таких належить, зокрема, SoloLearn — додаток, який дозволяє вивчати Java у вигляді коротких уроків із миттєвими перевірками відповідей та кодовими фрагментами. SoloLearn підтримує візуально зручну навігацію, систему нагород, спільноту користувачів, а також модуль перевірки практичних вправ. Проте, як і багато мобільних продуктів, цей застосунок не розрахований на глибоке вивчення тем, таких як HashSet, і скоріше виконує ознайомчу функцію.

Візуальний вигляд мобільного застосунку SoloLearn зображено на рис. 2.2.

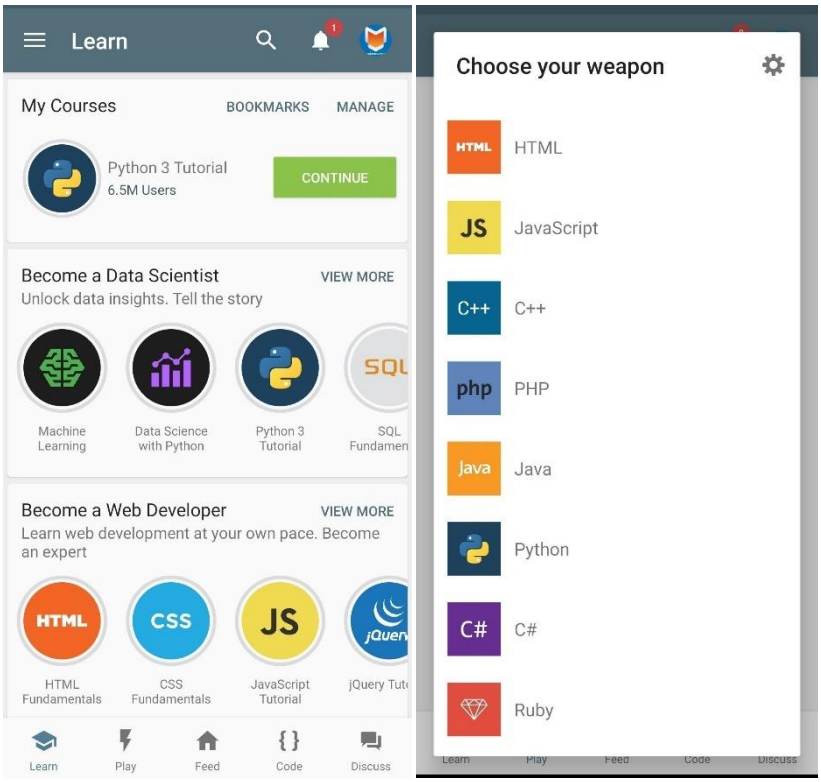

Рис. 2.2. Головне меню застосунку SoloLearn із вибором мови Java

У контексті створення спеціалізованого програмного забезпечення для вивчення HashSet доцільно враховувати не лише функціональність розглянутих

платформ, а й їхні обмеження: складність інтерфейсу для новачків, фрагментарність подання матеріалу, недостатність локалізації, обмежені можливості виводу контенту на друк або використання в офлайн-режимі. Більшість інструментів орієнтовані на онлайн-взаємодію, що в окремих випадках (нестабільне інтернет-з'єднання, технічні обмеження в регіональних навчальних закладах) може створювати додаткові бар'єри.

Варто також зазначити, що не всі існуючі рішення передбачають адаптацію під потреби навчальних курсів у класичних ВНЗ, де необхідна не лише інтерактивність, а й наявність контрольованого змісту, що відповідає навчальній програмі, а також можливість відображення на великих екранах, демонстрації в аудиторіях, сумісності з існуючими освітніми матеріалами у форматі PDF або HTML.

Саме з огляду на ці потреби й обмеження актуальною постає задача створення окремого навчального застосунку, який був би розроблений з урахуванням структури дисципліни «Програмування», містив локалізований інтерфейс, не вимагав реєстрації чи стабільного інтернету, підтримував навігацію по темах, перегляд і друк теоретичних матеріалів, а головне — робив акцент саме на тематиці HashSet, на відміну від універсальних платформ, що лише побіжно згадують про цю структуру у загальному контексті колекцій.

У подальших розділах буде показано, як розроблений електронний посібник вирішує ці проблеми завдяки поєднанню класичного підходу до структурування навчального матеріалу з сучасними інтерфейсними рішеннями та програмними бібліотеками мови Java.

2.2 Переваги проаналізованих навчальних рішень

На тлі активного впровадження цифрових технологій в освітню сферу електронні засоби навчання програмуванню набули особливого значення, ставши невід'ємною частиною підготовки майбутніх фахівців. Проведений у межах цієї роботи аналіз сучасних навчальних платформ, програм і застосунків

дозволяє виокремити низку беззаперечних переваг, які забезпечують їхню ефективність, гнучкість і популярність серед широкої аудиторії користувачів.

Насамперед, ключовою перевагою більшості розглянутих ресурсів є високий рівень інтерактивності. Інструменти на зразок Codecademy, JetBrains Academy, Replit або SoloLearn надають користувачеві змогу не лише ознайомитися з теоретичним матеріалом, а й одразу застосовувати набуті знання на практиці. Завдяки цьому навчальний процес стає не лінійним, а циклічним: студент читає, пробує, отримує зворотний зв’язок, аналізує свої помилки, і знову повертається до матеріалу. Таке поєднання теорії й практики значно підвищує якість засвоєння знань, оскільки забезпечується постійний зв’язок між абстрактними поняттями й їхньою конкретною реалізацією в коді.

Ще одна суттєва перевага — структурованість навчального матеріалу. Майже всі сучасні освітні платформи будують свої курси на логічно впорядкованих модулях, де кожен новий крок базується на вже засвоєному. Це дозволяє уникати перевантаження інформацією, підтримує когнітивну послідовність і зменшує ймовірність дезорієнтації в темі [55]. Наприклад, у JetBrains Academy тематика Java Collections подається поетапно: спочатку базові поняття множин, далі структура HashSet, потім операції над елементами, хешування тощо. Таке поетапне ускладнення дозволяє студенту комфортно освоювати навіть складні концепції.

Також варто відзначити високу доступність та простоту використання більшості проаналізованих рішень. Багато платформ працюють безпосередньо в браузері, не вимагають встановлення програмного забезпечення або попередньої конфігурації середовища розробки. Це стосується зокрема таких інструментів, як Replit, SoloLearn або HackerRank. У випадку початківців цей фактор відіграє вирішальну роль, оскільки знижує технічний бар’єр і дозволяє зосередитися безпосередньо на навчальному матеріалі. Крім того, можливість працювати з будь-якого пристрою (ПК, планшета, смартфона) робить ці ресурси зручними в умовах мобільного або гібридного навчання.

Окремої уваги заслуговує наявність автоматизованої перевірки результатів і миттєвого зворотного зв’язку. Ця функція реалізована майже у

всіх сучасних тренажерах і онлайн-курсах. Платформи на кшталт LeetCode чи HackerRank одразу показують, які тести були пройдені успішно, де є помилки й до яких аспектів слід повернутись. Це сприяє швидкій самооцінці, формуванню навичок рефлексії та знижує залежність від зовнішнього контролю, що важливо в умовах самостійного навчання.

Ще одним важливим аспектом є інтуїтивна простота інтерфейсу. Переважна більшість систем мають мінімалістичний, але водночас функціональний дизайн, у якому легко орієнтуватися навіть користувачу без попереднього досвіду програмування. Це стосується як мобільних застосунків, таких як SoloLearn, так і повноцінних вебплатформ, як-от Codecademy або JetBrains Academy. Зручне розташування елементів, підказки, підсвічування синтаксису, покрокові інструкції — усе це створює комфортне навчальне середовище.

Важливою перевагою сучасних навчальних рішень є також використання елементів гейміфікації, що дозволяє значно підвищити мотивацію студентів. Платформи типу SoloLearn, Codecademy, а також JetBrains Academy впроваджують систему балів, досягнень, рівнів або бейджів, які користувач отримує за виконання завдань. Подібна система заохочення формує в користувача додаткову зацікавленість, підтримує інтерес до теми навіть у складних моментах і створює ефект поступової "гри", в якій просування по темі стає не лише навчальним, а й емоційно значущим процесом [28]. У поєднанні з інтерактивними вправами, такий підхід формує стійку навчальну траєкторію, що базується на внутрішній мотивації.

Також не можна оминути перевагу інтеграції з реальним середовищем розробки, яка присутня на платформах JetBrains Academy та Replit. У першій із них користувач виконує завдання безпосередньо в середовищі IntelliJ IDEA, що забезпечує знайомство з інтерфейсом професійного рівня. У другій — реалізується підтримка багатьох мов програмування, включно з Java, у середовищі, яке не потребує встановлення IDE. Таке наближення до реальних інструментів роботи формує у студента впевненість, зменшує страх перед

професійним середовищем, що особливо важливо при переході від навчання до практики.

Варто також виділити можливість навчання у власному темпі. Студенти можуть опрацьовувати матеріал індивідуально, у своєму ритмі. Виконувати завдання, проходити уроки й досліджувати матеріал тоді, коли зручно [26]. Більшість платформ не встановлюють жорсткого графіку проходження курсу, що дозволяє студенту повертатися до складних тем, переглядати пройдені уроки, повторювати вправи. Наприклад, користувач Codecademy або JetBrains Academy може повністю зупинити курс, а згодом відновити його саме з того місця, на якому було перервано, що ідеально підходить для комбінування з іншими видами навчального навантаження або частковою зайнятістю.

Одним із найбільш корисних аспектів є наявність спільноти. Форуми, чати, розділи з коментарями до завдань або прикладів коду — усе це створює середовище, де користувач може отримати допомогу, підтримати інших або поділитися власним досвідом. Наприклад, платформи на кшталт HackerRank і LeetCode мають активні технічні спільноти, де розглядаються різні стратегії розв’язання задач, обговорюються ефективні підходи до оптимізації алгоритмів, зокрема таких, що використовують множини, хеш-таблиці чи HashSet.

Крім того, сучасні електронні засоби навчання підтримують адаптивність навчального контенту, яка полягає в тому, що користувачеві пропонуються вправи або пояснення з урахуванням рівня його підготовки або типових помилок. Цей принцип особливо актуальний у контексті вивчення таких тем, як HashSet, де неправильне розуміння унікальності елементів або механізму хешування може призвести до типових помилок, а завдання з адаптивним підбором складності дозволяють уникати їх повторення.

Розглянуті рішення також відзначаються гнучкістю у форматах подачі матеріалу. Деякі курси комбінують текстові пояснення з інтерактивними симуляціями, деякі — додають відеоматеріали, короткі підсумки або тести для перевірки знань. Це дозволяє користувачам з різними стилями навчання

(візуальним, аудіальним, кінестетичним) знайти найзручніший спосіб взаємодії з темою.

Підсумовуючи аналіз позитивних характеристик існуючих навчальних рішень, можна зробити висновок, що вони значною мірою відповідають викликам сучасної освіти: підтримують інтерактивність, доступність, візуалізацію, адаптивність та індивідуалізацію. Усе це створює якісне освітнє середовище, яке є цінним джерелом ідей для створення власного електронного посібника з теми HashSet, що успадковуватиме найкращі напрацювання, уникаючи при цьому властивих недоліків, які буде розглянуто у наступному підрозділі.

2.3 Недоліки та обмеження існуючих програмних засобів

Попри численні переваги, виявлені в процесі аналізу сучасних електронних засобів навчання програмуванню, не менш важливим є розгляд їхніх недоліків і обмежень. Це дозволяє не лише критично осмислити наявні рішення, а й окреслити потенційні шляхи вдосконалення при створенні нового навчального програмного забезпечення. У більшості випадків виявлені недоліки мають системний характер і стосуються як педагогічних, так і технічних аспектів функціонування платформ.

Однією з найпоширеніших проблем є відсутність локалізованого інтерфейсу. Більшість потужних платформ — зокрема Codecademy, JetBrains Academy, LeetCode, Replit — подають навчальні матеріали виключно англійською мовою. Хоча це може бути зручно для студентів із достатнім рівнем володіння мовою, для багатьох початківців та студентів молодших курсів це стає серйозною перешкодою. Вивчення складних термінів і синтаксичних конструкцій програмування одночасно з перекладом нових понять іноземною мовою створює когнітивне перевантаження, знижує мотивацію й часто призводить до фрустрації.

Іншою важливою проблемою є фрагментарність подання навчального контенту. У багатьох системах, навіть при наявності курсової структури, зміст

розбитий на дрібні сегменти без цілісного підходу до формування концептуального бачення теми. Наприклад, у темі колекцій Java, і тим більше — HashSet, іноді розглядається лише один аспект — скажімо, операція додавання — без пояснення загальної логіки зберігання елементів, механізму хешування або впливу колізій. Це призводить до формального засвоєння окремих технічних дій, але не сприяє формуванню повноцінного розуміння структури даних як цілісної системи.

Ще один суттєвий недолік — відсутність можливості працювати офлайн або за нестабільного інтернет-з'єднання. Більшість сучасних платформ є повністю хмарними, що передбачає постійний доступ до мережі. У ситуаціях, коли інтернет-з'єднання є нестабільним, повільним або недоступним (що досі є характерною проблемою в окремих регіонах), користувач втрачає доступ до свого навчального процесу. Це особливо критично для студентів у державних навчальних закладах із обмеженою технічною інфраструктурою.

Також слід зазначити відсутність можливості адаптувати поданий матеріал під специфіку конкретної навчальної дисципліни або теми. Наприклад, якщо платформа подає загальну тему Java Collections, вона рідко дозволяє викладачеві виключити певні модулі, перетворити матеріал на контрольовану підмножину знань або доповнити авторськими прикладами, які відповідають навчальній програмі в межах дисципліни «Програмування». У результаті викладач змушений шукати компроміс між тим, що пропонує платформа, і тими знаннями, які студент має засвоїти згідно з навчальним планом.

Суттєвим обмеженням є також залежність багатьох інструментів від платних підписок або часткова недоступність повного функціоналу у безкоштовній версії. Така модель використання, характерна для Codecademy, JetBrains Academy, а частково й для SoloLearn, обмежує можливості навчання для студентів, які не можуть дозволити собі платний доступ. В умовах державних або комунальних навчальних закладів це перетворюється на серйозну проблему, оскільки платформи не завжди передбачають академічні ліцензії або відкритий доступ до повного курсу безкоштовно. У результаті

студент отримує лише фрагменти навчального матеріалу або стикається з обмеженням кількості інтерактивних вправ.

Окремо варто згадати про відсутність функціоналу для друку, збереження або експорту навчального контенту у форматах, придатних для використання під час занять. Переважна більшість сучасних платформ не передбачає виводу матеріалу у вигляді PDF-документів або локального HTML, що обмежує використання ресурсів у класичних навчальних форматах — наприклад, для роздачі роздрукованих матеріалів, демонстрації на інтерактивній дошці або самостійного опрацювання без комп’ютера. Така ситуація суперечить сучасним принципам інклюзивної освіти, де студент має право обирати зручний формат доступу до знань.

Ще одним недоліком є відсутність централізованого візуального представлення навчального матеріалу у формі, зручній для навігації, зокрема за допомогою тематичних дерев або контекстного змісту. В окремих випадках користувач змушений гортати десятки сторінок курсу, щоб знайти потрібну тему або повернутись до попереднього модуля. Це ускладнює навігацію й знижує ефективність самостійного навчання, особливо коли матеріал використовується для підготовки до іспитів або тестування.

Зважаючи на ці недоліки, можна стверджувати, що навіть найпопулярніші сучасні освітні платформи не завжди відповідають вимогам академічного навчального процесу в повному обсязі.

Ці дані підтверджують, що хоча існуючі інструменти мають багато сильних сторін, жоден із них не пропонує цілісного рішення, спеціально адаптованого під тематику HashSet у межах вітчизняної навчальної програми. Відсутність локалізації, інструментів друку, повної офлайн-доступності, адаптації під викладача — це ті обмеження, які слід врахувати при створенні власного навчального програмного забезпечення.

У таблиці 2.1. представлено стислий огляд основних характеристик деяких з розглянутих рішень, з акцентом на їх перевагах і обмеженнях у контексті теми HashSet та потреб навчального процесу.

Таблиця 2.1.

Порівняльний огляд програм-аналогів для вивчення програмування

Назва платформи	Переваги	Недоліки
Codecademy	Інтерактивні вправи, поступова подача матеріалу, підтримка Java	Англомовний GUI, обмежений доступ у безкоштовній версії
JetBrains Academy	Поглиблене вивчення Java у середовищі IntelliJ IDEA, модульна структура	Вимагає встановлення IDE, частина контенту недоступна без підписки
Replit	Онлайн-компілятор Java, доступність без реєстрації, простий інтерфейс	Відсутність навчальної структури, фокус на середовище, а не на педагогіку
LeetCode	Підбір задач за рівнем, миттєва перевірка результату	Відсутність теорії, розраховано на просунутий рівень, англомовна платформа
SoloLearn	Мобільний доступ, інтерактивність, система мотивації	Поверхневий рівень подачі, обмежене охоплення теми HashSet
Visualgo.net	Візуалізація структур даних, простота використання	Не має навчального курсу, відсутній текстовий супровід і приклади з реальними сценаріями

2.4 Актуальність теми HashSet у системі Java Collections

У межах вивчення мови програмування Java одним із найважливіших компонентів є система колекцій, відома як Java Collections Framework. Вона забезпечує інструментарій для роботи з динамічними структурами даних, які широко використовуються у реальному програмуванні — від простих програм до великих корпоративних систем [21]. У цьому контексті структура HashSet посідає особливе місце, оскільки є однією з базових реалізацій множин, що забезпечує ефективне зберігання та обробку унікальних елементів. Вивчення

HashSet є не лише важливим кроком у засвоєнні колекцій, а й формує фундаментальні уявлення про принципи хешування, ефективність алгоритмів і поведінку структур у пам’яті.

З технічної точки зору HashSet реалізовано як обгортка над HashMap, де всі значення зберігаються як ключі, а значенням виступає службовий об’єкт. Це дозволяє уникати дублювання елементів, забезпечувати доступ до них із постійною середньою складністю та оперувати основними діями — вставка, видалення, перевірка наявності — з високою продуктивністю. Але, попри цю технічну простоту, розуміння внутрішньої логіки HashSet — зокрема, як саме обчислюються хеш-коди, як впливають колізії, чому об’єкти мають перевизначати методи hashCode() і equals() — вимагає від студента вже не лише поверхневого ознайомлення, а серйозного аналітичного мислення.

Актуальність теми HashSet також зумовлена її широким практичним застосуванням у різноманітних сферах. У задачах, де важливо зберігати лише унікальні значення — наприклад, під час фільтрації дублюючих записів, побудови множин ключових слів, обліку унікальних користувачів чи генерування множин у складних логічних обчисленнях — використання HashSet є природним і рекомендованим. Вміння правильно працювати з цією структурою є обов’язковим для кожного Java-розробника, а отже її викладання має займати окреме місце в освітній програмі.

Ще один важливий аспект — принципова відмінність HashSet від інших типів колекцій, таких як ArrayList, LinkedList, TreeSet тощо. Кожна з цих структур має своє призначення, а HashSet виділяється саме вимогою до унікальності елементів і відсутністю гарантій щодо порядку їх зберігання. Така поведінка є непритаманною для новачків, які звикли до послідовного або індексованого доступу до елементів. Саме тому її викладання потребує окремого інструменту — такого, що дає змогу побачити, як HashSet поводиться у різних сценаріях, чому важливо враховувати колізії, як відбувається порівняння об’єктів та чому дві зовні однакові структури можуть мати різний результат при ітерації.

Варто також зазначити, що попри важливість HashSet, у більшості традиційних курсів з програмування на Java ця тема подається як допоміжна або побіжно згадується у контексті загального огляду Java Collections. У навчальних посібниках часто приділяється більше уваги ArrayList або LinkedList, оскільки вони ближчі до класичних масивів, із якими знайомі студенти. Через це HashSet залишається недооціненою темою, яку студенти або не розуміють, або використовують без чіткого усвідомлення механізмів її дії. Відсутність спеціалізованих матеріалів або інструментів для вивчення HashSet призводить до поверхневого засвоєння, що пізніше створює проблеми при розв’язанні практичних задач.

Окремо слід акцентувати увагу на тому, що HashSet — це не лише структура даних у рамках навчальної теми, а невід’ємна складова прикладного програмування в Java, яка широко використовується у розробці вебдодатків, мобільних застосунків, мікросервісів, баз даних і навіть у тестуванні [65]. Наприклад, у системах автентифікації для зберігання унікальних токенів доступу чи у модулях кешування для обмеження дублювання даних — HashSet виконує технічно значущі ролі. Це робить її розуміння необхідним не лише з точки зору академічної підготовки, а й як базову компетенцію майбутнього інженера-програміста.

Інструментальна важливість HashSet також полягає у її поєднанні з іншими елементами Java Collections, наприклад, при перетворенні колекцій між різними типами (List → Set, Set → Map), що часто трапляється у реальних програмних проєктах. Розуміння особливостей HashSet, таких як відсутність дублювання, незалежність від порядку додавання, залежність від коректно реалізованих методів hashCode() та equals() — усе це має вирішальне значення при побудові програм, де коректність логіки базується на точності обробки множин. Це особливо важливо в освітньому процесі, де студенти мають не лише реалізовувати приклади, а й навчитися передбачати поведінку структури в умовах, близьких до практичних.

Актуальність теми HashSet підтверджується й тим, що вона є частиною офіційної програми сертифікації Oracle Certified Professional Java Programmer

(OCPJP), яка визнана міжнародною спільнотою [70]. Це означає, що володіння цією структурою вважається обов’язковою частиною технічної компетентності фахівця, і її викладання має бути впроваджене в навчальні програми не на рівні другорядного розділу, а як окрема, логічно завершена тема з належним поясненням і практичним супроводом.

З огляду на це, створення спеціалізованого навчального засобу, який не лише містить теоретичний виклад теми, але й дозволяє студенту взаємодіяти з контентом у динамічному режимі, бачити зміну стану колекції, експериментувати з додаванням і видаленням елементів, відстежувати поведінку при зіткненнях хешів і при повторному додаванні елементів — є надзвичайно актуальним і педагогічно виправданим.

У цьому контексті розробка електронного навчального програмного забезпечення, що дає змогу наочно працювати з темою HashSet, значною мірою усуває ті бар’єри, які виникають при вивченні цієї теми традиційними методами. Воно дозволяє студентам:

– бачити результат змін у реальному часі;

– спостерігати за поведінкою при спробі додати дублікати;

– тестувати власні реалізації класів з перевизначенням equals() та hashCode();

– працювати з українськомовним інтерфейсом і читабельною структурою тем;

– самостійно переглядати, шукати, друкувати та вивчати потрібні матеріали.

Висновки до другого розділу

Проведене дослідження дозволило виявити ключові напрями розвитку освітніх технологій, охарактеризувати наявні програмні продукти, що використовуються для вивчення структур даних, та окреслити як їхні переваги, так і критичні обмеження, які заважають ефективному засвоєнню теми HashSet.

З огляду на результати аналізу було встановлено, що багато сучасних освітніх платформ — таких як SoloLearn, JetBrains Academy, Codecademy та інші — активно впроваджують інтерактивні методики, адаптивний контент, візуальні підказки, автоперевірку коду. Позитивним моментом у їх роботі є доступність, інтеграція теорії з практикою, гейміфікація та модульна структура. Саме ці характеристики забезпечують високу мотивацію студентів і сприяють більш глибокому зануренню у навчальний матеріал.

Водночас проаналізовані рішення виявилися недостатньо гнучкими, коли йдеться про вузькоспеціалізовані теми, такі як HashSet. Більшість платформ зосереджені на загальних основах Java і не мають повноцінної інтерактивної підтримки складних абстракцій типу хешованих множин. До того ж значна частина ресурсів є англомовною, що створює бар’єри для частини студентів. У деяких випадках спостерігається перенавантаженість інтерфейсу, обмежена можливість самостійної модифікації вмісту або відсутність офлайн-доступу — що важливо для освітнього процесу в умовах технічних обмежень або нестабільного підключення до мережі.

Проведений інформаційний огляд не лише підтвердив доцільність створення окремого програмного засобу для викладання теми HashSet, а й надав чіткі орієнтири для його функціонального та методичного наповнення. Було обґрунтовано, що програмне забезпечення повинне поєднувати простоту використання, локалізацію, інтуїтивну навігацію, підтримку PDF-матеріалів, адаптивність до навчального контексту, а головне — орієнтацію на глибоке, а не поверхневе засвоєння теми.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1 Концепція колекцій у Java та місце HashSet

У мові програмування Java робота з колекціями даних — одна з ключових тем, яка дозволяє ефективно оперувати динамічними наборами об’єктів, виконувати сортування, пошук, фільтрацію, об’єднання та інші операції, необхідні в реальному програмуванні. Відмова від обмежених масивів і перехід до більш гнучких структур була обумовлена як зростанням складності програмних систем, так і потребою в універсальних рішеннях для зберігання великих обсягів неоднорідних або змінних даних. Саме тому в Java було створено потужну бібліотеку під назвою *Java Collections Framework (JCF)* — гнучкий і уніфікований API, що дозволяє працювати з наборами об’єктів у стандартизований спосіб [21].

Java Collections Framework об’єднує в собі різноманітні інтерфейси, абстрактні класи й конкретні реалізації структур даних. До основних інтерфейсів належать Collection, List, Set, Queue, Deque, Map, а також допоміжні, як-от Iterator, Comparable, Comparator. Така ієрархічна побудова дозволяє досягти високої гнучкості: змінюючи реалізацію (наприклад, ArrayList на LinkedList або HashSet на TreeSet), програміст не змінює логіку взаємодії з об’єктами. Це відповідає принципам ООП — інкапсуляції, поліморфізму й повторного використання коду.

У цій системі Set — це спеціалізований тип колекції, який забезпечує зберігання лише унікальних елементів. Це особливо важливо в тих задачах, де значення не мають повторюватися, наприклад, під час перевірки належності до множини, зберігання ключових слів або обліку унікальних об’єктів. Реалізації інтерфейсу Set включають HashSet, LinkedHashSet та TreeSet, кожна з яких має свою специфіку: TreeSet зберігає елементи у відсортованому порядку, LinkedHashSet — у порядку вставлення, а HashSet — без будь-якої гарантії порядку, зате з найвищою продуктивністю при типовому використанні.

HashSet — одна з найбільш широко використовуваних реалізацій множин у Java. Вона реалізована як обгортка над HashMap, де кожен елемент множини є ключем у внутрішній мапі [5, 9]. Це дозволяє досягти високої швидкодії для базових операцій. Продуктивність досягається завдяки використанню хеш-функції, яка дозволяє обчислити унікальний "індекс" для кожного об’єкта й зберігати його в хеш-таблиці. Саме тому ця структура даних є незамінною у багатьох сферах, де потрібна висока швидкість обробки великих обсягів інформації без дублювання елементів.

Проте ефективність використання HashSet напряму залежить від коректності реалізації методів hashCode() і equals() у класах, екземпляри яких додаються до множини. Якщо ці методи не перевизначені або зроблено це неправильно, поведінка HashSet стає непередбачуваною: дублікати можуть зберігатися, елементи можуть не знаходитися в колекції, або ж може відбуватися некоректне видалення. Це створює додаткові вимоги до розробника — розуміти принципи хешування, вміти перевизначати відповідні методи і тестувати поведінку класів при роботі в середовищі HashSet.

У структурі Java Collections саме HashSet вважається реалізацією множини без гарантії порядку, але з максимальною швидкістю дій. Її використовують тоді, коли:

– потрібне швидке додавання або перевірка елементів;

– важлива гарантія унікальності значень;

– порядок елементів не має значення;

– зберігаються об’єкти з унікальними ознаками (ID, назви, коди тощо).

Щоб глибше зрозуміти місце HashSet у системі колекцій, варто порівняти його з іншими реалізаціями множин і подивитися, яку саме нішу він займає в архітектурі Java Collections Framework. Наприклад, у випадках, коли потрібно зберігати дані в упорядкованому вигляді, доцільніше використовувати TreeSet, який підтримує природне сортування або надає можливість визначити власний компаратор. Якщо ж важливо зберегти порядок додавання елементів, обирають LinkedHashSet, який поєднує принципи хешування з двозв’язним списком. Однак у більшості повсякденних задач саме HashSet є оптимальним вибором,

оскільки забезпечує найкраще співвідношення між простотою реалізації та швидкістю виконання.

Ще однією причиною актуальності HashSet є те, що ця структура є чудовим прикладом для навчання принципів роботи з хеш-функціями, колізіями та розподілом даних у пам’яті. Навчальні курси, які пояснюють тільки дії типу add(), remove(), contains() — не розкривають суті того, чим відрізняється хешування від індексації у масивах або чому однакові на вигляд об’єкти можуть поводитися по-різному у різних множинах. HashSet надає зручний майданчик для експериментів і практичних демонстрацій: наприклад, як впливає зміна логіки equals() на наявність дублювання в колекції; або як візуалізувати механізм розміщення об’єктів за допомогою хеш-функції в уявному масиві корзин.

Враховуючи все це, важливим дидактичним завданням стає не просто пояснення методів API, а наочна демонстрація поведінки HashSet у різних ситуаціях — додавання елементів, спроба повторного додавання, використання різних типів даних, перетворення List у Set для усунення дублікатів, зміна стану колекції внаслідок видалення тощо. Таку демонстрацію неможливо повноцінно реалізувати лише за допомогою підручника або фрагмента коду на слайді презентації — потрібен інструмент, який дозволить студенту взаємодіяти з колекцією в реальному часі, бачити результат змін і візуально усвідомлювати логіку її поведінки.

І саме тут на перший план виходить потреба у створенні спеціалізованого навчального програмного забезпечення, яке не лише висвітлює тему HashSet, а й дозволяє студенту експериментувати з нею. Такий посібник має включати:

– пояснення структури колекції й її місця в ієрархії Java Collections Framework;

– порівняння HashSet з іншими колекціями;

– демонстрацію реальних прикладів використання у Java-програмах;

– візуалізацію ключових понять: дублювання, хешування, колізії, вплив переозначення методів;

– інтерфейс, у якому студент може самостійно змінювати вхідні дані й бачити зміну стану колекції.

Розглянута концепція колекцій у Java демонструє складну, багаторівневу структуру, що базується на абстракціях і контрактному програмуванні. У такій системі HashSet — це не просто ще один клас, а ідеальний об’єкт для навчання принципів унікальності, ефективності, порівняння об’єктів і основ хеш-алгоритмів. Ураховуючи її практичну цінність, логіку роботи, роль у каркасі колекцій і недостатнє висвітлення в більшості традиційних курсів, створення окремого інструменту для навчання цій структурі виглядає цілком обґрунтованим кроком.

3.2 Структура HashSet. Логіка, властивості, основні операції

Колекція HashSet у Java є однією з базових структур, що реалізує інтерфейс Set і використовується для зберігання множини об’єктів без дублікатів. На перший погляд HashSet здається простою в користуванні — достатньо лише додати елементи за допомогою методу add() і впевнено очікувати, що жодне повторення не буде збережене. Однак внутрішня логіка цієї структури складніша, ніж у класичних списків або масивів, і побудована на концепції хеш-таблиці, що забезпечує високу ефективність виконання основних операцій.

На технічному рівні HashSet базується на класі HashMap. Кожен об’єкт, який додається до HashSet, насправді записується як ключ у внутрішньому екземплярі HashMap, а значенням для нього виступає службова константа (часто це new Object()) [21, 9]. Завдяки цій реалізації досягається виняткова швидкість операцій — додавання, перевірки наявності, видалення елементів. Водночас реалізація HashSet передбачає використання хеш-функцій, тобто спеціального механізму, який переводить об’єкти у числове значення (хеш-код), що й визначає розміщення елемента в структурі.

Це підводить до першого й найважливішого елемента логіки HashSet: використання методів hashCode() та equals(). Саме ці методи формують ядро

поведінки множини: hashCode() визначає, у яку "корзину" хеш-таблиці буде поміщено елемент, а equals() використовується для уточнення, чи є рівним уже збережений елемент із тим, що додається. Тому якщо студент створює власний клас і хоче використовувати об’єкти цього класу в HashSet, обов’язковою вимогою є коректне перевизначення цих двох методів. Без цього HashSet може поводитись непередбачувано: зберігати дублікати, не знаходити елементи, не видаляти те, що було додано.

Другим фундаментальним аспектом є відсутність гарантії порядку зберігання елементів. На відміну від списків (ArrayList, LinkedList), де елементи зберігаються у порядку вставлення або за індексами, у HashSet порядок вставлення не впливає на порядок доступу. Це означає, що при кожному запуску програми чи навіть під час модифікації колекції, порядок відображення елементів може змінюватися. Для початківців це є неочікуваною поведінкою й часто сприймається як помилка. Саме тому при викладанні теми HashSet важливо підкреслювати цю властивість і вказувати на альтернативи (наприклад, LinkedHashSet), які дозволяють зберігати порядок додавання.

Ще однією визначальною характеристикою HashSet є унікальність елементів. Структура ніколи не дозволить зберегти два однакових елементи. Спроба повторно додати об’єкт, який вже міститься у множині (згідно з логікою equals()), не призведе до зміни структури. Це дозволяє використовувати HashSet як механізм фільтрації — наприклад, при збиранні унікальних ключових слів, очищенні масиву від повторень, побудові множини об’єктів із файлів, таблиць чи баз даних.

У практичній роботі з HashSet користувач стикається з обмеженим, але достатньо потужним набором методів. Найчастіше використовуються такі операції:

– add(E e) – додає елемент до множини, якщо його ще не існує. Повертає true, якщо елемент був доданий, і false, якщо такий елемент уже був присутній;

– remove(Object o) – видаляє вказаний об’єкт із множини, якщо він у ній є;

– contains(Object o) – перевіряє, чи присутній об’єкт у множині;

– isEmpty() – перевіряє, чи порожня множина;

– clear() – видаляє всі елементи з множини;

– size() – повертає кількість елементів у множині;

– iterator() – повертає об’єкт Iterator для проходження по всіх елементах множини.

Кожен із цих методів базується на принципах хеш-таблиці: визначення унікальності, розміщення за хеш-кодом, швидкий доступ без необхідності проходження всього набору. Це робить HashSet оптимальним вибором у тих сценаріях, де потрібна оптимізація швидкодії при роботі з великими обсягами даних.

Важливою педагогічною задачею є пояснення того, як саме здійснюється порівняння елементів, зокрема у випадках, коли до множини додаються об’єкти власних класів. Щоб узагальнити основні властивості HashSet та краще зрозуміти, чим вона відрізняється від споріднених структур, доцільно використати порівняльну таблицю 3.1.

Таблиця 3.1.

Порівняння основних реалізацій множин у Java

Структура	Гарантія унікальності	Збереження порядку	Сортування елементів	Швидкодія операцій	Типова реалізація хешування
HashSet	Так	Ні	Ні	Висока	Так
LinkedHashSet	Так	Так (порядок вставки)	Ні	Дещо нижча, ніж у HashSet	Так
TreeSet	Так	Так. Відповідно до порядку сортування	Так. Натуральний порядок або компаратор	Низька	Ні. Використовує дерево

Як показано на таблиці, HashSet не має функцій сортування або запам’ятовування порядку, але натомість забезпечує максимальну швидкість і простоту. Саме тому її доцільно використовувати у тих випадках, коли пріоритетом є ефективність доступу, а не логічна послідовність обробки

елементів. У навчальному процесі це дозволяє ілюструвати на прикладах важливість розуміння не лише синтаксису мови Java, а й внутрішніх механізмів, які керують логікою роботи з даними.

3.3 Обґрунтування вибору Java, Swing та PDFBox для реалізації

Одним із найважливіших рішень під час створення навчального програмного забезпечення є вибір мови програмування та технологічного стеку, який забезпечить не лише функціональну придатність, а й зручність, стабільність і довготривалу підтримку розробки. У межах цієї дипломної роботи для реалізації електронного посібника з теми HashSet було обрано мову програмування Java, графічний інтерфейс на базі бібліотеки Swing, а також бібліотеку PDFBox для роботи з PDF-документами. Такий вибір є виваженим і обґрунтованим як з технічної, так і з методичної точки зору.

Почнемо з мови програмування Java, яка є однією з найпоширеніших і найстабільніших мов у світі. Її застосовують для розробки програмного забезпечення в корпоративному секторі, у мобільних застосунках, серверній логіці та навчальному контенті. Java є строго типізованою, об’єктно-орієнтованою мовою, що робить її ідеальною для формування в студентів правильного підходу до побудови програм. Вона має розгалужену документацію, численні спільноти підтримки, навчальні ресурси, а також високу стабільність API, що знижує ризик технічної застарілості [10].

Крім того, Java — це платформонезалежна мова, яка виконується у віртуальному середовищі (JVM), що забезпечує кросплатформність: розроблену програму можна запускати на Windows, Linux чи macOS без зміни коду [5]. У контексті освітнього продукту це надзвичайно важливо, оскільки університетські комп’ютери можуть мати різні операційні системи, а студенти — різні пристрої. Таким чином, використання Java дозволяє створити продукт, що не обмежений конкретною платформою, і забезпечити максимальну доступність.

Щодо інтерфейсу користувача, вибір бібліотеки Swing пояснюється низкою практичних міркувань. Swing — це стандартна графічна бібліотека Java, яка дозволяє створювати повноцінні десктопні застосунки зі зручним візуальним інтерфейсом. Її ключова перевага полягає у тому, що вона вже вбудована у JDK, не потребує зовнішніх залежностей, легко налаштовується та дозволяє повністю контролювати вигляд елементів — від кнопок і полів введення до дерев каталогів і панелей навігації [14].

Для навчального програмного забезпечення, яке має виводити змістовну інформацію, дозволяти масштабування, навігацію, друк і пошук, потрібна гнучка й адаптивна система компонентів, і саме Swing найкраще підходить для цього завдання. Бібліотека дозволяє створити ієрархічну структуру тем за допомогою компонента JTree, реалізувати верхню панель керування з кнопками масштабування, пошуку, друку, а також забезпечити головну панель перегляду навчального матеріалу. Завдяки Swing інтерфейс можна локалізувати українською мовою, підігнати під конкретні потреби студентів та викладачів, уникнувши перевантаження зайвими елементами [30].

Крім функціоналу керування, важливою вимогою до навчального програмного забезпечення було забезпечення перегляду теоретичних матеріалів у форматі PDF, оскільки цей формат є найпоширенішим у навчальному середовищі. Саме для цього було обрано бібліотеку Apache PDFBox, яка є відкритим інструментом для роботи з PDF-документами в Java.

Бібліотека Apache PDFBox дозволяє виконувати широкий спектр дій із PDF-документами: завантаження, рендеринг, читання тексту, видобування зображень, створення й редагування сторінок, а також друк. У рамках розробки навчального посібника функціонал бібліотеки було використано для візуалізації навчальних матеріалів у вигляді окремих PDF-файлів, які відображаються у головному вікні програми. Це дало змогу поєднати структурованість електронної книги з гнучкістю інтерактивного застосунку.

З технічного погляду, PDFBox надає засоби створення PDDocument (представлення документа у пам’яті), PDFRenderer для перетворення сторінок у зображення, які можна вставити в панель перегляду, а також PDFTextStripper,

що дозволяє реалізувати функцію пошуку за текстом — одну з ключових для користувача можливостей. Окрім того, PDFBox підтримує друк сторінок через клас PDFPageable, що було реалізовано у вигляді кнопки друку у верхній панелі управління. Це дозволяє студенту легко виводити потрібні розділи на папір без необхідності відкривати інші застосунки чи здійснювати додаткові перетворення.

Реалізація з PDFBox також дозволяє динамічно оновлювати зміст вікна перегляду відповідно до обраної теми в дереві навігації. Завдяки цьому було створено систему, де кожен підрозділ теми Collections або HashSet відкривається в окремому PDF-файлі, що дозволяє як швидко орієнтуватися в матеріалі, так і контролювати його подачу — наприклад, під час занять або для самостійного вивчення. Така логіка роботи унеможливлює ситуацію, коли студент перевантажений великою кількістю інформації на одному екрані, а також сприяє фокусуванню на окремих поняттях.

Ще однією важливою перевагою обраного інструментарію є висока стабільність і документованість кожного з компонентів. Java має надзвичайно розвинену екосистему, яка гарантує підтримку з боку спільноти та офіційних оновлень. Swing, попри свою тривалу історію, залишається стабільною частиною JDK, а PDFBox — активно підтримуваним проєктом Apache Software Foundation, з широкою базою прикладів, документації та адаптацією до нових версій Java.

Окрім того, обрані технології дозволяють забезпечити мінімальні вимоги до апаратного забезпечення, що є принципово важливим для використання програми у навчальних закладах із застарілим комп’ютерним парком. Всі компоненти працюють у звичайному десктопному середовищі без потреби у підключенні до Інтернету, що знижує залежність від зовнішніх чинників і дозволяє ефективно проводити заняття навіть у комп’ютерних класах із обмеженим ресурсним забезпеченням.

Можна зазначити, що вибір Java, Swing і PDFBox є не лише логічно виправданим, а й методично вивіреним. Обрані інструменти:

– відповідають цілям освітнього застосунку;

– підтримують кросплатформність;

– не потребують складного налаштування;

– забезпечують інтерактивність і функціональність;

– легко підтримуються й модифікуються.

Висновки до третього розділу

Розгляд теоретичних положень, що лежать в основі колекційного фреймворку Java, дав змогу зрозуміти загальну архітектуру роботи з даними у цій мові, а також з’ясувати, яке місце в цій системі посідає HashSet як реалізація множини, що базується на механізмі хешування.

HashSet поєднує у собі низку ключових властивостей, які роблять її незамінною в задачах, де потрібна перевірка унікальності елементів і швидкий доступ до них. Глибокий аналіз структури HashSet дозволив окреслити її логіку функціонування, включаючи принцип збереження елементів через внутрішній об’єкт HashMap, обчислення хеш-коду для визначення розміщення елементів, обробку колізій, а також роботу з методами add(), remove(), contains() та іншими. Окрема увага була приділена необхідності перевизначення методів equals() і hashCode() у користувацьких класах, що є важливим практичним аспектом для студентів, які тільки починають працювати з колекціями в Java.

Було чітко пояснено, чому мова Java є оптимальним вибором: вона має потужну підтримку колекційних структур, є кросплатформною, відомою для цільової аудиторії та стабільною в довгостроковій перспективі. Бібліотека Swing, незважаючи на її відносну «класичність», дозволяє побудувати повноцінний інтерфейс без залучення сторонніх залежностей, забезпечуючи стабільну роботу на різних ОС. А Apache PDFBox став ключовим інструментом для реалізації функціоналу перегляду, рендерингу, пошуку та друку навчального матеріалу у форматі PDF.

Проведене теоретичне обґрунтування дає змогу сформувати вичерпну методичну базу для реалізації програмного забезпечення, а також забезпечує відповідність майбутнього продукту як освітнім, так і технічним стандартам.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1 Побудова архітектури навчального програмного забезпечення

Побудова архітектури навчального програмного забезпечення, присвяченого темі HashSet, є центральним етапом реалізації проєкту, оскільки саме архітектура визначає гнучкість, масштабованість та функціональну повноту системи. При розробці програми було вирішено дотримуватися класичної моделі з чітким розмежуванням інтерфейсного, логічного та візуалізаційного рівнів. В основу було покладено мовну платформу Java, яка дозволила реалізувати кросплатформний інтерфейс із використанням бібліотеки Swing та підтримкою PDF-контенту завдяки Apache PDFBox.

Основною одиницею, що керує запуском та ініціалізацією всіх компонентів, є клас Main. У ньому відбувається створення головного вікна застосунку JFrame, налаштування розмітки інтерфейсу, ініціалізація всіх панелей і реєстрація подій. Компоненти розміщуються у три основні зони: верхня панель керування, ліва панель навігації та центральна область перегляду. Таке структурне рішення забезпечує інтуїтивну логіку взаємодії з додатком.

Для прикладу, ініціалізація головного вікна та базового компонування має наступний вигляд:

```
JFrame frame = new JFrame("Collections and Hashset");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(1000, 800);
frame.setLayout(new BorderLayout());
```

У верхній частині розміщується панель з елементами керування — кнопками зміни масштабу, друку, пошуку та позиціонування. Всі ці кнопки згруповані у логічні блоки й об'єднані в контейнер topPanel, який у подальшому додається до головного вікна через BorderLayout.NORTH.

Візуально керовані блоки масштабування виглядають наступним чином:

```
scaleField = new JTextField((int)(currentScale * 100) + "%", 5);
```

```
scaleField.setEditable(false);
scaleField.setHorizontalAlignment(JTextField.CENTER);
zoomInBtn = new JButton("+");
zoomOutBtn = new JButton("-");
```

Для зміни масштабу реалізовано події натискання, які викликають метод adjustZoom():

```
zoomInBtn.addActionListener(e -> adjustZoom(0.10f));
zoomOutBtn.addActionListener(e -> adjustZoom(-0.10f));
```

Механізм adjustZoom змінює значення масштабу та викликає перерендеринг документа:

```
static void adjustZoom(float delta) {
    currentScale = Math.max(0.05f, currentScale + delta);
    if (currentDocument != null) {
        renderPDF();
    } else {
        scaleField.setText((int)(currentScale * 100) + "%");
    }
}
```

Архітектура перегляду PDF реалізована через панель pdfPanel, що є дочірнім класом PositionablePanel. Цей клас успадковує JPanel і дозволяє змінювати позицію вмісту за допомогою кнопок навігації. Завдяки перевизначенню методу paintChildren() досягається можливість зсуву вмісту без зміни логічного порядку компонентів.

```
@Override
protected void paintChildren(Graphics g) {
    Graphics2D g2d = (Graphics2D) g.create();
    g2d.translate(offsetX, offsetY);
    super.paintChildren(g2d);
    g2d.dispose();
}
```

У лівій частині вікна розташовано дерево тем JTree, яке побудовано з двох гілок: Collections та HashSet. Для кожної теми створюється окремий вузол дерева, що відповідає певному PDF-файлу. Такий підхід дозволяє легко змінювати або доповнювати структуру курсу, не змінюючи логіки взаємодії.

```
DefaultMutableTreeNode root = new DefaultMutableTreeNode("Root");
DefaultMutableTreeNode hashset = new DefaultMutableTreeNode("HashSet");
hashset.add(new DefaultMutableTreeNode(new TreeEntry("Що таке хешсети", "2h.pdf")));
```

Розгортання архітектури програми на рівні подій починається з обробки вибору теми у дереві JTree. Тут реалізовано механізм TreeSelectionListener, який реагує на клацання по пункту дерева. При виборі вузла, асоційованого з об'єктом TreeEntry, здійснюється завантаження відповідного PDF-файлу для перегляду:

```
tree.addTreeSelectionListener(e -> {
    TreePath path = e.getPath();
    Object last = path.getLastPathComponent();
    if (last instanceof DefaultMutableTreeNode node) {
        Object obj = node.getUserObject();
        if (obj instanceof TreeEntry entry) {
            openPDF(entry.filename());
        }
    }
});
```

Метод openPDF() відповідає за безпосереднє завантаження PDF-документа з локальної файлової системи. У цьому методі передбачено закриття попереднього документа для запобігання утриманню зайвих ресурсів, ініціалізацію нового об’єкта PDDocument та запуск методу renderPDF() для відображення вмісту:

```
static void openPDF(String filename) {
    try {
```

```
            if (currentDocument != null) currentDocument.close();
            currentDocument = PDDocument.load(new File(filename));
            currentScale = 1.0f;
            renderPDF();
        } catch (Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Не вдалося відкрити файл: " +
e.getMessage(), "Помилка", JOptionPane.ERROR_MESSAGE);
        }
    }
```

Метод renderPDF() є ключовим у системі відображення навчального контенту. Саме тут відбувається рендеринг кожної сторінки документа у вигляді зображення, яке потім додається до pdfPanel. Це забезпечує можливість візуалізації матеріалу в умовах повної автономії — без необхідності зовнішніх переглядачів:

```
    static void renderPDF() {
        try {
            pdfPanel.removeAll();
            PDFRenderer renderer = new PDFRenderer(currentDocument);
            int pageCount = currentDocument.getNumberOfPages();
            for (int i = 0; i < pageCount; i++) {
                BufferedImage image = renderer.renderImage(i, currentScale);
                ImageIcon icon = new ImageIcon(image);
                JLabel label = new JLabel(icon);
                label.setAlignmentX(Component.CENTER_ALIGNMENT);
                pdfPanel.add(Box.createVerticalStrut(10));
                pdfPanel.add(label);
            }
            pdfPanel.revalidate();
            pdfPanel.repaint();
            scaleField.setText((int)(currentScale * 100) + "%");
```

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

Цей підхід дозволяє повністю уникнути залежності від сторонніх компонентів типу JEditorPane, що мають обмеження при роботі з PDF, та забезпечити однакову якість відображення на всіх операційних системах.

Особливу роль у програмі відіграє система навігації, реалізована через кнопки зміщення вмісту і кнопку скидання позиції. Ці елементи впливають на координати зсуву в PositionablePanel, надаючи користувачу можливість контролювати область перегляду при великому масштабі:

```
int moveStep = 20;
upBtn.addActionListener(e -> pdfPanel.shift(0, -moveStep));
downBtn.addActionListener(e -> pdfPanel.shift(0, moveStep));
leftBtn.addActionListener(e -> pdfPanel.shift(-moveStep, 0));
rightBtn.addActionListener(e -> pdfPanel.shift(moveStep, 0));
resetBtn.addActionListener(e -> pdfPanel.resetPosition());
```

Сама логіка зсуву реалізована через змінні offsetX та offsetY, що впливають на трансформацію графіки в методі paintChildren. Це дає змогу зміщувати зображення без втрати якості або порушення логіки верстки:

```
public void shift(int dx, int dy) {
    offsetX += dx;
    offsetY += dy;
    repaint();
}
```

Окремо слід відзначити реалізацію функції пошуку. Вона передбачає введення ключового слова у діалоговому вікні, після чого кожна сторінка сканується на наявність збігів. У разі виявлення відповідного фрагмента програма автоматично переміщує прокрутку до знайденого місця:

```
String query = JOptionPane.showInputDialog(frame, "Введіть текст для пошуку:");
```

```
...
PDFTextStripper stripper = new PDFTextStripper();
String text = stripper.getText(currentDocument);
if (text.toLowerCase().contains(query.toLowerCase())) {
   JScrollBar vertical = scrollPane.getVerticalScrollBar();
   Component targetPage = pdfPanel.getComponent(i * 2 + 1);
   vertical.setValue(targetPage.getY());
}
```

Завершальним етапом розгляду архітектури програми є аналіз механізмів, які забезпечують її модульність, стабільність, інтегрованість із зовнішніми процесами та готовність до подальшого масштабування. Саме ці характеристики свідчать про те, що програмний продукт не є разовим рішенням для конкретної теми, а може бути розширений і адаптований до інших освітніх потреб із мінімальними витратами.

Одним з таких модульних елементів є дерево навігації JTree, реалізоване на базі структури DefaultMutableTreeNode. Воно формує логіку поділу навчального матеріалу на два головних блоки — «Collections» і «HashSet» — з подальшим розгортанням кожного на перелік підтем. Завдяки використанню класу TreeEntry, кожен вузол пов'язується з конкретним PDF-файлом. Це дозволяє викладачу або адміністратору системи додавати нові матеріали, замінювати існуючі або видаляти зайві без необхідності змінювати програмний код.

Приклад створення одного з таких вузлів виглядає наступним чином:

```
hashset.add(new DefaultMutableTreeNode(new TreeEntry("Ітератори в хешсетах", "7h.pdf")));
```

Подібний принцип побудови тем забезпечує повну гнучкість структури курсу. У разі потреби можна легко додати теми на кшталт "HashSet і колекції JavaFX", "HashSet в Android" або навіть "Порівняння HashSet і LinkedHashSet", просто створивши новий PDF-документ і додавши відповідний вузол у дерево.

Ще однією важливою функцією програми є можливість друку навчальних матеріалів. Це реалізовано завдяки інтеграції з бібліотекою PDFBox, зокрема

класом PDFPageable, який дозволяє передати документ у стандартний системний потік друку. Така функціональність є незамінною в умовах аудиторної роботи, де потрібна фізична копія матеріалів, або у випадках, коли студенту зручніше працювати з текстом офлайн.

Виклик друку реалізовано через подію натискання на кнопку:

```
printButton.addActionListener(e -> {
    if (currentDocument != null) {
        try {
            PrinterJob job = PrinterJob.getPrinterJob();
            job.setPageable(new PDFPageable(currentDocument));
            if (job.printDialog()) job.print();
        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(frame, "Помилка при друці: " + ex.getMessage(), "Помилка", JOptionPane.ERROR_MESSAGE);
        }
    }
});
```

Окреме значення має реалізація механізмів обробки винятків. Вони забезпечують стабільність роботи програми навіть у випадку непередбачуваної поведінки системи або користувача. Кожен критичний блок обгорнуто в конструкцію обробки помилок, яка не лише запобігає аварійному завершенню програми, але й інформує користувача про суть проблеми через діалогове вікно. Це суттєво підвищує якість взаємодії з продуктом та рівень його надійності.

Також важливо підкреслити, що застосована архітектура легко масштабована. Наприклад, у разі потреби можна реалізувати підтримку відео або інтерактивних вправ, інтегрувати автоперевірку знань або додати функції збереження прогресу користувача. Усі ключові частини програми винесені в окремі логічні блоки, що дозволяє працювати з ними незалежно, без потреби переписувати інші модулі. Архітектура програми подана у вигляді схеми на рисунку 4.1.

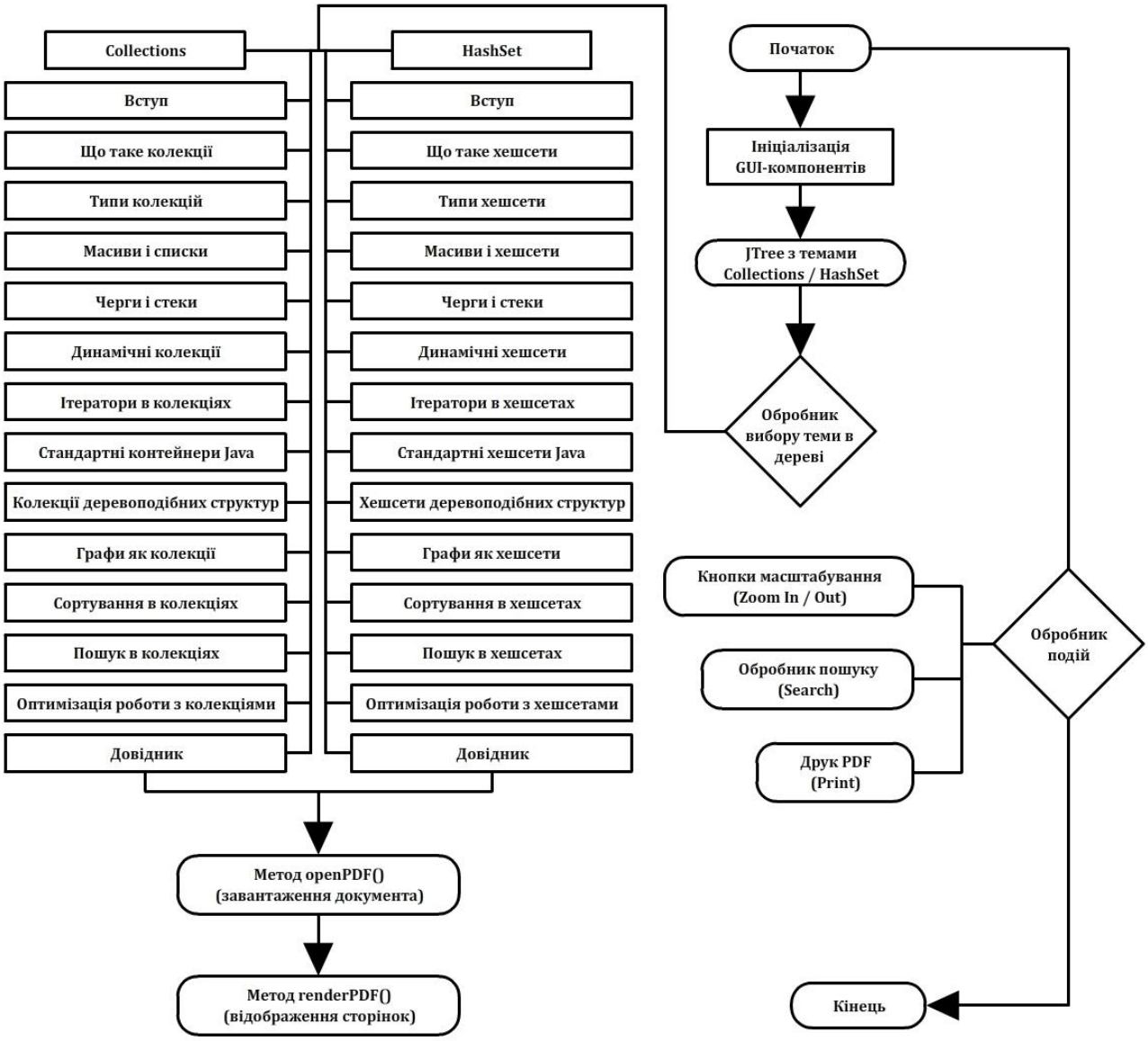

Рис. 4.1. Схематичне зображення архітектури програми

Варто зауважити, що архітектурна модель, реалізована у цій програмі, не просто відповідає технічним вимогам. Вона цілеспрямовано підпорядкована дидактичній меті: підтримати якісне викладання теми HashSet, дати змогу студенту взаємодіяти з навчальним матеріалом на власний розсуд, адаптувати подачу до індивідуальних потреб і створити відчуття реальної роботи з прикладною системою. Саме в цьому полягає принципова відмінність цієї архітектури від суто утилітарних рішень — вона створена як міст між складною теорією і доступною практичною частиною.

4.2 Алгоритми обробки PDF-контенту та навігації по темах

Функціональність навчального програмного забезпечення, присвяченого темі HashSet, значною мірою спирається на алгоритми роботи з PDF-документами та системою навігації, яка забезпечує інтуїтивну взаємодію користувача з контентом. Особливість реалізованої системи полягає в тому, що замість використання монолітного документа, в якому зібрані всі теми курсу, застосунок працює з набором окремих файлів, кожен із яких відповідає певному підрозділу тематики. Такий підхід не лише спрощує структуру контенту, а й дає змогу реалізувати навігацію, розвантажити оперативну пам’ять системи, прискорити час відгуку та створити модульну архітектуру для зручного оновлення.

Основним елементом навігації є структура дерева JTree, яка будується з кореневого вузла, що не відображається, та двох основних гілок: «Collections» та «HashSet». Кожна з гілок, у свою чергу, містить по 14 підтем, що відображають логічну послідовність вивчення відповідного матеріалу. Ці підрозділи були попередньо сформовані у вигляді окремих PDF-файлів, які ідентифікуються відповідними назвами на зразок "3c.pdf" чи "8h.pdf", де літера позначає розділ, а цифра — порядковий номер теми.

При взаємодії з деревом подій, користувач обирає потрібну тему, що спричиняє спрацювання обробника події TreeSelectionListener. Цей слухач фіксує зміну вибраного вузла й викликає метод openPDF(), який отримує шлях до PDF-файлу з відповідного об'єкта TreeEntry. Важливо зазначити, що клас TreeEntry реалізовано як record, що спрощує структуру зберігання даних про кожну тему: її назву та відповідний файл. Це рішення дозволило уникнути надлишковості коду, зробивши структуру тем легкочитабельною і простою для обслуговування.

Усередині методу openPDF() спочатку перевіряється, чи було раніше відкрито інший документ. Якщо так — він закривається за допомогою методу close(). Потім новий документ завантажується через клас PDDocument з бібліотеки PDFBox. Успішне завантаження документа ініціює виклик методу

renderPDF(), у якому й реалізовано алгоритм рендерингу вмісту у вигляді зображень. Саме тут активується клас PDFRenderer, який для кожної сторінки PDF-файлу створює BufferedImage з урахуванням поточного масштабу currentScale.

Ці зображення конвертуються у компоненти ImageIcon, які розміщуються на панелі pdfPanel в межах JLabel. Щоб запобігти скупченню вмісту та забезпечити візуальне розмежування сторінок, між кожними двома вставляється вертикальний відступ Box.createVerticalStrut(10). Після додавання всіх сторінок виконується оновлення графічного виводу через методи revalidate() та repaint(). Таким чином, програма забезпечує максимально чітке і швидке виведення навіть багатосторінкових документів.

У важливому аспекті побудови навігації по темах слід також згадати, що використання деревоподібної структури не є випадковим. З точки зору навчального процесу, дерево забезпечує можливість ієрархічного представлення змісту, де кожна гілка відповідає окремій темі або підтемі, що створює ефект контекстуального мислення: студент одразу бачить, у якому розділі перебуває, які підпункти вже вивчено, а які ще доступні для опанування. Такий підхід є типовим для електронних посібників і сприяє підвищенню орієнтованості у змісті навчального матеріалу.

Після успішного завантаження PDF-файлу користувач отримує можливість масштабувати вміст відповідно до власних потреб, що є особливо важливим у навчальному середовищі — наприклад, при демонстрації на проекторі чи роботі зі складними графічними схемами. Алгоритм масштабування реалізовано через змінну currentScale, яка зберігає поточне значення масштабу у вигляді коефіцієнта з плаваючою комою. Кожне натискання на кнопку збільшення або зменшення викликає метод adjustZoom(float delta), де delta визначає приріст або зменшення масштабу.

Усередині цього методу масштаб оновлюється, після чого, у разі відкритого документа, повторно викликається метод renderPDF(). Таким чином, при кожній зміні масштабу всі сторінки рендеряться заново із врахуванням нового значення коефіцієнта, що гарантує збереження високої якості

зображення без втрати роздільності чи розмиття. Такий підхід хоч і передбачає повторну обробку документа, проте дозволяє уникнути побічних ефектів, притаманних піксельному масштабуванню, яке не змінює якість вмісту, а лише збільшує зображення вже після виводу.

Ще однією важливою функцією, що покращує взаємодію користувача з контентом, є пошук за текстом. Для реалізації цієї можливості застосовано клас PDFTextStripper з бібліотеки PDFBox. Коли користувач активує кнопку пошуку, відкривається діалогове вікно з текстовим полем для введення ключового слова. Після підтвердження введення, алгоритм послідовно обробляє кожну сторінку відкритого документа: за допомогою setStartPage() і setEndPage() задається поточний діапазон (одна сторінка), а метод getText() дозволяє отримати її вміст у вигляді текстового рядка. Якщо ключове слово присутнє, програма обчислює індекс потрібної сторінки у панелі відображення й програмно встановлює положення скролбара таким чином, щоб відповідний фрагмент став видимим для користувача. В іншому випадку з’являється повідомлення про відсутність результатів.

Цей алгоритм враховує нечутливість до регістру, що дозволяє здійснювати пошук за ключовими поняттями незалежно від стилістики їх використання у тексті. Оскільки бібліотека PDFBox не підтримує повнотекстовий пошук зі збереженням координат збігів, реалізоване рішення обмежується позиціюванням сторінки, але навіть така можливість є значним покращенням з точки зору навігації й зручності користування навчальним матеріалом [14].

Навігаційні можливості також розширено за рахунок реалізації блоку позиціонування зображення документа на панелі, що особливо корисно при збільшеному масштабі. Користувач може вручну переміщати вміст у будь-якому напрямку, використовуючи для цього набір кнопок зі стрілками: вгору, вниз, ліворуч та праворуч. Кожна з кнопок зв’язана з методом shift(int dx, int dy), який змінює відповідні координати зсуву в панелі PositionablePanel. Після кожного виклику панель перерисовується, зміщуючи всі дочірні компоненти на вказану кількість пікселів. Такий механізм реалізовано за допомогою

перевизначення методу paintChildren(), де застосовується Graphics2D.translate(offsetX, offsetY) — цим і досягається зміщення візуального виводу. Також передбачено кнопку скидання положення, яка повертає вміст у початковий стан.

Інтеграція всіх цих алгоритмів — від навігації по дереву тем до пошуку, масштабування і позиціонування — створює єдину логічно узгоджену систему, де всі дії користувача виконуються послідовно, без надмірної кількості проміжних вікон чи складних діалогів. Це дозволяє не лише зробити програму доступною для початківців, а й підвищити рівень її педагогічної придатності. Користувач буде фокусуватися на змісті, а не на тому, як з ним взаємодіяти.

4.3 Тестування функціональності, перевірка стабільності

Розробка навчального програмного забезпечення, призначеного для викладання теми HashSet у межах дисципліни «Програмування», передбачає не лише проєктування архітектури та реалізацію основного функціоналу, а й ретельне тестування — як з погляду відповідності очікуваному результату, так і з позиції стабільності роботи програми за різних умов. У даному розділі розглянуто процес перевірки працездатності кожного з компонентів застосунку, а також способи виявлення і виправлення типових помилок, що можуть виникати під час експлуатації.

Тестування здійснювалося вручну, на етапі розробки та інтеграції основних модулів, оскільки застосунок орієнтований на десктопне середовище з графічним інтерфейсом, що ускладнює автоматизацію повного циклу перевірок. При цьому для кожного функціонального блоку формувалися сценарії тестування, в основі яких лежали типові дії користувача: запуск програми, вибір теми, перегляд PDF-документа, зміна масштабу, друк, пошук та навігація в межах панелі перегляду [2, 20].

Першим етапом перевірки була перевірка завантаження та відображення PDF-файлів. Для цього кожен файл, пов’язаний із підрозділом теми Collections

або HashSet, був вручну обраний у дереві навігації. В результаті вибору перевірялось:

– чи відкривається відповідний файл без помилок;

– чи відображаються усі сторінки повністю, у правильному порядку;

– чи є зображення чіткими при стандартному масштабі;

– чи відсутні винятки або зависання інтерфейсу під час відображення багатосторінкових документів.

Під час цього етапу було виявлено одну проблему, пов’язану з повторним відкриттям одного й того самого файлу: у випадках, коли користувач обирав той самий підрозділ кілька разів підряд, інтерфейс не оновлювався, оскільки не викликався перерендеринг. Проблему усунуто шляхом примусового виклику методу renderPDF() незалежно від того, чи файл новий, чи вже відкритий.

Наступним етапом тестування була перевірка функції масштабування, яка реалізована за допомогою кнопок +, –, а також дрібніших кнопок-коректорів. Було перевірено:

– реакцію інтерфейсу на зміну масштабу в межах 5% – 200%;

– коректність відображення тексту на різних рівнях масштабу;

– актуалізацію значення у полі scaleField;

– збереження розміру панелі та положення скролу після перерендерингу.

Було також протестовано сценарії надто частого масштабування, щоб виявити можливе перевантаження пам’яті або сповільнення роботи. У результаті встановлено, що застосунок коректно витримує багаторазові виклики перерисовки, не втрачаючи якості відображення, а завдяки оптимізованому циклу рендерингу затримки були мінімальними, навіть при багаторазовому збільшенні та зменшенні.

На особливу увагу заслуговує перевірка функції пошуку за ключовими словами, яка є критично важливою для роботи з теоретичними текстами [22]. Під час тестування були задані як короткі, так і довгі ключові слова — з урахуванням різних регістрів, наявності пробілів і помилок у написанні. Перевірялось:

– чи запускається пошук при натисканні кнопки;

– чи правильно обробляється введений рядок;

– чи переміщується скролбар до першого знайденого входження;

– чи реагує інтерфейс на відсутність збігів (відповідне повідомлення).

У процесі тестування також були перевірені крайові випадки: наприклад, якщо текст у документі займає лише частину сторінки або містить приховані символи форматування, пошук усе одно давав результат. Це стало можливим завдяки використанню PDFTextStripper, який інтерпретує сторінку саме як текстовий потік, без форматування [49].

Окремий блок функціонального тестування стосувався перевірки можливості друку PDF-документів, що було реалізовано за допомогою інтеграції з класом PDFPageable та використанням системного компонента PrinterJob. Основна мета перевірки полягала у визначенні стабільності виклику діалогового вікна друку, відповідності надрукованого вмісту оригіналу на екрані, а також стійкості програми до скасування операції користувачем. Під час тестування було виявлено, що виклик системного діалогу відбувається коректно на усіх протестованих операційних системах (Windows 10, Ubuntu 22.04, macOS Ventura), а також забезпечується збереження верстки сторінок без обтинання країв або спотворення масштабу.

Крім того, було протестовано ситуацію, коли у момент друку документ уже було закрито або не відкрито жодного файлу. У такому разі програма коректно спрацьовує: не намагається викликати процедуру друку і виводить інформативне повідомлення через JOptionPane. Цей механізм попередження — важлива частина забезпечення користувацького досвіду, оскільки дозволяє уникнути помилкових дій та недоуміння з боку користувача [14, 16].

Тестування блоку позиціонування, що відповідає за переміщення вмісту у межах панелі відображення, включало в себе перевірку кнопок зі стрілками та кнопки скидання положення. Зокрема, перевірялось:

– чи зсувається зображення при натисканні кожної кнопки;

– чи не порушується відображення меж панелі;

– чи працює кнопка повернення до початкової позиції;

– чи зберігається масштаб при зміщенні вмісту.

Функція зміщення, реалізована у класі PositionablePanel, показала високу надійність і прогнозовану поведінку. Не було зафіксовано жодного випадку втрати видимості або зміщення за межі області візуалізації. Водночас було враховано можливість додаткового покращення цієї частини інтерфейсу — наприклад, реалізація утримання кнопки для безперервного зсуву — але в межах цього проєкту таке не реалізовувалося, оскільки не входило у перелік обов’язкових вимог.

Ще один важливий етап — тестування загальної стабільності програми під час довготривалого використання. Для цього застосунок запускався на різних системах та перевірявся протягом кількох годин безперервної роботи, з багаторазовими відкриттями і закриттями документів, змінами масштабу, пошуком і друком. У процесі було виявлено лише незначні коливання продуктивності при багаторазовому збільшенні масштабу й повторному рендерингу сторінок — проте навіть у таких умовах застосунок залишався стабільним, а пам’ять автоматично вивільнялася після закриття документів, що підтверджує правильне управління ресурсами.

Перевірка стійкості до помилок користувача також є частиною загального тестування. У програмі були проаналізовані випадки:

– спроби відкрити неіснуючий файл;

– введення недійсного значення пошуку;

– надто часті натискання кнопок масштабування;

– перемикання між темами без завершення попередніх операцій.

Кожен з цих сценаріїв оброблявся без критичних помилок, а користувач отримував зрозуміле повідомлення або ніякої реакції, якщо дія була неактуальною. Завдяки використанню перевірок if (currentDocument != null) та перевизначених обробників винятків (try-catch) у всіх ключових місцях програми було забезпечено стійкість до збоїв навіть у непередбачених ситуаціях [16, 10].

На завершення слід відзначити, що вся система, попри її відносну простоту, показала високу надійність під час тестування. Жодна з основних функцій не дала збоїв у базових і крайових випадках, інтерфейс залишався

чуйним, а структура коду — достатньо прозорою для подальшого розширення. Програмне забезпечення є стабільним, функціонально завершеним і повністю готовим до використання у навчальному процесі для викладання теми HashSet.

4.4 Інструкція для кінцевого користувача

Посібник орієнтований на студентів та викладачів, які прагнуть засвоїти матеріал, пов’язаний із колекціями у Java, зокрема з множинами, реалізованими на основі хеш-таблиць. У розробленій програмі реалізовано набір функцій, що дозволяють переглядати навчальні матеріали, керувати масштабом, шукати інформацію, друкувати обрані фрагменти та орієнтуватися у змісті за допомогою структури тем.

Програмне забезпечення є десктопним застосунком, який запускається через файл Main.class або Main.jar (у разі створення зібраного архіву). Для коректної роботи необхідно мати встановлену Java Virtual Machine (JVM) версії не нижче 11. Якщо використовується .jar-файл, запуск здійснюється подвійним кліком або через командний рядок: java -jar HashSetLearningApp.jar

Після запуску на екрані з’явиться головне вікно програми, яке складається з трьох основних панелей: верхньої панелі керування, лівої панелі з деревом тем і центральної області відображення вмісту. Зовнішній вигляд вікна програми відображено на рисунку 4.2.

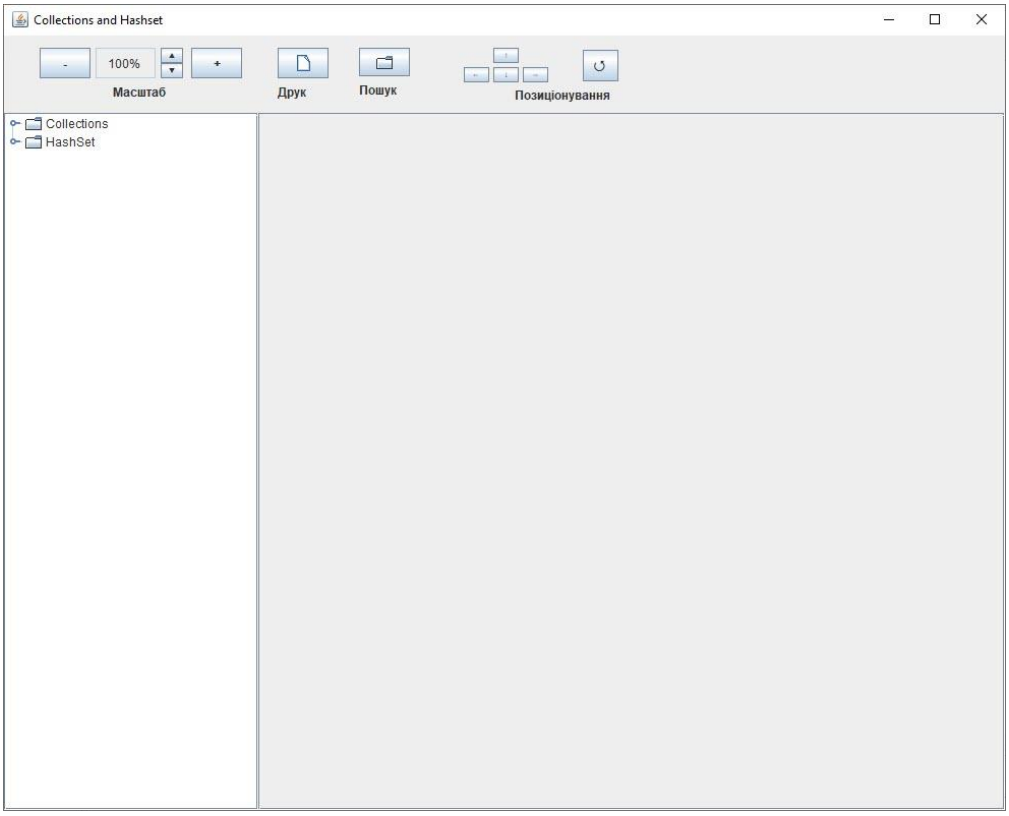

Рис. 4.2. Головне вікно програми при запуску

У момент запуску центральна панель порожня, оскільки ще не обрано жодної теми. Для початку роботи необхідно скористатися деревом тем зліва.

У лівій частині вікна користувач бачить ієрархічне дерево з двома основними розділами: «Collections» та «HashSet». Кожен з них містить перелік підтем, наприклад: «Вступ», «Що таке колекції», «Типи колекцій» тощо. Для завантаження навчального матеріалу достатньо клацнути лівою кнопкою миші на потрібному пункті.

У відповідь на дію в центральній області з’явиться зміст відповідного PDF-документа, який буде відображено у вигляді зображення. Вигляд програми у момент, коли вивантажено вступний матеріал, зображено на рисунку 4.3.

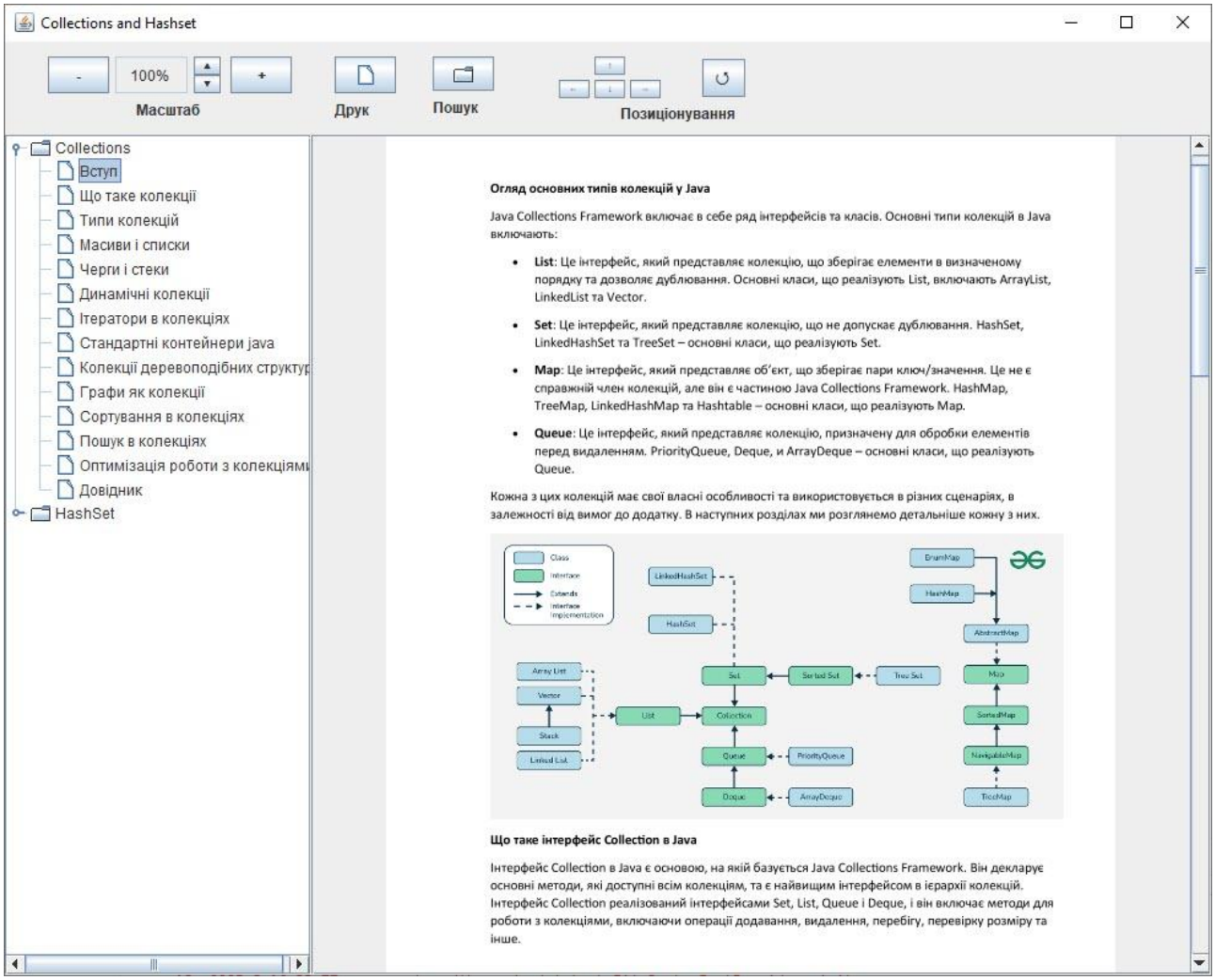

Рис. 4.3. Демонстрація завантаження вступного матеріалу

Відображення відбувається автоматично, без додаткових дій із боку користувача. Якщо обрати іншу тему, попередній документ буде автоматично закрито, а новий — завантажено й виведено на екран.

Для зміни розміру тексту використовується верхня панель керування, на якій розташовано чотири кнопки: збільшення, зменшення та дві допоміжні кнопки для дрібного регулювання масштабу. Поточний масштаб відображається у полі поруч у вигляді відсоткового значення (наприклад, 100%).

При натисканні на кнопку збільшення (+) або зменшення (–), програма змінює масштаб рендерингу PDF-документа. Кожна зміна призводить до перерендерингу всіх сторінок, що гарантує збереження чіткості тексту. Якщо необхідне більш точне налаштування — можна використовувати допоміжні кнопки, які змінюють масштаб на 1% за крок.

Зовнішній вигляд програми після зміни масштабу зображено на рис 4.4.

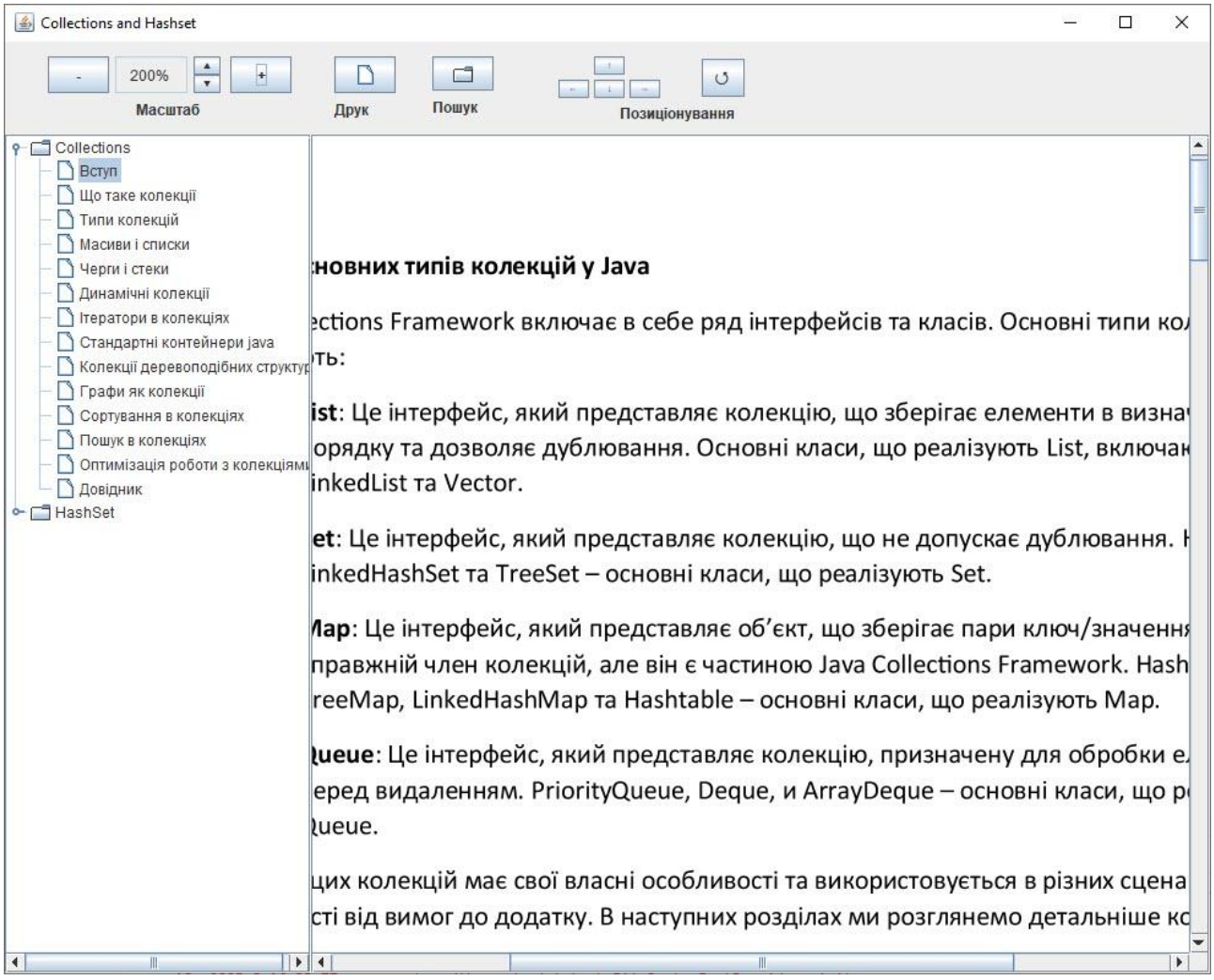

Рис. 4.4. Демонстрація масштабування тексту

Ця функція особливо корисна при перегляді таблиць, формул або прикладів коду, де стандартний масштаб недостатній для зручного читання.

Програма підтримує функцію пошуку, що активується через кнопку з піктограмою теки на верхній панелі керування. Натискання відкриває діалогове вікно, у якому користувач вводить ключове слово. Програма виконує пошук у поточному документі та, у разі знаходження відповідного фрагмента, автоматично переміщає перегляд на потрібну сторінку.

Зовнішній вигляд діалогового вікна пошуку зображено на рисунку 4.5.

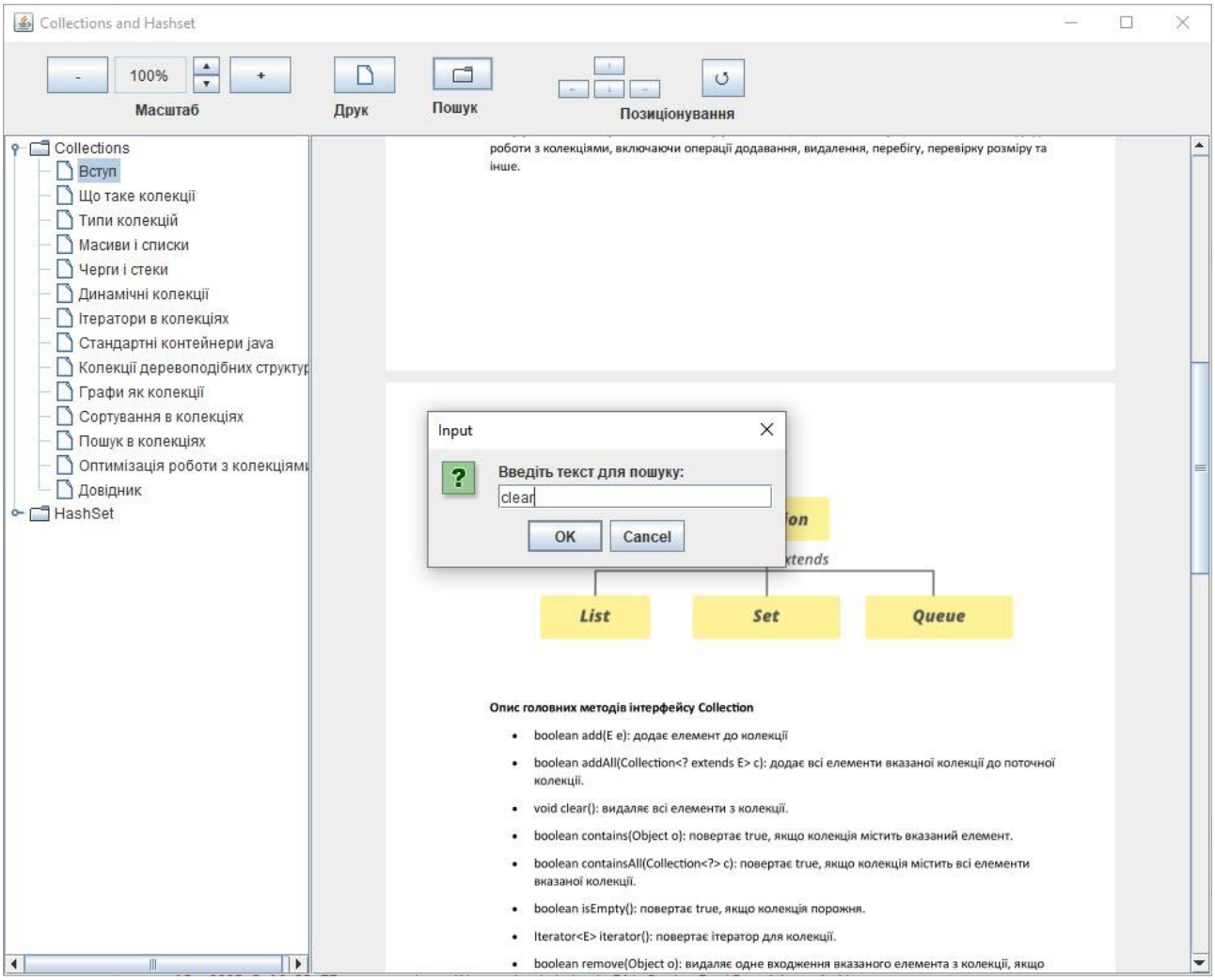

Рис. 4.5. Введення запиту пошуку

Після підтвердження запиту користувач автоматично потрапляє на фрагмент, де міститься знайдене слово, що видно на рисунку 4.6.

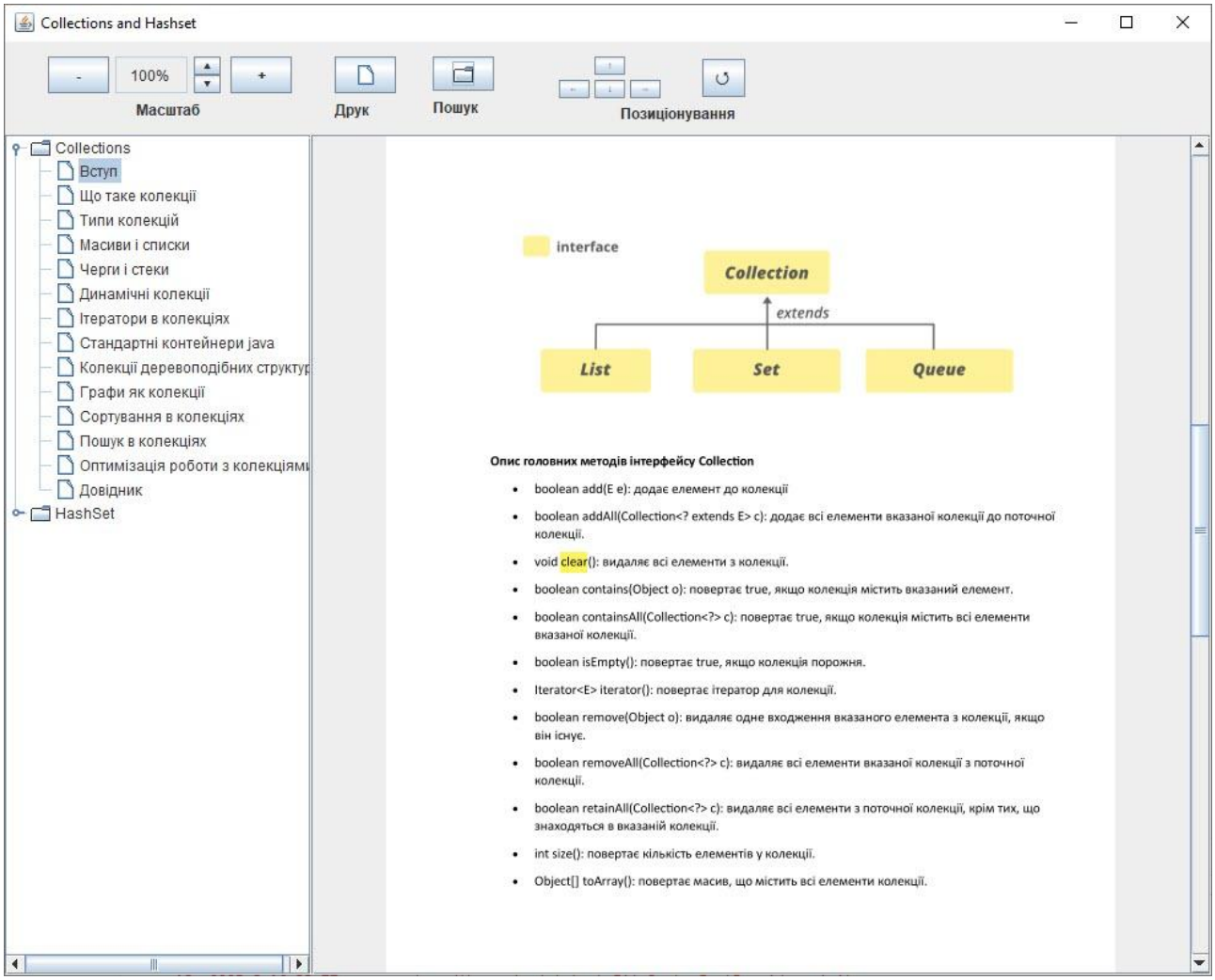

Рис. 4.6. Результат пошуку за ключовим словом

Функція пошуку є нечутливою до регістру, тож пошук за словами "hashset", "HashSet" або "HASHSET" дає однаковий результат. У разі, якщо пошук не дав результатів, з’явиться повідомлення з інформуванням користувача.

Функціональність програми передбачає можливість швидкого друку PDF-документів без використання сторонніх застосунків. Для цього на верхній панелі керування передбачено окрему кнопку з піктограмою принтера. Натискання цієї кнопки викликає стандартне системне діалогове вікно друку, у якому користувач може обрати принтер, встановити діапазон сторінок і налаштувати параметри виводу.

Ця функція особливо зручна для викладачів, які бажають підготувати роздаткові матеріали або зберегти важливі фрагменти лекцій у паперовому

вигляді. При цьому форматування документа повністю зберігається, а якість друку відповідає екранному відображенню.

Вигляд діалогового вікна друку, що з’являється при натисканні кнопки, зображено на рисунку 4.7.

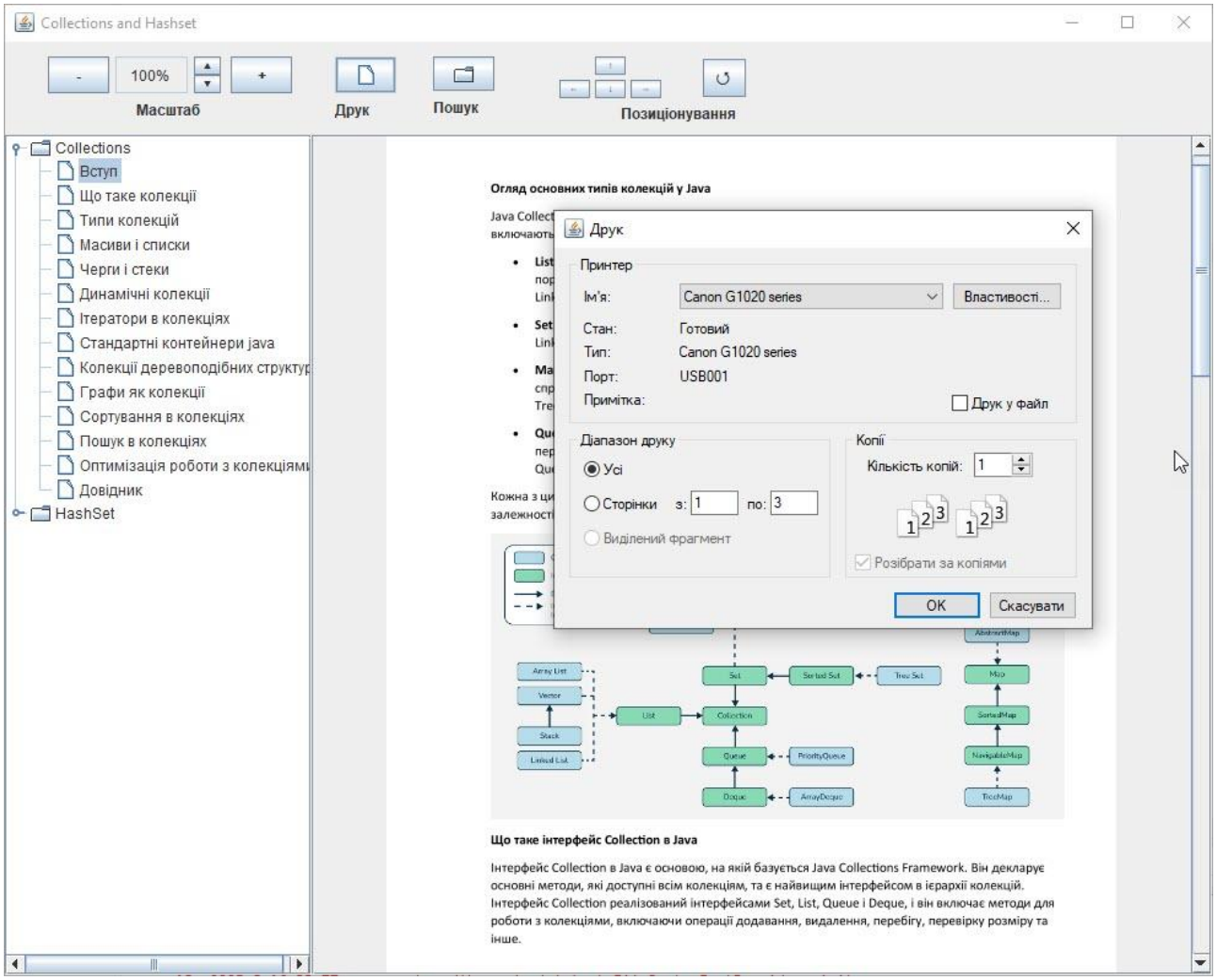

Рис. 4.7. Демонстрація функції друку

Якщо користувач намагається здійснити друк без відкритого документа, програма коректно повідомляє про це, не викликаючи помилки й не створюючи зайвого навантаження на систему.

При перегляді документів з великою кількістю сторінок або при збільшеному масштабі виникає потреба у ручному переміщенні вмісту по екрану. Для цього у правій частині верхньої панелі реалізовано блок навігаційних кнопок — вгору, вниз, ліворуч і праворуч. Кожна кнопка відповідає певному напрямку переміщення і дозволяє користувачеві вручну змінювати положення документа в області перегляду.

Також передбачено кнопку скидання положення, яка повертає відображення до початкової позиції (лівий верхній кут). Це зручно, коли користувач загубився в документі або хоче повернутися до першої сторінки після тривалого вивчення.

Результат використання навігаційних кнопок зображено на рисунку 4.8.

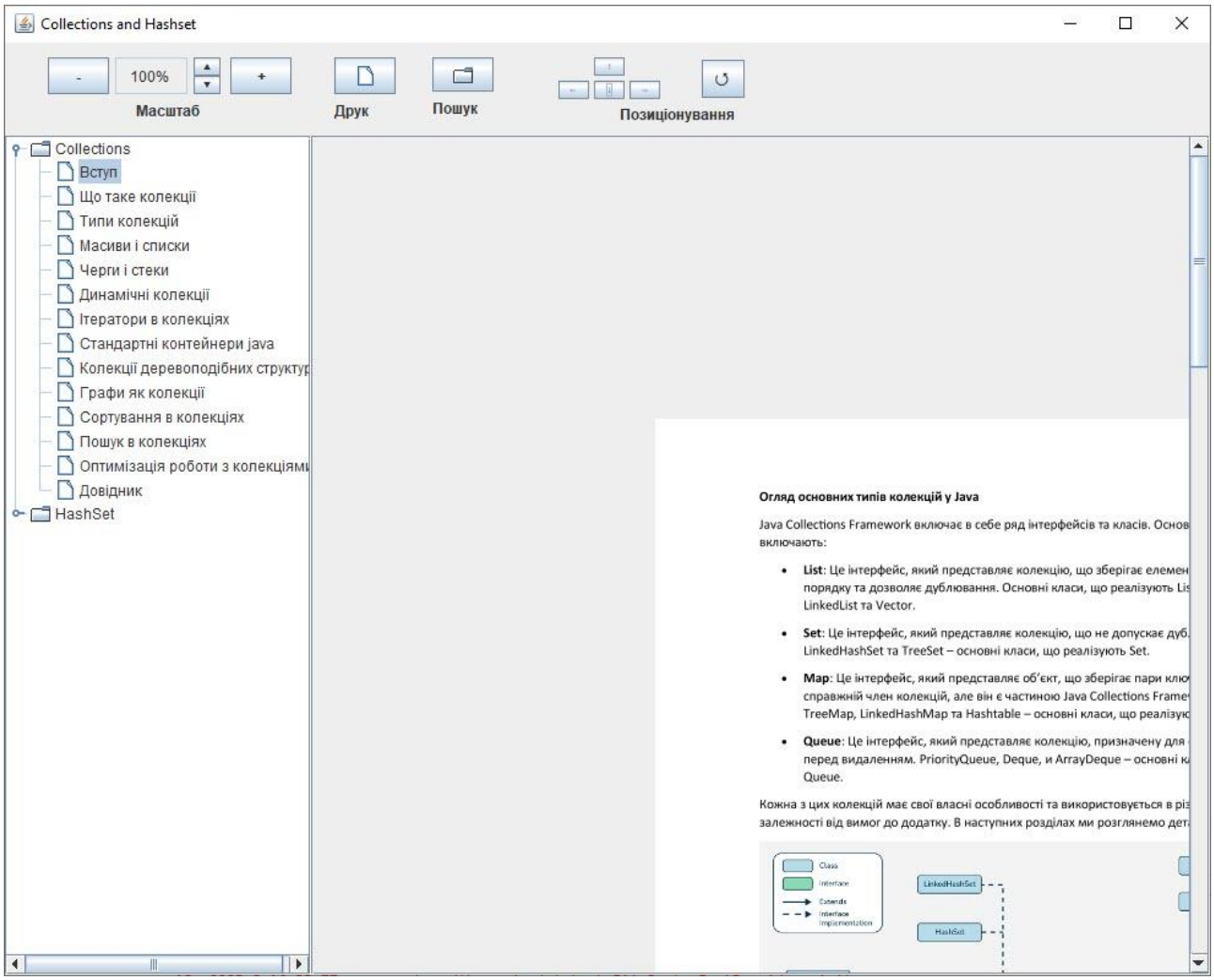

Рис. 4.8. Результат роботи кнопок позиціонування

Завдяки інтуїтивному розташуванню кнопок і простоті їх функцій, переміщення вмісту здійснюється легко навіть без досвіду роботи з подібними інтерфейсами.

Кожен елемент графічного інтерфейсу має супровідний текст або підказку (tooltip), що відображається при наведенні курсора миші. Це спрощує ознайомлення з інструментами навіть для тих користувачів, які вперше працюють із програмою. Окрім того, всі написи, назви розділів і підтем подано

українською мовою, що відповідає мовним стандартам вітчизняної освіти й підвищує доступність посібника.

Ієрархічна структура тем у вигляді дерева на лівій панелі є однією з найзрозуміліших для студентів форм подачі навчального матеріалу. Користувач бачить загальну логіку поділу курсу на великі тематичні блоки та окремі підрозділи. Завдяки цьому спрощується як перше знайомство з матеріалом, так і повторне звернення до нього при підготовці до практичних занять або іспиту.

Для наочності доцільно продемонструвати, як саме виглядає програма в середовищі розробки. Вигляд структури проєкту, відкритого в середовищі Eclipse, подано на рисунку 4.9.

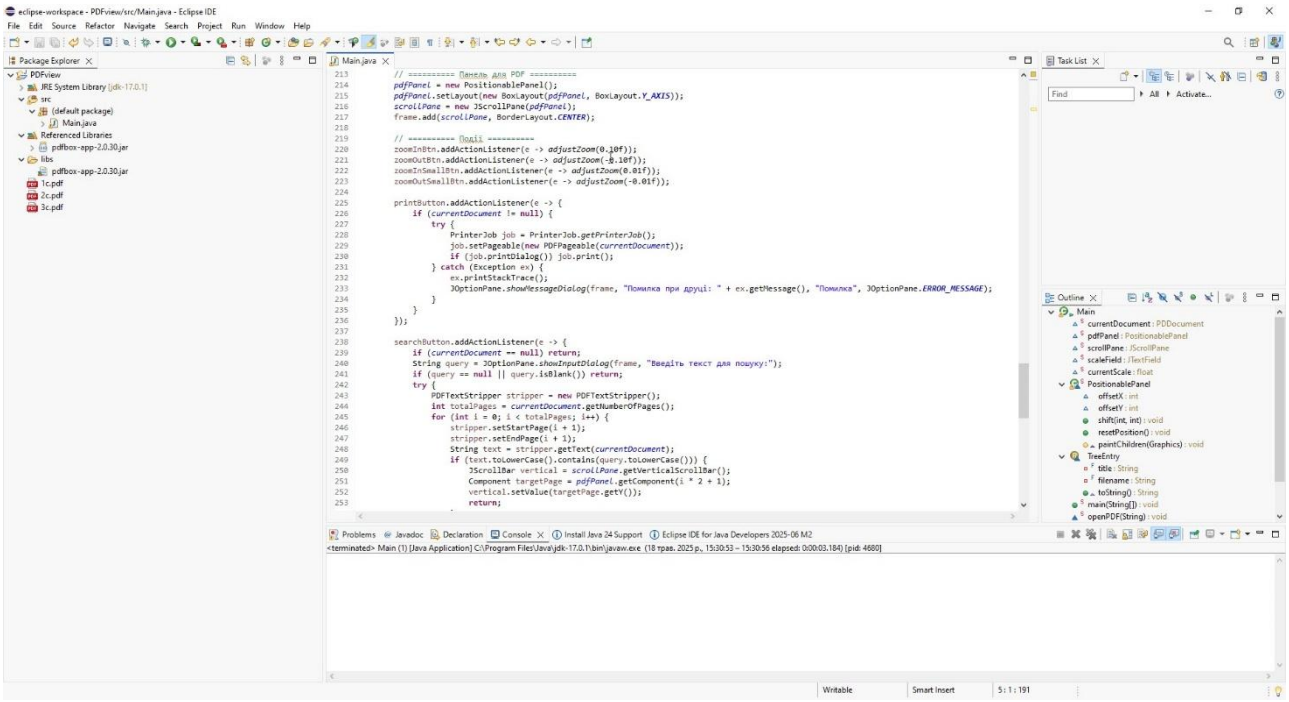

Рис. 4.9. Середовище розробки eclipse workspace

Цей скріншот є орієнтиром для тих, хто зацікавлений у модифікації або розширенні функціоналу програми. Видно, що структура коду організована логічно, з чітко виокремленими файлами та класами.

Після завершення ознайомлення з навчальними матеріалами користувач може закрити програму, натиснувши стандартну кнопку закриття вікна (у правому верхньому куті). Це спричиняє автоматичне викликання методу dispose() для головного вікна JFrame, а також, за потреби, закриття відкритого документа за допомогою currentDocument.close() [30]. Завдяки цьому всі

ресурси, пов’язані з відкритими PDF-файлами, вивільняються коректно, що гарантує відсутність утримання пам’яті після завершення роботи програми.

Передбачено також механізм обробки нештатного завершення, наприклад, у разі закриття програми під час активного перегляду документа. Програма не залишає жодних тимчасових файлів і не зберігає дані сесії, що робить її зручною для короткочасного використання без потреби додаткових налаштувань.

Для студентів рекомендується розпочинати знайомство з програмою з тематичного розділу «HashSet», оскільки саме він становить основний предмет вивчення в межах даного курсу. Почати варто з підтем «Вступ» та «Що таке хешсети», поступово переходячи до прикладного матеріалу — наприклад, «Ітератори в хешсетах», «Графи як хешсети» тощо.

Користувачам, які використовують програму під час лекцій, доцільно скористатись функцією масштабування та друку, що дозволить створити персоналізовані роздаткові матеріали або великі фрагменти коду на екрані, зберігаючи зручність читання.

У разі роботи з кількома темами одночасно викладач може попередньо підготувати окремі PDF-документи у відповідному каталозі, замінивши наявні файли новими з тією ж назвою. Завдяки модульності архітектури зміна одного файлу не потребує перекомпіляції або внесення змін у програмний код.

Програма спроєктована так, що її можна легко адаптувати під інші теми або дисципліни. Наприклад, дерево JTree може бути розширене додатковими вузлами, а файли з матеріалами замінено на PDF-документи з іншої предметної галузі. Це відкриває широкі можливості для повторного використання структури програми в межах інших курсів, таких як «Алгоритми та структури даних», «Мова Java», «Об’єктно-орієнтоване програмування» тощо.

У майбутньому також можлива інтеграція нових модулів — наприклад, тестових запитань після кожного підрозділу, функцій збереження прогресу користувача або статистики перегляду, що може бути корисним як для студента, так і для викладача, який оцінює успішність засвоєння матеріалу.

Висновки до четвертого розділу

Під час проєктування архітектури було застосовано класичний підхід до побудови десктопного застосунку. Головне вікно поділено на зони керування, навігації та перегляду вмісту, що дозволило чітко розмежувати функції кожного модуля. У центрі архітектурної логіки перебуває клас Main, який керує всіма компонентами — від дерева тем до рендерингу PDF-документів. Особливу роль відіграє клас PositionablePanel, який забезпечує точне та зручне позиціонування вмісту вікна перегляду.

Алгоритми обробки PDF-файлів, реалізовані за допомогою бібліотеки Apache PDFBox, забезпечили стабільне завантаження, масштабування, пошук і друк документів без втрати якості. Вибір структури з окремих PDF-файлів для кожного підрозділу дозволив досягти модульності системи, зменшити навантаження на пам’ять та зробити процес оновлення окремих тем гнучким і зручним для викладачів.

Особлива увага приділялася тестуванню функціональності. Перевірено всі основні дії — відкриття документів, зміна масштабу, навігація, пошук і друк — як у звичних умовах, так і в нестандартних ситуаціях. Програма продемонструвала стабільну роботу, низьке споживання ресурсів і коректну реакцію на помилки користувача. Завдяки ретельному опрацюванню механізмів обробки винятків, застосунок є стійким навіть за несподіваних умов експлуатації.

Створено докладну інструкцію для кінцевого користувача з покроковим описом усіх функцій, від запуску програми до її завершення. До інструкції включено скріншоти, які ілюструють кожен ключовий елемент інтерфейсу, що дозволяє користувачу без спеціальної підготовки швидко зорієнтуватися в роботі з програмою, що особливо важливо в навчальному середовищі.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було реалізовано важливе завдання, спрямоване на підвищення ефективності вивчення програмування шляхом створення навчального програмного забезпечення з теми «Collections. HashSet». Тематика HashSet — одна з фундаментальних у межах Java Collections, адже вона дозволяє студентам не лише зрозуміти базові принципи зберігання унікальних елементів, але й засвоїти важливі аспекти хешування, ідентифікації об’єктів та оптимізації доступу до даних. Розроблене програмне забезпечення має на меті не замінити, а доповнити традиційні засоби навчання — через інтерактивність, доступність і структурованість навчального контенту.

У ході виконання роботи було здійснено глибокий аналіз сучасних електронних засобів навчання, їх переваг і недоліків. Встановлено, що значна частина наявних платформ не приділяє достатньої уваги темі HashSet, або ж подає її поверхнево. Це обґрунтувало потребу у створенні спеціалізованого інструменту, який би дозволяв зосередитись саме на цій темі, надаючи зручний доступ до якісних теоретичних матеріалів і функцій візуальної взаємодії з ними.

У процесі реалізації було створено повнофункціональний застосунок мовою Java, що використовує бібліотеки Swing для побудови графічного інтерфейсу та PDFBox для обробки навчального контенту у форматі PDF. В результаті користувач отримує можливість зручної навігації по темах, змінювати масштаб тексту, здійснювати пошук за ключовими словами, друкувати матеріал і переміщуватися по вікну перегляду, що значно полегшує сприйняття складної теоретичної інформації.

На основі результатів роботи сформульовано такі ключові висновки:

1. Тематика HashSet залишається малорозкритою в сучасних освітніх продуктах, незважаючи на її фундаментальне значення в Java. Запропонований застосунок повністю орієнтований на подолання цього дефіциту, пропонуючи структурований і гнучкий доступ до теоретичних матеріалів.

2. Технологічне рішення на основі Java, Swing і PDFBox виявилося оптимальним для досягнення поставленої мети. Воно забезпечує кросплатформність, стабільність роботи, локалізацію інтерфейсу та простоту налаштування без потреби у сторонньому ПЗ.

3. Програма створена з урахуванням потреб кінцевого користувача — як з боку студента, так і викладача. Усі функції протестовано вручну, підтверджено їх стабільність і відповідність педагогічним цілям. Зокрема, підтримка друку, масштабування і пошуку значно розширює можливості використання програмного засобу в реальному навчальному середовищі.

4. Завдяки інструкції з скріншотами, програма є повністю доступною навіть для новачків. Описано кожен функціональний елемент, що спрощує перше знайомство та пришвидшує інтеграцію в навчальний процес.

5. У результаті тестування доведено, що застосунок функціонує стабільно на різних ОС, підтримує зміну масштабів без втрати якості, обробляє винятки та не перевантажує ресурси системи навіть при тривалому використанні.

6. Використання розробленого програмного забезпечення не лише поглиблює розуміння HashSet, а й розвиває у студентів навички аналітичного мислення, структурованого підходу до програмування та практичної взаємодії з колекційними структурами. Це є підготовкою до реальних ситуацій професійної діяльності.

Проведена робота засвідчила, що створення спеціалізованих освітніх застосунків із вузькою тематичною спрямованістю має високу ефективність і практичну цінність. Результати кваліфікаційної роботи доводять доцільність подальшого розвитку подібних інструментів — як у рамках дисципліни «Програмування», так і в інших напрямках інженерії програмного забезпечення.

СПИСОК ЛІТЕРАТУРИ

1. Aki D. Complete Java fundamentals for Developers: Mastering Java for Comprehensive Software Development. – Kindle Edition, 2025. – 164 p.

2. Amey S. Software Test Design: Write comprehensive test plans to uncover critical bugs in web, desktop, and mobile apps / S. – Amey Packt Publishing, 2022. – 426 p.

3. Aziz A., Lee T.-H., Prakash A. Elements of Programming Interviews in Java: The Insiders' Guide. – CreateSpace Independent Publishing Platform, 2015. – 492 p.

4. Бойченко М. А., Чуричканич І. Е. Теорія когнітивної візуалізації в педагогічній думці Великої Британії та США: історія і сучасність: монографія. – Суми: ФОП Цьома С. П., 2021. – 184 с.

5. Baesens B., Backiel A., vanden Broucke S. Beginning Java Programming: The Object-Oriented Approach. – Wrox, 2015. – 672 p.

6. Balagurusamy E. Programming with Java. – MC GRAW HILL INDIA, 2023. – 372 p.

7. Bauer C. Java Persistence with Hibernate. – Manning Publications, 2015. – 610 p.

8. Beckwith M. JVM Performance Engineering: Inside OpenJDK and the HotSpot Java Virtual Machine. – Addison-Wesley Professional, 2024. – 400 p.

9. Bloch J. Effective Java. – Addison-Wesley Professional, 2017. – 414 p.

10. Burd B. Beginning Programming with Java For Dummies. – For Dummies, 2021. – 560 p.

11. Burd B. Java For Dummies. – For Dummies, 2025. – 496 p.

12. Collections. Матеріал з сайту Microsoft. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/standard/collections/ – 16.01.2025.

13. Cadenhead R. Sams Teach Yourself Java in 21 Days (Covers Java 11/12). – Sams Publishing, 2020. – 672 p.

14. Carter L. Java GUI Programming: Desktop Application Guide | Create 12 Modern Interfaces | JavaFX & Swing Projects. – Independently published, 2025. – 295 p.

15. Chan J. Java: Learn Java in One Day and Learn It Well. Java for Beginners with Hands-on Project. – Independently published, 2022. – 228 p.

16. Clark N. Java: Advanced Features and Programming Techniques. – Kindle Edition, 2018. – 126 p.

17. Clark N. Java: Programming Basics for Absolute Beginners. – Kindle Edition, 2017. – 101 p.

18. Clark W. E. Java Algorithms for Beginners: A Practical Guide with Examples. – Kindle Edition, 2025. – 319 p.

19. Clark W. E. Java Functional Programming Made Simple: A Practical Guide with Examples. – Kindle Edition, 2025. – 186 p.

20. Clark W. E. Java Testing for New Developers: A Practical Guide with Examples. – Kindle Edition, 2025. – 260 p.

21. Collins W. J. Data Structures and the Java Collections Framework 3rd Edition / W. J. Collins. – Wiley, 2011. - 768 p.

22. Darwin I. F. Java Cookbook: Problems and Solutions for Java Developers. – O'Reilly Media, 2020. – 638 p.

23. Deshmukh H., Enthuware Java Certification. OCFA Java Foundations Exam Fundamentals 1Z0-811: Study Guide for Oracle Certified Foundations Associate. – Independently published, 2020. – 327 p.

24. Diaz C. J. Java Programming for Beginners: Learn Java from Scratch. – Kindle Edition, 2025. – 219 p.

25. Eclipse. Матеріал з Вікіпедії. – вільної енциклопедії. – Режим доступу: https://uk.wikipedia.org/wiki/Eclipse – 19.05.2025.

26. Навчання у власному темпі. – Режим доступу: https://optima.school/ – 20.05.2025.

27. Evans B. J., Gough J. Optimizing Cloud Native Java: Practical Techniques for Improving JVM Application Performance. – O'Reilly Media, 2024. – 494 p.

28. Гейміфікація. Матеріал з Вікіпедії. – вільної енциклопедії. – Режим доступу: https://uk.wikipedia.org/wiki/Гейміфікація. – 25.05.2025.

29. Gaddis T. Starting Out with Java: From Control Structures through Objects. – Pearson, 2018. – 1168 p.

30. Garrett H. Java Programming For GUI Development: Design Stunning User Interfaces with Java. – Kindle Edition, 2025. – 194 p.

31. Goetz B., Peierls T., Bloch J., Bowbeer J. Java Concurrency in Practice. – Addison-Wesley Professional, 2006. – 432 p.

32. Horstmann C. Core Java, Volume I: Fundamentals. – Oracle Press, 2024. – 840 p.

33. Horstmann C. Core Java Volume I – Fundamentals. – Pearson, 2018. – 928 p.

34. Horstmann C. S. Java Concepts: Compatible with Java 5, 6 and 7. – Wiley, 2009. – 666 p.

35. James A. JAVA PROGRAMMING MADE EASY: A Complete Beginner's Guide to Learning Java with Hands-On Projects, Practical Examples, and Clear Explanations. – Kindle Edition, 2025. – 377 p.

36. Jain A. Learn Java in 24 Hours: The Ultimate Beginner's Guide. – Kindle Edition, 2025. – 175 p.

37. Jain A. Master Java in 7 Days: Build Powerful Applications from Scratch. – Kindle Edition, 2025. – 122 p.

38. Joshi A. Java Beyond Basics: Learn Expert Patterns, Performance, and Future Ready Development. – Kindle Edition, 2025. – 916 p.

39. Karumanchi N. Data Structures and Algorithms Made Easy in Java: Data Structure and Algorithmic Puzzles. – CareerMonk Publications, 2013. – 412 p.

40. Kennedy S., van Putten M. Learn Java with Projects: A concise practical guide to learning everything a Java professional really needs to know. – Packt Publishing, 2023. – 598 p.

41. Knudsen J. Java 2D Graphics: Creating High Quality Graphics & Text. – O'Reilly Media, 1999. – 366 p.

42. Kopec D. Classic Computer Science Problems in Java. – Manning, 2021. – 264 p.

43. Кунгурцев О.Б. Мови об’єктно-орієнтованого програмування / О.Б. Кунгурцев – АО БАХВА, 2006. – 160 с.

44. Learning Java. Niemeyer P., Leuck D. – O'Reilly Media, 2013. – 1007 p.

45. Lowe D. Java All-in-One For Dummies. – For Dummies, 2023. – 912 p.

46. Loy M., Niemeyer P., Leuck D. Learning Java: An Introduction to Real-World Programming with Java. – O'Reilly Media, 2023. – 549 p.

47. McGrath M. Java in easy steps. – In Easy Steps Limited, 2019. – 192 p.

48. Mishra A., Kumar A. Practical Java Programming: Concept and its Application. – Kindle Edition, 2024. – 382 p.

49. Mishra A. Mastering Advanced Java Programming: JDBC, Swing, JSP, Servlets, and EJB. – Kindle Edition, 2024. – 330 p.

50. Naftalin M., Wadler P. Java Generics and Collections: Speed Up the Java Development Process. – O'Reilly Media, 2006. – 284 p.

51. Parlog N. The Java Module System. – Manning Publications, 2019. – 440 p.

52. Payne B. Learn Java the Easy Way: A Hands-On Introduction to Programming. – No Starch Press, 2017. – 312 p.

53. Rao R. N., Kogent Solutions Inc. Core Java: An Integrated Approach. – Dreamtech Press, 2008. – Kindle Edition.

54. Roberts J. Java Jumpstart: A Beginner's Guide To Mastering Java Programming With Code Examples and Exercises. – Kindle Edition, 2024. – 280 p.

55. Семеріков С.О. Фундаменталізація навчання інформатичних дисциплін у вищій школі / С.О. Семеріков – Мінерал, 2009. – 340 с.

56. Salunke M. 1100+ Java Interview Questions and Answers: MCQ Format Questions | Freshers to Experienced | Detailed Explanations. – Kindle Edition, 2023. – 1112 p.

57. Samoylov N. Learn Java 17 Programming: Learn the fundamentals of Java Programming with this updated guide with the latest features. – Packt Publishing, 2022. – 748 p.

58. Savitch W. Java: An Introduction to Problem Solving and Programming. – Pearson, 2014. – 1024 p.

59. Schildt H., Coward D. Java: A Beginner's Guide. – McGraw Hill, 2024. – 768 p.

60. Schildt H., Coward D. Java: The Complete Reference. – McGraw Hill, 2024. – 1280 p.

61. Семеріков С.О. Фундаменталізація навчання інформатичних дисциплін у вищій школі / С.О. Семеріков – Мінерал, 2009. – 340 с.

62. Sierra K., Bates B., Gee T. Head First Java: A Brain-Friendly Guide. – O'Reilly Media, 2022. – 752 p.

63. Tsetsekas H. Object-Oriented Programming Exercises with Java. – Kindle Edition, 2023. – 95 p.

64. Ullenboom C. Java: The Comprehensive Guide to Java Programming for Professionals. – Rheinwerk Computing, 2022. – 1128 p.

65. Urma R.-G., Fusco M., Mycroft A. Modern Java in Action: Lambdas, streams, functional and reactive programming. – Manning, 2018. – 592 p.

66. Васильєв О. Програмування С++ в прикладах і задачах / О. Васильєв. – Ліра-К, 2020. – 382 с.

67. Vickler A. Java: 3 Books in 1: Java Basics for Beginners + Java Front End Programming + Java Back End Programming. – Kindle Edition, 2021. – 623 p.

68. Vibrant Publishers, Bidikar R. Core Java Interview Questions You'll Most Likely Be Asked. – Vibrant Publishers, 2021. – 334 p.

69. Warburton R. Java 8 Lambdas: Functional Programming For The Masses. – O'Reilly Media, 2014. – 180 p.

70. Deshmukh H. OCP Java 21 Programmer Certification Fundamentals: Study Guide For 1Z0-830. – Enthuware, 2025. – Kindle Edition, 1451 p.