БАКАЛАВРСЬКА РОБОТА

БР. ІП – 10.00.00.000 ПЗ

Група ІП-22-4

Пелехович Андрій

2026

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Пелехович Андрій Богданович

(прізвище, ім'я, по-батькові)

УДК 004.942

(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка програмного забезпечення для магазину коврів на Java базованих стеках

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня Пелехович А. Б.

(підпис, ініціали та прізвище здобувача)

Науковий керівник Лютак Ігор Зіновійович, доктор технічних наук, професор

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологі

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

***ЗАТВЕРДЖУЮ*:**
Зав. кафедрою *ІПЗ*
доц. В.В. Бандура
“ ” 2026 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Пелеховичу Андрію Богдановичу

(прізвище, ім’я, по-батькові)

1. ***Тема проекту (роботи)*** ***“Розробка програмного забезпечення для магазину коврів на Java базованих стеках”***

керівник проекту (роботи) Лютак Ігор Зіновійович, доктор технічних наук, професор

Затверджені наказом закладу вищої освіти від “ 21 ” квітня 2026 р. № 179/7

2. ***Строк подання студентом проекту (роботи)*** 10 червня 2026 р

3. ***Вихідні дані до проекту (роботи)*** Офіційна документація Java SE API, документація Swing Framework, документація бібліотеки Gson, специфікації формату JSON (RFC 8259, ECMA-404), технічні вимоги до програмного забезпечення магазину килимів.

4. ***Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)***

1. Огляд існуючих програмних рішень для автоматизації торгівлі та складського обліку. Теоретичні основи проєктування Java-базованих інформаційних систем
2. Аналіз вимог до програмного забезпечення і постановка задачі
3. Проєктування архітектури системи, моделей даних та користувацького інтерфейс
4. Реалізація програмного забезпечення з використанням JSON для зберігання даних
5. Тестування програмного продукту

5. ***Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)***

1. Use-case діаграма програмного забезпечення для магазину килимів (рис. 2.1 ст. 27).
2. Діаграма класів системи обліку товарів і замовлень (рис. 3.2, ст. 34).
3. User-flow створення замовлення в системі (рис. 3.3, ст. 36).
4. Інтерфейс головного вікна програми з таблицями товарів і замовлень (рис. 4.3, ст. 50).
5. Вікно деталей замовлення зі редагуванням та скасуванням замовлення (рис. 4.7, ст. 53).

1. ***Консультанти розділів проекту (роботи)***

Розділ	***Консультант***	***Підпис, дата***	
		Завдання видав	***Завдання прийняв***

2. ***Дата видачі завдання*** _____

Керівник _____ Лютак І.З.
(підпис) (розшифровка підпису)

Завдання прийняв до виконання _____ Пелехович А.Б.
(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	***Назва етапів дипломного проєкту (роботи)***	***Строк виконання етапів проєкту***	***Примітка***
1	Визначення та обґрунтування теми роботи	17.01.2026	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	23.02.2026	виконано
3	Побудова моделі або алгоритму власного рішення	10.03.2026	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	27.04.2026	виконано
5	Оформлення пояснювальної записки бакалаврської роботи	10.06.2026	виконано

Студент – дипломник _____ Пелехович А.Б.
(підпис) (розшифровка підпису)

Керівник роботи _____ Лютак І.З.
(підпис) (розшифровка підпису)

АНОТАЦІЯ

Бакалаврська робота містить 75 сторінки, 20 рисунків, список використаних джерел із 35 найменувань, 2 додатки.

Мета роботи: розробка десктопного програмного забезпечення для автоматизації роботи продавця магазину килимів із використанням Java-базованого стеку.

Об'єкт дослідження: процеси обліку товарів, формування замовлень та контролю складських залишків у магазині килимів.

Предмет дослідження: методи та засоби розробки десктопного Java-додатку для управління товарами, замовленнями, статусами та залишками продукції.

Результати дослідження: розроблено Java-застосунок із графічним інтерфейсом Swing для обліку килимів, створення замовлень, перевірки залишків, розрахунку вартості, роботи зі статусами та збереження даних у JSON-файлах.

У першому розділі розглянуто існуючі рішення та теоретичні основи розробки.

У другому розділі визначено вимоги й основні сценарії використання. **У третьому розділі** спроєктовано архітектуру, модель даних і графічний інтерфейс.

У четвертому розділі описано реалізацію системи.

У п'ятому розділі проведено тестування основних функцій.

Висновок: створено локальний десктопний Java-застосунок, який автоматизує основні процеси роботи продавця магазину килимів і зменшує ризик помилок під час обліку товарів та замовлень.

КЛЮЧОВІ СЛОВА: JAVA, SWING, JSON, GSON, МАГАЗИН КИЛИМІВ, СКЛАДСЬКИЙ ОБЛІК, ЗАМОВЛЕННЯ, АВТОМАТИЗАЦІЯ.

SUMMARY

The bachelor's thesis contains 75 pages, 20 figures, a list of references with 35 titles, and 2 appendices.

Objective: to develop desktop software for automating the work of a carpet store seller using a Java-based technology stack.

Research object: processes of product accounting, order creation, and inventory control in a carpet store.

Subject of research: methods and tools for developing a desktop Java application for managing products, orders, statuses, and inventory in a carpet store.

Research results: a desktop Java application with a Swing interface was developed for product accounting, order management, length validation, cost calculation, status handling, and JSON data storage.

The first section reviews existing trade and inventory automation solutions and the theoretical foundations of Java-based information systems.

The second section defines functional and non-functional requirements, describes the user of the system, and presents the main use cases.

The third section describes the design of the software architecture, data model, order subsystem, service layer, data storage mechanism, and graphical user interface.

The fourth section presents the implementation of the main system modules, including product management, order processing, JSON data storage, and the Swing interface.

The fifth section describes the testing of the main system functions and analyzes possible directions for further development.

Conclusion: a local desktop Java application was created to automate product and order accounting in a carpet store and reduce user errors.

KEYWORDS: JAVA, SWING, JSON, GSON, DESKTOP APPLICATION, CARPET STORE, INVENTORY ACCOUNTING, ORDERS, AUTOMATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ERP — Enterprise Resource Planning, система планування ресурсів підприємства.

JSON — JavaScript Object Notation, текстовий формат збереження та обміну даними.

ООП — об’єктно-орієнтоване програмування.

POS — Point of Sale, система автоматизації продажів у торговельній точці.

SKU — Stock Keeping Unit, одиниця складського обліку товару.

Swing — бібліотека Java для створення графічного інтерфейсу користувача.

UML — Unified Modeling Language, уніфікована мова моделювання.

ВСТУП

У сучасних умовах цифровізація охоплює майже всі сфери діяльності людини, зокрема торгівлю, облік товарів, управління замовленнями та взаємодію з клієнтами. Використання програмного забезпечення у сфері малого та середнього бізнесу дозволяє значно спростити виконання рутинних операцій, зменшити кількість помилок, пришвидшити обробку інформації та підвищити ефективність роботи працівників. Особливо важливим це є для магазинів, де необхідно постійно контролювати наявність товарів, обробляти замовлення, змінювати їх статуси та забезпечувати актуальність складських залишків.

Магазин килимів має певну специфіку порівняно з іншими видами роздрібної торгівлі. Товар може зберігатися як у вигляді рулонів, так і у вигляді окремих шматків, а продаж часто здійснюється не за кількістю одиниць, а за довжиною. Через це продавцю необхідно не лише бачити загальний перелік товарів, а й контролювати ширину, довжину, матеріал, колір, тип килима, ціну за метр та залишок на складі. При ручному веденні такого обліку зростає ризик помилок: можна продати більшу довжину, ніж є в наявності, некоректно розрахувати вартість замовлення або не оновити залишок товару після продажу.

Актуальність теми зумовлена потребою у створенні простого та зручного програмного засобу для автоматизації роботи продавця магазину килимів. Така система повинна забезпечувати швидкий перегляд товарів, додавання та редагування позицій, створення замовлень, автоматичний розрахунок вартості, контроль наявної довжини товару, а також роботу зі статусами замовлень. Особливу увагу потрібно приділити зручності графічного інтерфейсу, оскільки система орієнтована саме на продавця, який має швидко виконувати основні операції без зайвих складнощів.

Метою роботи є розробка десктопного програмного забезпечення для автоматизації роботи продавця магазину килимів із використанням Java-базованого стеку. У межах реалізації передбачається створення графічного інтерфейсу засобами Swing для роботи з товарами, замовленнями, таблицями,

фільтрами та діалоговими вікнами, а також організація збереження даних про товари й замовлення у форматі JSON. Такий підхід дозволяє реалізувати повний цикл роботи з інформацією — від завантаження та перегляду даних до їх редагування, збереження та подальшого використання без залучення складної серверної інфраструктури.

Завданнями дослідження є аналіз предметної області магазину килимів; визначити основні вимоги до програмного забезпечення; спроєктувати структуру системи та основні класи предметної області; реалізувати облік товарів із поділом на рулони та шматки; розробити механізм створення замовлень з кількома позиціями; забезпечити перевірку доступної довжини товару на складі; реалізувати автоматичний розрахунок загальної вартості замовлення; додати зміну статусів замовлень, скасування, відновлення та повне видалення; реалізувати фільтрацію, сортування та візуальне виділення даних у таблицях; забезпечити збереження та завантаження інформації з JSON-файлів.

Об’єктом дослідження є процеси обліку товарів, формування замовлень та контролю складських залишків у магазині килимів.

Предметом дослідження є методи та засоби розробки десктопного Java-додатку для автоматизації роботи продавця магазину килимів, управління товарами, замовленнями, статусами та залишками продукції.

Методи дослідження включають аналіз предметної області, об’єктно-орієнтоване проєктування, розробку програмного забезпечення мовою Java, використання колекцій та потокової обробки даних, побудову графічного інтерфейсу засобами Swing, роботу з JSON-файлами, а також тестування основних сценаріїв використання системи.

Практичне значення роботи полягає у створенні прикладного програмного продукту, який може бути використаний продавцем магазину килимів для спрощення щоденної роботи. Розроблена система дозволяє швидко переглядати каталог товарів, контролювати залишки, створювати замовлення, змінювати їх статуси, виконувати скасування або відновлення замовлень, а також

зменшити ймовірність помилок під час розрахунку вартості та списання товару зі складу.

Розроблене програмне забезпечення може бути розширене у майбутньому шляхом підключення бази даних, додавання ролей користувачів, формування звітів, друку чеків або накладних, а також інтеграції з веб-інтерфейсом чи іншими системами обліку. Таким чином, створений додаток є основою для подальшого розвитку повноцінної інформаційної системи для магазину килимів.

Бакалаврська робота містить 75 сторінки, 20 рисунків, 5 розділів, список використаних джерел із 35 найменувань, 2 додатки

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ ТОРГІВЛІ ТА СКЛАДСЬКОГО ОБЛІКУ. ТЕОРЕТИЧНІ ОСНОВИ ПРОЄКТУВАННЯ JAVA-БАЗОВАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Аналіз сучасного стану автоматизації торгівлі та складського обліку

У сучасних умовах цифровізація бізнес-процесів є важливою складовою розвитку підприємств торгівлі. Робота магазину пов’язана з постійною обробкою інформації про товари, залишки, замовлення та фінансові операції, тому використання програмного забезпечення дозволяє зменшити кількість ручних дій, пришвидшити обслуговування покупців і знизити ризик помилок. Одним із ключових елементів такої діяльності є управління запасами, яке передбачає відстеження товарів від виробника до складу та пунктів продажу з метою забезпечення наявності потрібного товару в потрібний час [3].

Автоматизація торгівлі охоплює ведення каталогу товарів, контроль залишків, створення замовлень, розрахунок вартості покупки та збереження історії операцій. У системах складського обліку важливою перевагою є отримання актуальної інформації про товарні позиції та їх рух [4], що дозволяє уникати оформлення замовлень на відсутній товар.

Ручний облік ускладнює пошук товару й оновлення даних. У матеріалах NetSuite автоматизоване управління запасами визначається як використання технологій для відстеження, контролю та оптимізації запасів без постійного ручного втручання [5]. Для ідентифікації товарів у торгівлі використовується SKU — внутрішній код товарної позиції [6], аналогом якого в розроблюваній системі є унікальний ідентифікатор килима.

Окрім контролю залишків, автоматизація торгівлі дає змогу швидше обробляти інформацію про рух товарів і зменшувати залежність від ручного введення даних. У роздрібній торгівлі важливо не лише знати кількість товару, а

й своєчасно оновлювати інформацію після продажу, повернення або переміщення. У сучасних POS- і складських системах управління запасами розглядається як спосіб уникнення продажу більшої кількості товару, ніж фактично є в наявності [19].

Для магазину килимів така вимога є особливо важливою, оскільки товар може продаватися не лише одиницями, а й частинами рулону. Якщо залишок не оновлюється автоматично, продавець може помилково оформити замовлення на довжину, якої вже немає на складі. Тому автоматизація повинна забезпечувати не тільки збереження інформації про товар, а й контроль його фактичної доступності під час створення замовлення.

Особливої уваги потребують магазини, де товар має додаткові фізичні параметри, зокрема магазини килимів. Килими можуть продаватися не лише як готові вироби, а й як частини рулону певної довжини. Тому продавцю потрібно контролювати назву, матеріал, колір, тип, ціну, розміри, формат зберігання та фактичний залишок товару.

У магазині килимів товар може бути представлений як рулон або шматок. Рулон передбачає продаж частини матеріалу за заданою довжиною, а шматок є готовим залишком фіксованого розміру. Через це система повинна перевіряти доступну довжину, не дозволяти продаж понад залишок, автоматично списувати товар і перераховувати суму замовлення.

Універсальні програмні рішення для торгівлі переважно орієнтовані на продаж товарів поштучно, роботу з касою, клієнтською базою або звітністю. Для невеликого магазину килимів такі системи можуть бути надлишковими або недостатньо пристосованими до продажу товарів за довжиною, тому доцільною є розробка спеціалізованого десктопного застосунку.

Для реалізації такого застосунку може використовуватися Java-базований стек. Бібліотека Swing надає компоненти для створення графічного інтерфейсу, а JTable дозволяє відображати дані у вигляді таблиць [7], [8]. Для збереження інформації про товари та замовлення доцільно застосувати формат JSON і

бібліотеку Gson, яка забезпечує перетворення Java-об’єктів у JSON і зворотне відновлення об’єктів із JSON-рядків [27].

Основні процеси автоматизації наведено на рисунку 1.1.

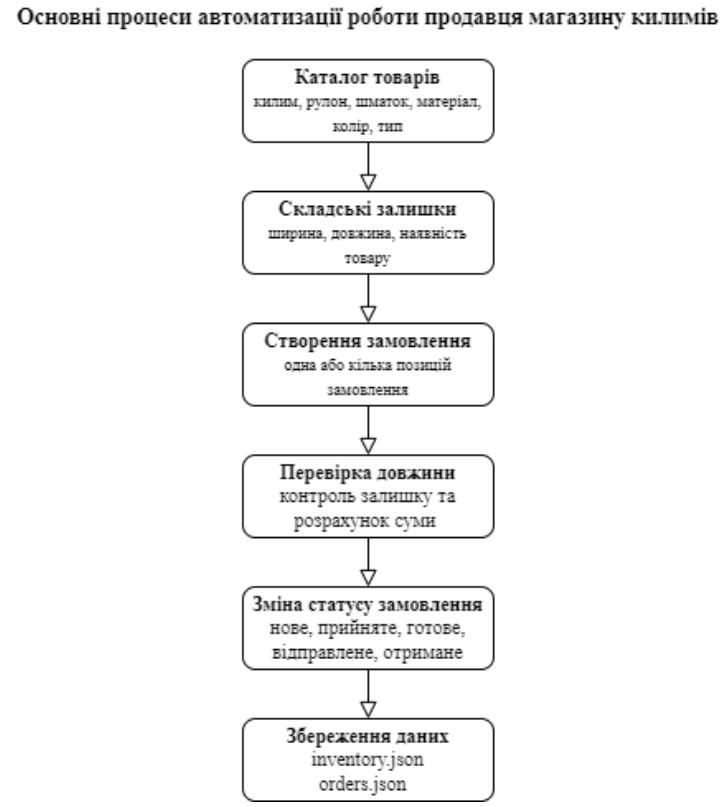


Рисунок 1.1 – Основні процеси автоматизації роботи продавця магазину килимів

Отже, автоматизація торгівлі та складського обліку є актуальною для магазину килимів через необхідність контролю довжини товару, поділу килимів на рулони та шматки, точного розрахунку вартості замовлень і підтримки актуального стану складу. Саме тому розробка десктопного Java-додатку для продавця магазину килимів є доцільним і практично значущим завданням.

1.2 Огляд існуючих програмних рішень для автоматизації магазинів і складського обліку

На ринку програмного забезпечення існує значна кількість рішень для автоматизації торгівлі, складського обліку та управління замовленнями. Такі

системи відрізняються за призначенням, масштабом, способом розгортання та рівнем складності. Умовно їх можна поділити на кілька груп: POS-системи для роздрібної торгівлі, ERP- та складські системи, хмарні сервіси для управління товарами й замовленнями, а також прості табличні рішення на базі Microsoft Excel або Google Sheets.

Одним із поширених класів програмного забезпечення є POS-системи. Вони призначені для організації продажів, контролю залишків, роботи з касою та формування звітності. Наприклад, Square POS надає інструменти для відстеження продажів, контролю запасів, формування звітів про залишки та створення сповіщень про низький рівень товарів . Іншим прикладом є Poster POS, який позиціонується як хмарна система для кафе, ресторанів і магазинів та містить функції складського обліку, зокрема контроль залишків, постачання, інвентаризації, списання і переміщення товарів [10], [11]. Такі рішення зручні для стандартної роздрібної торгівлі, однак не завжди враховують специфіку продажу товарів за довжиною.

Окрему групу становлять ERP- та складські системи, які мають ширші можливості і використовуються для комплексного управління бізнес-процесами. Наприклад, Odoo Inventory дозволяє керувати запасами, постачанням, переміщеннями товарів і поповненням складу [12]. BAS Управління торгівлею також орієнтоване на автоматизацію торговельної діяльності, аналіз поточного стану підприємства та отримання управлінських звітів [13]. Такі системи мають потужний функціонал, проте для невеликого магазину килимів можуть бути надлишковими через складність налаштування та велику кількість можливостей, які не використовуються у щоденній роботі продавця.

Хмарні сервіси, зокрема Zoho Inventory, забезпечують управління товарами, замовленнями, складами, доставкою та іншими процесами через Інтернет [14]. Їхньою перевагою є доступність з різних пристроїв та можливість інтеграції з іншими сервісами. Водночас для локального робочого місця продавця така модель може бути необов’язковою, особливо якщо система використовується в межах одного магазину. Найпростішим варіантом обліку залишаються

електронні таблиці. Microsoft пропонує готові шаблони інвентаризації, які дозволяють вести облік товарів і кількості [15]. Проте таблиці не забезпечують повноцінної перевірки бізнес-правил, автоматичного контролю залишків і статусів замовлень.

Окремо можна виділити системи електронної комерції та POS-рішення, які поєднують облік товарів, продажі й контроль залишків. Наприклад, Shopify POS дозволяє створювати товари, відстежувати й коригувати залишки через адміністративну панель або POS-інтерфейс [19]. Такі рішення зручні для магазинів, які працюють одночасно онлайн і офлайн, однак вони орієнтовані переважно на стандартні товарні позиції.

Порівняння існуючих рішень показує, що більшість із них має широкий набір функцій, але не завжди підтримує специфічну логіку продажу товарів за довжиною. Для магазину килимів важливо, щоб система враховувала залишок рулону, забороняла продаж понад доступну довжину та автоматично перераховувала суму замовлення. Саме тому власний десктопний застосунок може бути більш зручним для невеликого магазину, ніж універсальна POS- або ERP-система.

Результати порівняння наведено в таблиці 1.1.

Таблиця 1.1

Порівняльна характеристика програмних рішень для автоматизації торгівлі та складського обліку

Тип рішення	**Приклади**	**Основні можливості**	**Переваги**	**Обмеження**
POS-системи	Square POS, Poster POS	Продажі, залишки, каса, звітність	Швидке оформлення продажів	Слабка підтримка продажу за довжиною
ERP / складські системи	Odoo Inventory, BAS	Запаси, постачання, переміщення	Комплексна автоматизація	Надлишковість для малого магазину
Хмарні сервіси	Zoho Inventory	Товари, замовлення, склади	Доступ через Інтернет	Залежність від мережі
Електронні таблиці	Microsoft Excel, Google Sheets	Ручний облік, прості розрахунки	Доступність, простота використання	Високий ризик помилок
Власна Java-система	Десктопний застосунок	Каталог, замовлення, залишки, JSON	Урахування специфіки килимів	Потребує розробки

Для магазину килимів особливо важливо, щоб система враховувала специфіку товару. На відміну від стандартних товарів, які продаються поштучно, килими можуть зберігатися у вигляді рулонів або шматків, а продаж рулону часто здійснюється за заданою довжиною. Тому програмне забезпечення має не лише зберігати перелік товарів, а й перевіряти доступну довжину, автоматично списувати продану частину, перераховувати загальну вартість замовлення та не дозволяти створити замовлення на більшу довжину, ніж є на складі.

Отже, аналіз існуючих програмних рішень показує, що універсальні POS-, ERP-, хмарні та табличні системи мають корисні можливості для торгівлі й складського обліку, однак не завжди зручно адаптуються до потреб магазину килимів. Для такої предметної області доцільно створити спеціалізований десктопний застосунок, який враховує облік товарів за довжиною, поділ на рулони та шматки, перевірку залишків і створення замовлень із кількома позиціями.

1.3 Основні поняття та особливості предметної області магазину килимів

Предметна область магазину килимів поєднує роздрібну торгівлю, складський облік і управління замовленнями. На відміну від стандартних торговельних систем, де товар часто продається поштучно, у магазині килимів важливими є фізичні характеристики товару: ширина, довжина, матеріал, колір, тип і формат зберігання.

Базовою сутністю системи є килим як товарна позиція. Для його ідентифікації використовується унікальний номер, що виконує роль внутрішнього коду товару. Подібний підхід застосовується у роздрібній торгівлі через поняття SKU — унікальної одиниці складського обліку, яка допомагає відстежувати товари, керувати запасами й аналізувати продажі [6]. У розроблюваній системі ідентифікатор килима використовується для пошуку товару, редагування його характеристик і додавання до замовлення.

Використання унікального ідентифікатора товару є важливим для коректної роботи каталогу. Він дозволяє однозначно визначити потрібну позицію навіть тоді, коли кілька килимів мають подібну назву, однаковий колір або матеріал. У практиці роздрібної торгівлі SKU використовується для внутрішнього відстеження товарів, контролю залишків і аналізу продажів [6]. У розроблюваній системі аналогічну роль виконує ідентифікатор килима.

Також важливо враховувати, що товарна позиція в магазині килимів має більше характеристик, ніж звичайний поштучний товар. Наприклад, два килими можуть мати однакову назву, але відрізнятися шириною, довжиною, матеріалом або форматом зберігання. Через це система повинна зберігати не лише назву та ціну, а й повний набір параметрів, необхідних для пошуку, фільтрації та правильного оформлення замовлення.

Килим описується назвою, шириною, довжиною, матеріалом, кольором, типом, форматом і ціною за метр. Назва спрощує пошук товару в каталозі, розміри визначають його фізичні параметри, а матеріал, колір і тип використовуються для фільтрації. Ціна за метр є основою для автоматичного розрахунку вартості позиції замовлення.

Каталог товарів у такій системі виконує роль основного джерела інформації для продавця. Через нього користувач отримує дані про наявні килими, їх характеристики, ціну та залишок. Тому каталог повинен бути зручним для перегляду, пошуку й редагування. Наявність фільтрації за матеріалом, кольором, типом або форматом товару дозволяє швидше знаходити потрібну позицію серед великої кількості записів.

Важливою особливістю є поділ килимів на рулони та шматки. Рулон передбачає можливість продажу частини товару за заданою довжиною, тому система повинна перевіряти доступний залишок і після продажу зменшувати довжину рулону. Шматок є готовим залишком фіксованого розміру, тому його довжина під час продажу не повинна довільно змінюватися.

Для уникнення неточностей довжина товару в системі зберігається в

сантиметрах, а в інтерфейсі може відображатися в метрах. Наприклад, 500 см відповідає 5 м. Такий підхід спрощує перевірку залишків, розрахунок вартості та оновлення складських даних. Якщо введена довжина перевищує фактичний залишок, створення позиції має бути заборонене, що відповідає загальній ідеї управління запасами [3].

Окремою сутністю є замовлення, яке містить одну або кілька позицій, загальну суму та поточний статус. Позиція замовлення пов’язує конкретний килим із довжиною, що продається, і використовується для розрахунку вартості. Якщо замовлення містить кілька позицій, система повинна окремо розрахувати суму кожної з них і сформувати загальну вартість покупки.

Важливим процесом є оновлення складських залишків після створення, скасування або відновлення замовлення. Після продажу відповідна довжина рулону повинна списуватися зі складу. У разі скасування замовлення товар має повертатися назад до залишку. Це забезпечує актуальність даних і зменшує ризик помилок під час подальшого продажу.

Статуси замовлення відображають етап його обробки: нове, прийняте в роботу, готове до відправки, відправлене, отримане або повернуте/скасоване. Такий підхід відповідає ідеї workflow, коли об’єкт проходить через послідовність визначених станів [16]. Завдяки статусам продавець може швидко визначити поточний стан кожного замовлення.

З точки зору програмної реалізації ці поняття доцільно подати у вигляді окремих класів і перелічуваних типів. Килим може бути представлений класом Carpet, його розміри — CarpetDetails, атрибути — CarpetAttributes, а матеріал, колір і тип — enum-класами. Замовлення реалізується класом Order, позиція замовлення — OrderItem, а статус — OrderStatus.

Основні сутності предметної області магазину килимів наведено на рисунку 1.2.

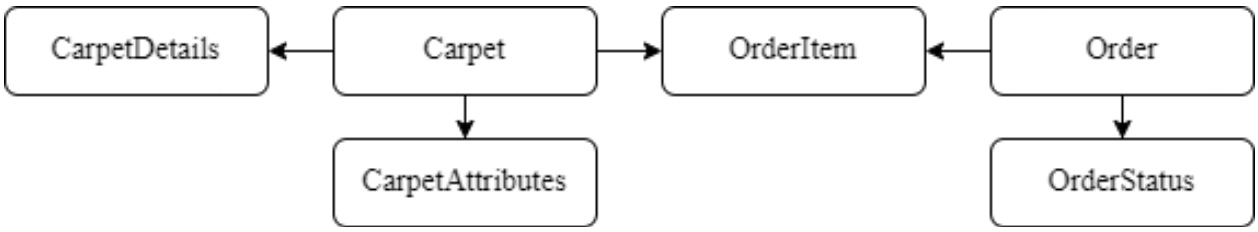


Рисунок 1.2 – Основні сутності предметної області магазину килимів

Отже, предметна область магазину килимів вимагає врахування продажу товарів за довжиною, поділу на рулони та шматки, контролю залишків, розрахунку вартості та підтримки статусів замовлення. Ці поняття формують основу для подальшого проєктування структури програмної системи.

1.4 Теоретичні основи проєктування Java-базованих десктопних інформаційних систем

Проєктування десктопної інформаційної системи передбачає вибір технологій, структури програмного коду та способу організації взаємодії між частинами програми. Для розробки програмного забезпечення магазину килимів доцільно використати Java-базований стек, оскільки він дозволяє реалізувати об’єктну модель предметної області, графічний інтерфейс користувача, роботу з колекціями та збереження даних у файлах.

Java підтримує об’єктно-орієнтоване програмування, основними поняттями якого є класи, об’єкти, інтерфейси, наслідування та пакети [17]. Для розроблюваної системи це є важливим, оскільки основні сутності магазину килимів можуть бути представлені окремими класами. Наприклад, килим, його розміри, атрибути, замовлення, позиція замовлення та статус замовлення можуть мати власну структуру й відповідальність.

Важливим елементом Java-застосунків є колекції. Інтерфейс Collection у Java представляє групу об’єктів, які називаються елементами [18]. У межах системи колекції можуть використовуватися для зберігання списку килимів,

списку замовлень і позицій у межах одного замовлення. Наприклад, каталог товарів може бути поданий як список об’єктів Carpet, а замовлення — як список об’єктів OrderItem.

Для зручної організації коду доцільно розділити систему на кілька логічних частин: model, service, storage та ui. Пакет model містить класи предметної області, service відповідає за бізнес-логіку, storage забезпечує збереження і завантаження даних, а ui містить графічний інтерфейс користувача. Такий поділ спрощує підтримку програми, оскільки зміни в одному шарі менше впливають на інші частини системи.

Для реалізації такої структури важливим є принцип розділення відповідальності між частинами програми. Класи моделі повинні описувати дані предметної області, сервісний шар — виконувати перевірки та основні операції, сховище — відповідати за збереження інформації, а інтерфейс — забезпечувати взаємодію з користувачем. Такий підхід спрощує супровід програми та дозволяє змінювати окремі частини системи без повної переробки застосунку [20].

Для створення графічного інтерфейсу в роботі використовується Swing. Згідно з документацією Oracle, пакет javax.swing містить набір легких компонентів для побудови графічного інтерфейсу користувача [7]. У межах системи Swing дозволяє створити головне вікно, кнопки, діалогові вікна, поля введення, таблиці товарів і замовлень.

Під час розробки графічного інтерфейсу важливе значення мають стандартні компоненти Swing. Наприклад, JFrame використовується як основне вікно застосунку, JTable — для відображення табличних даних, а JOptionPane — для показу повідомлень і діалогів підтвердження [21], [22], [23]. Це відповідає потребам системи магазину килимів, де основна інформація подається у вигляді таблиць товарів і замовлень.

Одним із ключових компонентів інтерфейсу є JTable, який використовується для відображення двовимірних таблиць даних [8]. У розроблюваному застосунку таблиці застосовуються для показу каталогу килимів і списку замовлень, що дозволяє продавцю швидко переглядати основну

інформацію про товари, їх розміри, ціну, формат, а також суму й статус замовлень. Практичні рекомендації щодо створення та налаштування таблиць у Swing наведені в офіційних навчальних матеріалах Oracle [24].

Для збереження даних використовується формат JSON. Він є зручним для локального десктопного застосунку, оскільки дозволяє зберігати інформацію у зрозумілому текстовому форматі без використання бази даних. У Java для роботи з JSON застосовується бібліотека Gson, яка забезпечує перетворення Java-об’єктів у JSON і зворотне відновлення даних [27]. Завдяки цьому товари зберігаються у файлі inventory.json, а замовлення — у файлі orders.json.

Отже, Java-базований стек є доцільним для реалізації десктопної системи магазину килимів. Java забезпечує об’єктно-орієнтовану основу, Swing і JTable формують графічний інтерфейс, а JSON у поєднанні з Gson забезпечує просте локальне збереження даних.

1.5 Висновки по розділу

У першому розділі було розглянуто сучасний стан автоматизації торгівлі та складського обліку, а також проаналізовано існуючі програмні рішення для магазинів. Встановлено, що універсальні POS-, ERP-, хмарні та табличні системи мають широкі можливості, однак не завжди враховують специфіку магазину килимів, зокрема продаж товарів за довжиною, поділ на рулони й шматки та контроль залишків.

Також було визначено основні поняття предметної області: товарна позиція, каталог, складський залишок, замовлення, позиція замовлення та статус. Обґрунтовано доцільність використання Java-базованого стеку, Swing, JTable, JSON і Gson для розробки локального десктопного застосунку. Отримані результати створюють основу для подальшого аналізу вимог, постановки задачі та проєктування системи.

РОЗДІЛ 2. АНАЛІЗ ВИМОГ І ПОСТАНОВКА ЗАДАЧІ

2.1 Аналіз предметної області та користувача системи

Предметною областю розроблюваного програмного забезпечення є діяльність магазину килимів, пов’язана з обліком товарів, контролем складських залишків і формуванням замовлень. Основним користувачем системи є продавець, який щоденно працює з каталогом товарів, перевіряє їх наявність, створює замовлення для покупців і контролює їхній поточний стан.

Особливістю магазину килимів є те, що товари можуть зберігатися у вигляді рулонів або окремих шматків. Рулони можуть продаватися частинами залежно від потрібної покупцеві довжини, тому для них необхідно контролювати доступний залишок. Шматки мають фіксований розмір і продаються як готові позиції. Через це система повинна враховувати формат товару, його ширину, довжину, матеріал, колір, тип і ціну за метр.

Основні процеси роботи продавця включають перегляд каталогу, додавання й редагування товарів, створення замовлень, автоматичний розрахунок вартості, зміну статусів, скасування та відновлення замовлень. Під час створення замовлення система повинна перевіряти доступну довжину товару, забороняти продаж понад залишок, списувати товар після продажу та повертати його на склад у разі скасування.

Важливою частиною роботи магазину є правильний контроль залишків. Якщо облік ведеться вручну, продавець може помилково вказати більшу довжину товару, ніж фактично є на складі, або не врахувати повернення після скасування замовлення. Автоматизація цих дій дозволяє зменшити кількість помилок і забезпечити актуальність даних у каталозі.

Оскільки система орієнтована на продавця, її інтерфейс має бути простим і зрозумілим. Основна інформація повинна відображатися у вигляді таблиць: окремо список товарів і список замовлень. Для зручності роботи доцільно

передбачити фільтрацію, сортування та кольорове виділення важливих станів, наприклад нових або скасованих замовлень і товарів із малим залишком.

Крім обліку товарів, важливим процесом є формування замовлення. Покупець може обрати один або кілька килимів, тому система повинна підтримувати додавання кількох позицій до одного замовлення. Для кожної позиції необхідно враховувати вибраний товар, довжину, ціну за метр і підсумкову вартість.

Під час роботи із замовленнями важливо забезпечити правильний зв’язок між продажем і складськими залишками. Якщо товар продається з рулону, його доступна довжина повинна автоматично зменшуватися. Якщо замовлення скасовується або відновлюється, система має відповідно повертати товар на склад або повторно перевіряти його наявність.

Також система повинна допомагати продавцю швидко орієнтуватися у великій кількості записів. Для цього доцільно використовувати фільтри, сортування та статуси замовлень. Такі можливості дають змогу швидше знаходити потрібні товари, контролювати виконання замовлень і зменшувати кількість помилок під час щоденної роботи.

Оскільки основним користувачем системи є продавець, програмне забезпечення повинно враховувати його повсякденні робочі дії. Продавець має швидко переглядати інформацію про товар, перевіряти його наявність, оформлювати замовлення та бачити результат виконаної операції без складних налаштувань. Тому система повинна бути орієнтована на простоту використання й мінімальну кількість зайвих дій.

Також важливо врахувати, що розроблюване програмне забезпечення призначене для локального використання в межах одного робочого місця. Це означає, що система повинна працювати без підключення до серверної частини, зберігати дані між запусками програми та забезпечувати доступ до основної інформації без потреби в складній інфраструктурі.

Ще одним важливим аспектом предметної області є необхідність швидкого отримання актуальної інформації про товар. Продавець повинен бачити не лише

назву килима, а й його основні характеристики: формат, розміри, матеріал, колір, тип і ціну. Це дозволяє швидше консультувати покупця, підбирати потрібний товар і перевіряти можливість його продажу без додаткового пошуку в окремих записах або документах.

Також слід врахувати, що в процесі роботи продавець виконує не одну окрему дію, а послідовність пов’язаних операцій. Наприклад, під час оформлення замовлення він спочатку знаходить потрібний товар, перевіряє його залишок, додає позицію, підтверджує замовлення та після цього контролює його статус. Тому система повинна підтримувати логічний і послідовний процес роботи, у якому кожна дія пов’язана з попередньою та наступною.

Отже, розроблюване програмне забезпечення має бути спрямоване на автоматизацію повсякденних дій продавця магазину килимів. Воно повинно поєднувати облік товарів, контроль залишків, створення замовлень, управління статусами та локальне збереження даних у зручній для використання формі.

2.2 Визначення вимог і сценаріїв використання системи

Вимоги до програмного забезпечення визначають основні можливості системи та загальні характеристики її роботи. Оскільки застосунок призначений для продавця магазину килимів, головна увага приділяється роботі з товарами, замовленнями, складськими залишками та статусами.

До основних функціональних вимог системи належать: перегляд каталогу товарів, додавання й редагування килимів, поділ товарів на рулони та шматки, створення замовлень, додавання кількох позицій до замовлення, перевірка доступної довжини товару, автоматичний розрахунок вартості, оновлення складських залишків, перегляд деталей замовлення, зміна статусу, скасування та відновлення замовлення, фільтрація, сортування, підтвердження дій користувача, збереження та завантаження даних.

Окрему увагу потрібно приділити роботі із замовленнями. Замовлення може містити одну або кілька позицій, тому система повинна дозволяти продавцю

додавати різні товари та вказувати потрібну довжину для рулонних килимів. Після додавання позицій програма має автоматично розраховувати вартість кожної позиції та загальну суму замовлення. Це зменшує кількість ручних розрахунків і підвищує точність оформлення продажу.

Нефункціональні вимоги визначають якість роботи системи. Інтерфейс програми має бути простим, наочним і зручним для щоденного використання. Дані про товари й замовлення повинні відображатися у вигляді таблиць, а важливі стани можуть виділятися кольором. Система має перевіряти введені дані, запобігати некоректним діям, забезпечувати цілісність складських залишків і виконувати критичні операції лише після підтвердження користувача.

Також важливими вимогами є швидкодія, підтримуваність і розширюваність системи. Операції перегляду, фільтрації, сортування та створення замовлень мають виконуватися без помітних затримок. Програмний код повинен бути поділений на логічні частини: модель даних, сервісний шар, роботу зі сховищем і графічний інтерфейс. Такий підхід спрощує внесення змін і дозволяє надалі розширювати функціональність системи.

Основним користувачем системи є продавець. Він взаємодіє з графічним інтерфейсом програми та виконує основні сценарії: перегляд каталогу товарів, додавання нового товару, редагування товару, створення замовлення, перегляд деталей замовлення, зміна статусу, скасування або відновлення замовлення, фільтрація та сортування даних.

Під час визначення вимог важливо враховувати не лише основні функції, а й обмеження, які повинні запобігати некоректній роботі системи. Зокрема, програма не повинна дозволяти створювати порожні замовлення, додавати товар без потрібних характеристик, вводити від’ємну або нульову довжину, а також продавати рулонний товар у кількості, що перевищує фактичний залишок. Такі обмеження забезпечують правильність даних і зменшують ризик помилок користувача.

Окремим сценарієм використання є робота з уже створеним замовленням. Продавець повинен мати змогу переглянути його склад, перевірити суму, змінити

статус, за потреби скасувати або відновити замовлення. При цьому система має контролювати, щоб усі зміни впливали на складські залишки коректно, а критичні дії виконувалися лише після підтвердження користувача.

Для наочного подання основних сценаріїв використання створено use-case діаграму, на якій користувачем виступає продавець, а основними варіантами використання є робота з товарами та замовленнями.

Use-case діаграму основних сценаріїв використання програмного забезпечення наведено на рисунку 2.1.

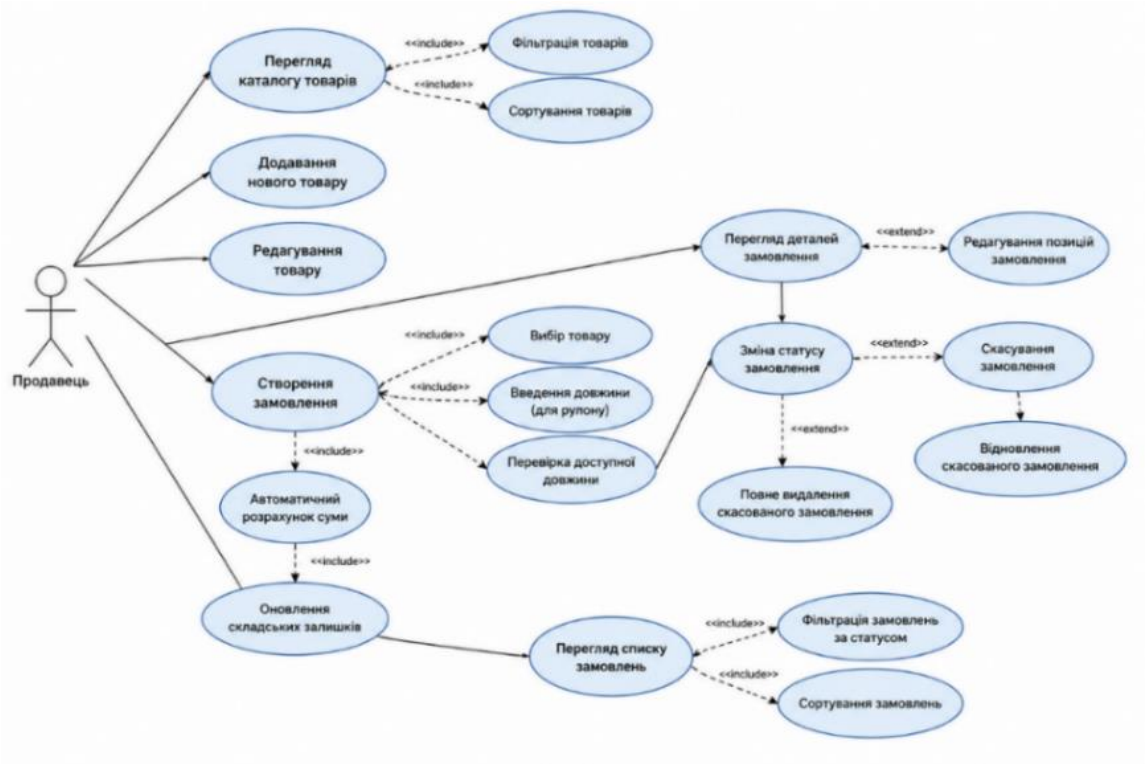


Рисунок 2.1 – Use-case діаграма програмного забезпечення для магазину килимів

На діаграмі показано основні дії продавця в системі: перегляд каталогу товарів, додавання та редагування килимів, створення замовлення, перегляд його деталей, зміну статусу, скасування або відновлення замовлення. Також передбачено фільтрацію й сортування даних, які допомагають швидше знаходити потрібну інформацію в таблицях.

Отже, визначення вимог і моделювання сценаріїв використання дозволяють встановити основні дії користувача та створюють основу для

подальшого проєктування структури програмного забезпечення й графічного інтерфейсу.

2.3 Постановка задачі на розробку програмного забезпечення

На основі аналізу предметної області та визначених вимог необхідно розробити десктопне програмне забезпечення для автоматизації роботи продавця магазину килимів. Система має забезпечувати облік товарів, контроль складських залишків, створення замовлень, розрахунок їх вартості та збереження даних між запусками програми.

Розроблюване програмне забезпечення повинно бути реалізоване мовою Java з використанням графічного інтерфейсу Swing. Дані про товари та замовлення зберігатимуться у JSON-файлах, що дозволить використовувати систему локально без серверної частини або бази даних.

У межах розробки потрібно створити модель даних для опису товарів, їх характеристик, замовлень, позицій замовлення та статусів. Модель повинна враховувати поділ килимів на рулони та шматки, а також забезпечувати зберігання даних про розміри, ціну, матеріал, колір і тип товару.

Також необхідно реалізувати функціонал роботи з товарами та замовленнями. Система повинна підтримувати додавання й редагування товарів, створення замовлень із кількома позиціями, перевірку доступної довжини, автоматичний розрахунок вартості, зміну статусів, скасування та відновлення замовлень. Особливу увагу потрібно приділити правильному оновленню залишків після створення або скасування замовлення.

Окремою задачею є проєктування графічного інтерфейсу, який має забезпечувати зручну взаємодію продавця із системою. У головному вікні потрібно передбачити відображення каталогу товарів і списку замовлень у вигляді таблиць, а для додавання товарів, створення замовлень і перегляду деталей використовувати окремі діалогові вікна.

Також необхідно передбачити механізми перевірки введених даних і

підтвердження критичних дій. Система повинна запобігати створенню порожніх замовлень, продажу товару понад доступний залишок, введенню некоректної довжини або ціни.

Для забезпечення зручності супроводу програмного забезпечення проєкт доцільно поділити на окремі логічні частини: модель даних, сервісний шар, механізм збереження та графічний інтерфейс. Це спростить подальше внесення змін і розширення функціональності системи.

Також важливо забезпечити коректну роботу програми після повторного запуску. Дані про товари, їх залишки та замовлення не повинні втрачатися після закриття застосунку. Тому система має завантажувати інформацію з файлів під час запуску та зберігати оновлений стан після додавання товарів, створення замовлень, редагування даних або зміни статусів.

У межах розробки необхідно виконати такі основні задачі:

- розробити структуру Java-проєкту з поділом на окремі пакети для моделі даних, сервісного шару, збереження інформації та графічного інтерфейсу;
- створити модель даних системи для опису килимів, їх розмірів, атрибутів, формату зберігання, замовлень, позицій замовлення та статусів;
- реалізувати облік товарів із можливістю додавання, редагування, перегляду й пошуку килимів у каталозі;
- забезпечити поділ товарів на рулони та шматки з урахуванням різної логіки їх продажу;
- реалізувати створення замовлень із кількома позиціями та автоматичним розрахунком вартості кожної позиції й загальної суми;
- забезпечити перевірку доступної довжини товару під час додавання рулонного килима до замовлення;
- реалізувати автоматичне оновлення складських залишків після створення, скасування або відновлення замовлення;
- передбачити зміну статусів замовлення відповідно до етапів його обробки;

- реалізувати механізми перевірки введених даних і підтвердження критичних дій користувача;
- створити графічний інтерфейс користувача засобами Swing із таблицями товарів і замовлень, кнопками керування та діалоговими вікнами;
- забезпечити фільтрацію, сортування та зручне відображення даних для швидкої роботи продавця;
- реалізувати локальне збереження та завантаження даних про товари й замовлення у JSON-файлах.

Результатом виконання поставленої задачі має стати локальний десктопний Java-застосунок, який дозволяє продавцю магазину килимів швидко працювати з товарами та замовленнями, контролювати складські залишки, розраховувати вартість покупки та зменшувати кількість ручних операцій під час обліку.

2.4 Висновки по розділу

У другому розділі було проаналізовано предметну область магазину килимів і визначено основного користувача системи — продавця. Встановлено, що програмне забезпечення має враховувати специфіку товарів, зокрема поділ килимів на рулони та шматки, контроль доступної довжини й автоматичне оновлення складських залишків.

Було визначено функціональні та нефункціональні вимоги до системи, а також змодельовано основні сценарії використання. Це дозволило встановити, які дії повинен виконувати продавець у програмі та які можливості має забезпечувати графічний інтерфейс.

На основі проведеного аналізу сформульовано постановку задачі на розробку десктопного Java-застосунку із використанням Swing та JSON-файлів для локального збереження даних.

РОЗДІЛ 3. ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Загальна архітектура програмного забезпечення

Проєктування програмного забезпечення для магазину килимів передбачає визначення загальної структури системи, розподілу відповідальності між її частинами та способу взаємодії між основними модулями. Оскільки розроблюваний застосунок є десктопною Java-системою з графічним інтерфейсом Swing і збереженням даних у JSON-файлах, доцільно використати багаторівневу архітектуру з поділом на окремі логічні пакети.

У межах проєкту система поділяється на такі основні частини:

- **model** – пакет, що містить класи предметної області;
- **service** – пакет, що відповідає за бізнес-логіку;
- **storage** – пакет для збереження та завантаження даних;
- **ui** – пакет графічного інтерфейсу користувача;
- **data** – каталог із JSON-файлами, у яких зберігаються товари та замовлення.

Такий поділ дозволяє відокремити дані, логіку їх обробки, механізм збереження та інтерфейс користувача. Це спрощує підтримку програми, оскільки зміни в одному модулі не потребують повного переписування всієї системи. Наприклад, якщо у майбутньому JSON-файли будуть замінені на базу даних, основні зміни стосуватимуться пакета storage, тоді як класи моделі та інтерфейс можуть залишитися майже без змін.

Пакет model містить основні сутності предметної області. До нього належать класи для опису килимів, їх характеристик, атрибутів, списку товарів, замовлень, позицій замовлення та статусів. Саме цей пакет формує об’єктну модель системи, яка використовується іншими частинами програми.

Пакет service відповідає за виконання основної бізнес-логіки. У ньому реалізуються операції додавання та редагування товарів, створення замовлень, перевірка доступної довжини товару, розрахунок вартості, зміна статусів,

скасування та відновлення замовлень. Завдяки винесенню цієї логіки в окремий шар графічний інтерфейс не перевантажується обробкою даних.

Пакет storage забезпечує роботу з файлами. Він відповідає за завантаження інформації з JSON-файлів під час запуску програми та збереження оновлених даних після змін. У межах системи використовуються файли inventory.json для зберігання товарів і orders.json для зберігання замовлень.

Пакет ui містить графічний інтерфейс користувача. Саме в ньому реалізовано головне вікно програми, таблиці товарів і замовлень, форми додавання та редагування килимів, вікно створення замовлення, вікно деталей замовлення, фільтрацію, сортування та підтвердження дій користувача.

Загальна взаємодія між частинами системи відбувається таким чином: користувач виконує дію в інтерфейсі, після чого ui звертається до відповідного сервісу. Сервіс обробляє запит, працює з об’єктами моделі та за потреби звертається до шару збереження. Після виконання операції дані оновлюються в інтерфейсі та зберігаються у JSON-файлах.

Схематично архітектуру системи наведено на рисунку 3.1.

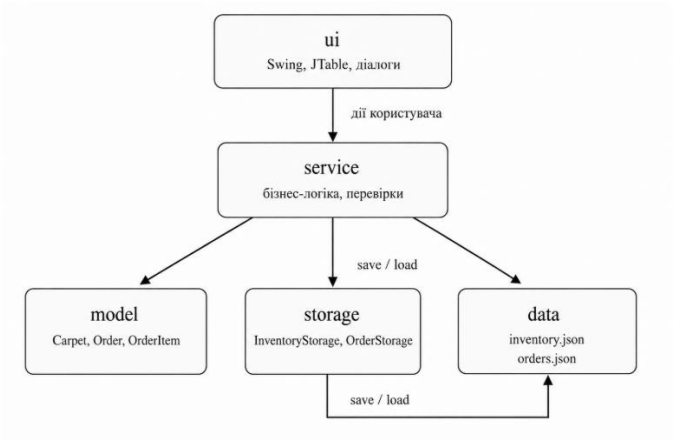


Рисунок 3.1 – Структура Java-проєкту та взаємодія основних модулів системи

Отже, запропонована архітектура дозволяє розділити систему на логічні частини, кожна з яких має власну відповідальність. Такий підхід робить програмне забезпечення зрозумілішим, зручнішим для підтримки та придатним до подальшого розширення.

3.2 Проєктування моделі даних і підсистеми замовлень

Проєктування моделі даних є важливим етапом розробки програмного забезпечення, оскільки саме на цьому рівні визначаються основні сутності системи, їхні властивості та зв’язки між ними. Для магазину килимів модель даних повинна відображати інформацію про товари, їх характеристики, складські залишки, замовлення та позиції замовлень.

Основною сутністю системи є килим, який у програмі представлений класом Carpet. Цей клас містить ідентифікатор товару, назву, розмірні характеристики, атрибути та ціну за метр. Ідентифікатор використовується для пошуку товару в системі, назва — для відображення у каталозі, а ціна — для розрахунку вартості позиції замовлення.

Для кращої організації даних характеристики килима поділено на окремі логічні частини. Розміри та формат товару винесено в клас CarpetDetails, який містить ширину, довжину та ознаку того, чи є товар рулоном. Такий підхід дозволяє окремо працювати з фізичними параметрами килима та спрощує перевірку залишків під час створення замовлення.

Атрибути килима винесено в клас CarpetAttributes. Він містить матеріал, колір і тип килима. Для цих характеристик доцільно використовувати перелічувані типи Material, CarpetColor і CarpetType. Це зменшує ймовірність помилок під час введення даних, оскільки користувач обирає значення з обмеженого набору, а не вводить їх вручну.

Для зберігання списку товарів використовується клас Inventory, який містить колекцію об’єктів Carpet. Він є логічним представленням складу магазину та забезпечує можливість працювати з набором товарів як з єдиною структурою. У подальшій реалізації цей список використовується сервісним шаром для додавання, редагування, пошуку та відображення товарів.

Окрему частину моделі даних становить підсистема замовлень. Основною сутністю тут є клас Order, який містить ідентифікатор замовлення, список позицій,

загальну суму та поточний статус. Оскільки одне замовлення може містити кілька різних килимів, для опису окремої позиції використовується клас OrderItem.

Клас OrderItem пов’язує конкретний килим із довжиною, яка додається до замовлення. На основі довжини та ціни за метр розраховується вартість позиції за формулою:

вартість позиції = довжина в метрах × ціна за метр.

Наприклад, якщо покупець замовляє 5 метрів килима за ціною 500 грн за метр, то вартість позиції становитиме 2500 грн. Загальна сума замовлення формується як сума вартостей усіх його позицій.

Для відображення стану замовлення використовується перелічуваний тип OrderStatus. Він містить основні етапи обробки замовлення: нове, прийняте в роботу, готове до відправки, відправлене, отримане та повернуте/скасоване. Завдяки цьому система може відображати поточний стан кожного замовлення та дозволяти продавцю змінювати його відповідно до етапу виконання.

Зв’язки між основними класами моделі даних можна описати таким чином: Carpet містить об’єкти CarpetDetails і CarpetAttributes; Inventory зберігає список об’єктів Carpet; Order містить список об’єктів OrderItem; кожен OrderItem пов’язаний з конкретним об’єктом Carpet; статус замовлення визначається через OrderStatus.

Діаграму класів системи наведено на рисунку 3.2.

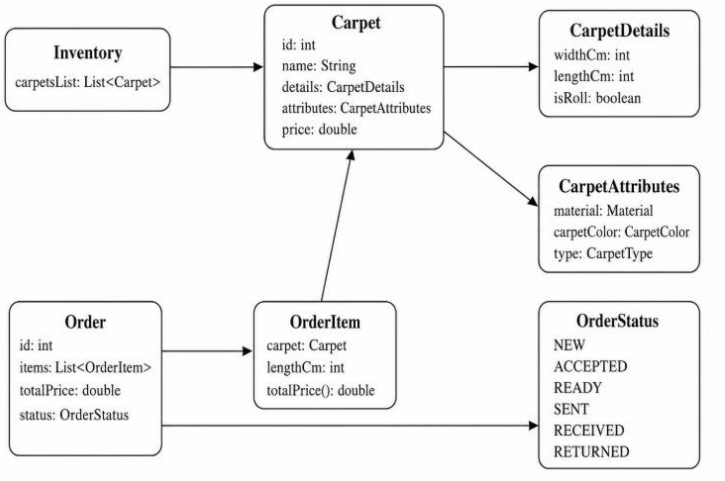


Рисунок 3.2 – Діаграма класів системи обліку товарів і замовлень

Підсистема замовлень є однією з основних частин програмного забезпечення, оскільки саме вона забезпечує оформлення покупки, розрахунок вартості та зміну складських залишків. Для магазину килимів ця підсистема має враховувати специфіку продажу товарів за довжиною, можливість додавання кількох позицій до одного замовлення та контроль доступної кількості товару на складі.

Під час створення замовлення система повинна виконувати перевірку доступної довжини товару. Якщо товар є рулоном, продавець може ввести потрібну довжину, але вона не повинна перевищувати залишок на складі. Якщо товар є шматком, його довжина є фіксованою, тому система повинна використовувати повну довжину такого товару без можливості довільної зміни.

Після успішного додавання позиції до замовлення залишок товару має бути оновлений. Для рулону зменшується доступна довжина, а для шматка товар фактично вважається проданим. Такий підхід дозволяє підтримувати актуальний стан складу та уникати ситуацій, коли один і той самий товар продається повторно.

Важливою частиною підсистеми є скасування та відновлення замовлення. У разі скасування статус змінюється на RETURNED, а товари повертаються на склад. Відновлення можливе лише після повторної перевірки наявності потрібної довжини товару. Якщо залишку недостатньо, система не повинна дозволяти відновлення такого замовлення.

Також передбачено редагування деталей замовлення. Продавець може переглянути позиції, змінити довжину рулонного товару або видалити окрему позицію. Якщо після видалення замовлення стає порожнім, воно не повинно залишатися у списку активних замовлень.

Процес створення замовлення включає вибір товару, введення довжини для рулону, перевірку залишку, додавання позиції, розрахунок суми та підтвердження замовлення. Цей процес доцільно подати у вигляді user-flow діаграми.

Схематично процес створення замовлення в системі наведено на рисунку 3.3.

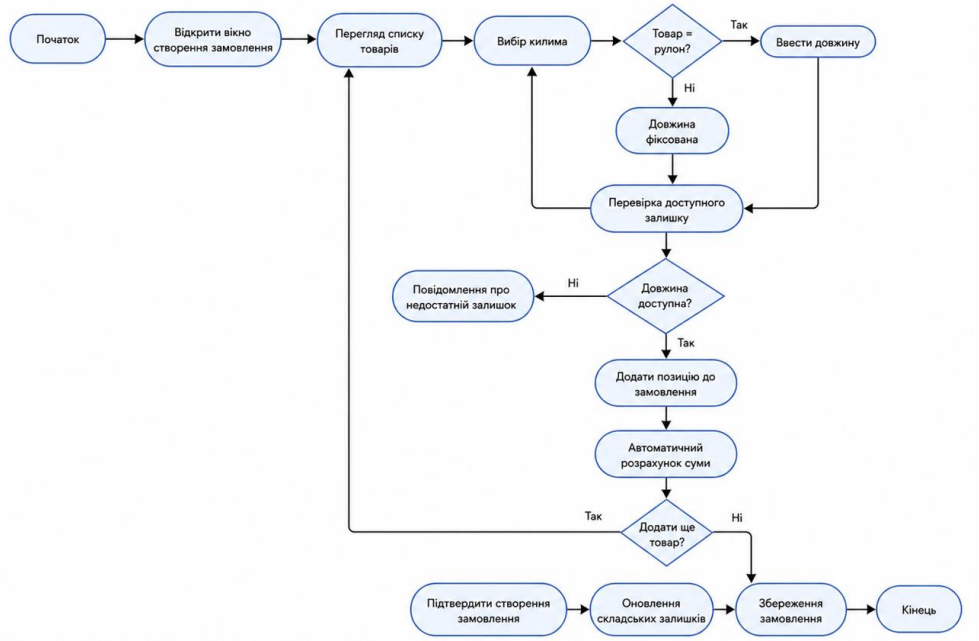


Рисунок 3.3 – User-flow створення замовлення в системі

Отже, спроєктована модель даних відображає основні сутності предметної області магазину килимів і зв’язки між ними. Поєднання моделі товарів із підсистемою замовлень дозволяє забезпечити створення замовлень із кількома позиціями, перевірку залишків, автоматичний розрахунок вартості, зміну статусів, скасування та відновлення.

3.3 Проєктування сервісного шару та механізму збереження даних

Сервісний шар є проміжною частиною між графічним інтерфейсом користувача, моделлю даних і механізмом збереження інформації. Його основне призначення полягає в обробці бізнес-логіки системи. Завдяки виділенню сервісного шару інтерфейс користувача не виконує складних операцій напряму, а лише передає запити до відповідних сервісів.

У розроблюваній системі сервісний шар представлений класами

InventoryService та OrderService. Клас InventoryService відповідає за роботу з товарами магазину, а OrderService — за створення, зміну та обробку замовлень. Такий поділ дозволяє окремо керувати логікою складського обліку та логікою замовлень.

InventoryService використовується для роботи з каталогом килимів. До його основних задач належать завантаження списку товарів, додавання нового килима, редагування характеристик наявного товару, отримання списку всіх позицій та збереження змін. Саме через цей сервіс графічний інтерфейс отримує дані для таблиці товарів і оновлює їх після додавання або редагування.

Окремо слід зазначити, що InventoryService забезпечує єдиний доступ до списку товарів для інших частин програми. Це дозволяє уникнути ситуацій, коли різні елементи інтерфейсу працюють із різними копіями даних. Усі зміни в каталозі товарів проходять через сервісний шар, після чого оновлені дані можуть бути відображені в таблиці та збережені у файлі.

Клас OrderService відповідає за логіку роботи із замовленнями. Він забезпечує створення нового замовлення, додавання позицій, перевірку доступної довжини товару, розрахунок загальної суми, зміну статусу, скасування та відновлення замовлення. Оскільки замовлення може містити кілька позицій, сервіс повинен коректно працювати зі списком об’єктів OrderItem.

Однією з найважливіших функцій OrderService є перевірка складських залишків. Під час додавання килима до замовлення сервіс перевіряє, чи достатньо доступної довжини товару. Якщо введена довжина перевищує залишок, позиція не додається до замовлення. Якщо перевірка успішна, сервіс додає позицію, перераховує суму замовлення та оновлює залишок товару.

Під час скасування замовлення OrderService змінює його статус на RETURNED і повертає товари на склад. Відновлення скасованого замовлення виконується лише після повторної перевірки залишків. Якщо потрібної кількості товару недостатньо, відновлення не дозволяється. Такий підхід запобігає помилкам у складському обліку.

Механізм збереження даних є важливою частиною програмного забезпечення, оскільки забезпечує збереження інформації про товари та замовлення між запусками програми. Для розроблюваної десктопної системи було обрано файлове збереження даних у форматі JSON. Такий підхід є достатнім для локального застосунку, який використовується продавцем на одному робочому місці та не потребує складної серверної інфраструктури.

У системі передбачено збереження двох основних груп даних: каталогу товарів і списку замовлень. Для цього використовуються окремі JSON-файли: inventory.json та orders.json. Файл inventory.json містить інформацію про килими, їх розміри, формат, матеріал, колір, тип і ціну. Файл orders.json зберігає дані про замовлення, позиції замовлень, загальну суму та статус.

Для роботи з файлами в проєкті передбачено пакет storage. Основними класами цього шару є InventoryStorage, OrderStorage та JsonStorageService. InventoryStorage відповідає за завантаження і запис списку товарів, OrderStorage виконує аналогічні операції для замовлень, а JsonStorageService використовується для серіалізації та десеріалізації даних у форматі JSON.

Винесення роботи з файлами в окремий пакет також спрощує подальший розвиток системи. Якщо в майбутньому виникне потреба замінити JSON-файли на базу даних, основні зміни будуть стосуватися саме шару збереження. При цьому сервісний шар і графічний інтерфейс можна буде змінювати мінімально.

Важливою вимогою до механізму збереження є узгодженість даних між товарами та замовленнями. Наприклад, після створення замовлення змінюється не лише список замовлень, а й залишок відповідного товару в каталозі. Тому після таких операцій потрібно зберігати оновлений стан як товарів, так і замовлень, щоб після повторного запуску програми дані залишалися актуальними.

Під час запуску програми дані з JSON-файлів завантажуються у відповідні об’єкти системи та передаються до сервісного шару. Він використовує їх для відображення у таблицях, створення замовлень, редагування товарів і перевірки залишків. Після дій, які змінюють дані, система зберігає оновлений стан назад у файли.

Отже, сервісний шар забезпечує основну логіку роботи системи та зв’язує графічний інтерфейс із моделлю даних і файловим сховищем. Поєднання сервісного шару з механізмом збереження даних дозволяє забезпечити коректну роботу з товарами та замовленнями, оновлення складських залишків і збереження актуального стану системи між запусками програми.

3.4 Проєктування графічного інтерфейсу користувача

Графічний інтерфейс користувача є основним засобом взаємодії продавця із системою. Оскільки програмне забезпечення призначене для щоденної роботи в магазині килимів, інтерфейс має бути простим, зрозумілим і орієнтованим на швидке виконання основних дій: перегляд товарів, створення замовлень, редагування даних і зміну статусів.

Головне вікно програми проєктується як основна робоча область продавця. У ньому передбачено дві таблиці: таблицю товарів і таблицю замовлень. Таблиця товарів містить дані про килими: ідентифікатор, назву, формат, розміри, тип, матеріал, колір і ціну. Таблиця замовлень відображає ідентифікатор, кількість позицій, загальну суму та поточний статус.

Така побудова головного вікна дозволяє продавцю одночасно контролювати два основні об’єкти системи: каталог товарів і список замовлень. Це зручно для роботи, оскільки під час оформлення продажу користувач може швидко порівняти інформацію про наявні килими, їх залишки та вже створені замовлення без переходу між окремими розділами програми.

У верхній частині вікна доцільно розмістити кнопки керування: оновлення даних, додавання й редагування товару, створення замовлення та вихід із програми. Це дозволяє продавцю швидко виконувати базові операції без переходу між складними меню.

Для зручності роботи з даними передбачаються фільтрація та сортування. Для товарів можна використовувати фільтри за форматом, типом, матеріалом і

кольором, а для замовлень — за статусом. Сортування за стовпцями дозволяє швидко впорядковувати товари або замовлення за потрібними параметрами.

Окремо проєктується вікно додавання та редагування товару. У ньому продавець заповнює основні характеристики килима: назву, ширину, довжину, матеріал, колір, тип і ціну. Для полів із фіксованими значеннями доцільно використовувати випадаючі списки.

Вікно створення замовлення повинно містити таблицю доступних товарів, поле для введення довжини, кнопку додавання позиції та список доданих товарів. Для шматків поле довжини блокується, оскільки вони продаються як готові залишки, а для рулонів продавець вводить потрібну довжину, після чого система перевіряє залишок.

Вікно деталей замовлення призначене для перегляду його складу та подальшої обробки. У ньому відображаються позиції замовлення, ціна, довжина й сума, а також передбачаються елементи для зміни статусу, редагування або видалення позицій, скасування, відновлення чи повного видалення замовлення.

Для запобігання випадковим змінам передбачено підтвердження критичних дій: видалення позиції, скасування або відновлення замовлення, повне видалення скасованого замовлення та зміну статусу. Підтвердження виконується через діалогове вікно з варіантами «ОК» і «Скасувати».

Важливою частиною інтерфейсу є візуальне виділення станів: нові замовлення можуть позначатися зеленим кольором, скасовані — червоним, а товари з малим залишком — окремим виділенням.

Під час проєктування інтерфейсу важливо врахувати послідовність дій продавця. Основні операції мають виконуватися без зайвих переходів між вікнами, щоб користувач міг швидко знайти товар, додати його до замовлення, перевірити залишок і завершити оформлення покупки. Такий підхід зменшує час роботи із системою та робить її зручною для щоденного використання.

Для введення даних доцільно використовувати не лише текстові поля, а й елементи вибору. Наприклад, матеріал, колір, тип товару та статус замовлення

краще подавати у вигляді випадаючих списків. Це зменшує кількість помилок під час введення та забезпечує однакове збереження значень у системі.

Окрему увагу потрібно приділити повідомленням для користувача. Якщо продавець вводить некоректну довжину, залишку недостатньо або не вибрано товар, система повинна показати зрозуміле повідомлення про помилку. Такі повідомлення допомагають швидко виправити дію без додаткового пошуку причини помилки.

Також інтерфейс має забезпечувати своєчасне оновлення даних після виконання операцій. Після додавання товару, створення замовлення, зміни статусу або скасування замовлення таблиці повинні оновлюватися, щоб продавець одразу бачив актуальний стан каталогу, замовлень і складських залишків.

Отже, спроєктований інтерфейс забезпечує зручну й безпечну роботу продавця з товарами та замовленнями завдяки таблицям, діалоговим вікнам, фільтрам, сортуванню, кольоровому виділенню та підтвердженню дій.

3.5 Висновки по розділу

У третьому розділі було виконано проєктування програмного забезпечення для магазину килимів. Визначено загальну архітектуру системи та її поділ на основні модулі: model, service, storage, ui і data.

Було спроєктовано модель даних предметної області, зокрема класи для опису килимів, їх характеристик, замовлень, позицій замовлення та статусів. Окремо розглянуто підсистему замовлень, яка забезпечує створення замовлень із кількома позиціями, перевірку залишків, розрахунок вартості, скасування та відновлення.

Також було описано сервісний шар системи, механізм збереження даних у JSON-файлах і структуру графічного інтерфейсу користувача. Запропоновані проєктні рішення створюють основу для практичної реалізації програмного забезпечення.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ПРОЄКТУ

4.1 Реалізація структури Java-проєкту та моделі даних

Реалізація програмного забезпечення для магазину килимів виконана у вигляді десктопного Java-застосунку з графічним інтерфейсом Swing. Для зручної організації коду проєкт поділено на окремі пакети, кожен із яких відповідає за певну частину функціональності системи. Такий підхід дозволяє відокремити модель даних, бізнес-логіку, механізм збереження інформації та графічний інтерфейс користувача.

Фактична структура реалізованого Java-проєкту в середовищі IntelliJ IDEA наведена на рисунку 4.1.

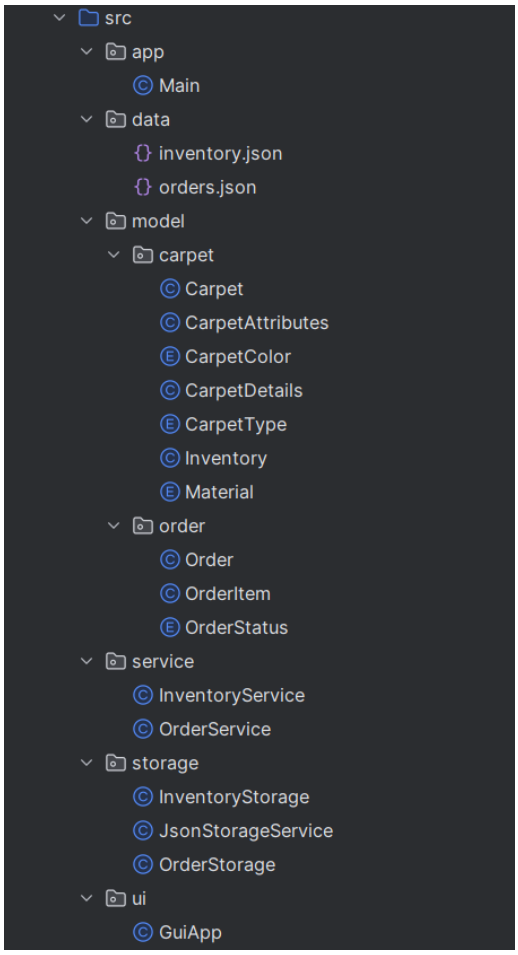


Рисунок 4.1 – Структура Java-проєкту в середовищі IntelliJ IDEA

У структурі проєкту передбачено пакети app, model, service, storage, ui, а також каталог data, у якому розміщуються JSON-файли з даними. Пакет app містить головний клас Main, який є точкою входу в програму. У ньому створюються необхідні сервіси, завантажуються дані та запускається графічний інтерфейс користувача.

Пакет model містить класи предметної області та поділяється на два підпакети: model.carpet і model.order. У підпакеті model.carpet реалізовано класи для опису товарів магазину: Carpet, CarpetDetails, CarpetAttributes, Inventory, а також перелічувані типи Material, CarpetColor і CarpetType. Вони використовуються для збереження інформації про килим, його розміри, характеристики, формат і належність до певного типу.

Головним класом для опису товару є Carpet. Він містить ідентифікатор, назву, розмірні характеристики, атрибути та ціну за метр. Ідентифікатор використовується для пошуку товару в системі, назва відображається у таблиці товарів, а ціна застосовується під час розрахунку вартості позиції замовлення.

Для зберігання фізичних характеристик товару використовується клас CarpetDetails. Він містить ширину, довжину та ознаку того, чи є килим рулоном. У системі ширина і довжина зберігаються в сантиметрах, що дозволяє уникнути неточностей під час розрахунків. Для зручності користувача ці значення можуть відображатися в інтерфейсі у метрах. Для рулонів поле довжини означає доступний залишок товару, а для шматків — фіксовану довжину готового залишку.

Атрибути товару винесено в клас CarpetAttributes. Він містить матеріал, колір і тип килима. Для цих характеристик використовуються перелічувані типи Material, CarpetColor і CarpetType. Такий підхід зменшує ризик помилок при введенні даних, оскільки користувач обирає значення з визначеного набору, а не вводить їх вручну.

Для зберігання списку товарів використовується клас Inventory, який містить колекцію об’єктів Carpet. Через нього система працює з каталогом товарів як з єдиною структурою. У процесі роботи програми цей список використовується

для відображення товарів у таблиці, пошуку килима за ідентифікатором, додавання нових позицій, редагування характеристик і перевірки залишків.

У підпакеті model.order реалізовано класи Order, OrderItem і перелічуваний тип OrderStatus. Клас Order описує замовлення та містить список позицій, загальну суму і поточний статус. Клас OrderItem представляє окрему позицію замовлення, тобто зв’язок між вибраним килимом, довжиною продажу та розрахованою вартістю. OrderStatus визначає можливі стани замовлення: нове, прийняте в роботу, готове до відправки, відправлене, отримане та повернуте або скасоване.

Пакет service містить класи сервісного шару: InventoryService і OrderService. Вони відповідають за основну логіку роботи з товарами та замовленнями. Пакет storage призначений для роботи з файлами та містить класи InventoryStorage, OrderStorage і JsonStorageService, які забезпечують збереження й завантаження даних у форматі JSON. Пакет ui містить клас GuiApp, у якому реалізовано графічний інтерфейс користувача.

Каталог data містить два основні файли: inventory.json і orders.json. У файлі inventory.json зберігається інформація про товари магазину, а у файлі orders.json — дані про створені замовлення. Для роботи з цими файлами використовується бібліотека Gson, яка дозволяє перетворювати Java-об’єкти у JSON-структуру та відновлювати їх під час запуску програми.

Загальна логіка роботи застосунку полягає в тому, що під час запуску програми дані завантажуються з JSON-файлів, передаються до сервісного шару та відображаються в інтерфейсі. Після додавання товару, створення замовлення, редагування або зміни статусу відповідні дані оновлюються і зберігаються назад у JSON-файли.

Фрагменти реалізації основних класів моделі даних наведено **додатку Б.**

Отже, реалізована структура Java-проєкту забезпечує чіткий поділ відповідальності між частинами системи. Модель описує основні дані предметної області, сервісний шар відповідає за бізнес-логіку, сховище працює з файлами, а графічний інтерфейс забезпечує взаємодію продавця із системою. Такий підхід

робить програму зрозумілою, підтримуваною та придатною до подальшого розвитку.

4.2 Реалізація обліку товарів і механізму роботи із замовленнями

У програмному забезпеченні для магазину килимів основою складського обліку є класи, що описують товар, його характеристики та список наявних позицій. Для цього в проєкті використовується пакет model.carpet, у якому реалізовано класи Carpet, CarpetDetails, CarpetAttributes, Inventory, а також перелічувані типи Material, CarpetColor і CarpetType.

Головним класом для опису товару є Carpet. Він містить ідентифікатор, назву, розмірні характеристики, атрибути та ціну за метр. Ідентифікатор використовується для пошуку товару в системі, назва відображається у таблиці товарів, а ціна застосовується під час розрахунку вартості позиції замовлення.

Для зберігання фізичних характеристик товару використовується клас CarpetDetails. Він містить ширину, довжину та ознаку того, чи є килим рулоном. У системі ширина і довжина зберігаються в сантиметрах, що дозволяє уникнути неточностей під час розрахунків. Для рулонів поле довжини означає доступний залишок товару, а для шматків — фіксовану довжину готового залишку.

Атрибути товару винесено в клас CarpetAttributes. Він містить матеріал, колір і тип килима. Для цих характеристик використовуються перелічувані типи Material, CarpetColor і CarpetType. Такий підхід зменшує ризик помилок при введенні даних, оскільки користувач обирає значення з визначеного набору.

Для зберігання списку товарів використовується клас Inventory, який містить колекцію об’єктів Carpet. Через нього система працює з каталогом товарів як з єдиною структурою. Цей список використовується для відображення товарів у таблиці, пошуку килима за ідентифікатором, додавання нових позицій, редагування характеристик і перевірки залишків.

Особливістю реалізації є врахування формату товару. Рулон передбачає продаж частини товару за заданою довжиною, тому після продажу його залишок

повинен зменшуватися. Шматок є готовим залишком фіксованого розміру, тому його довжина не змінюється довільно під час оформлення замовлення.

Механізм роботи із замовленнями реалізовано в пакеті model.order, який містить класи Order, OrderItem та перелічуваний тип OrderStatus. Клас Order описує замовлення покупця і містить ідентифікатор, список позицій, загальну суму та поточний статус. Список позицій реалізовано як колекцію об’єктів OrderItem, що дозволяє додавати до одного замовлення кілька різних килимів.

Клас OrderItem використовується для опису окремої позиції замовлення. Він містить посилання на конкретний килим і довжину, яка продається. На основі цих даних розраховується вартість позиції за формулою:

вартість позиції = довжина у метрах × ціна за метр.

Загальна сума замовлення формується як сума вартостей усіх його позицій. Це дозволяє продавцю одразу бачити актуальну вартість покупки та зменшує потребу в ручних розрахунках.

Під час додавання товару до замовлення враховується формат килима. Якщо товар є рулоном, продавець може вказати потрібну довжину, після чого система перевіряє, чи не перевищує вона фактичний залишок. Якщо доступної довжини недостатньо, позиція не додається до замовлення. Якщо товар є шматком, його довжина вважається фіксованою.

Для відображення етапу обробки замовлення використовується перелічуваний тип OrderStatus. У системі передбачено статуси NEW, ACCEPTED, READY, SENT, RECEIVED і RETURNED. В інтерфейсі ці значення відображаються українською мовою: «Нове», «Прийнято в роботу», «Готове до відправки», «Відправлено», «Отримано/Закрито» та «Повернуто/Скасовано».

Окремо реалізовано логіку скасування та відновлення замовлення. У разі скасування статус змінюється на RETURNED, а товари, які входили до замовлення, повертаються на склад. Відновлення можливе лише після перевірки наявності потрібної довжини товару. Якщо залишку недостатньо, система не дозволяє відновити таке замовлення.

Також передбачено можливість редагування деталей замовлення. Продавець може переглядати список позицій, змінювати довжину для рулонних товарів або видаляти окремі позиції. Якщо після видалення всіх позицій замовлення стає порожнім, воно не повинно залишатися у списку активних замовлень.

Отже, реалізовані класи товарів і замовлень забезпечують облік килимів, поділ товарів на рулони та шматки, перевірку доступної довжини, автоматичний розрахунок вартості, зміну статусів, скасування та відновлення замовлень. Така реалізація дозволяє автоматизувати основні процеси складського обліку й оформлення продажу в магазині килимів.

Фрагменти реалізації основних класів моделі даних наведено **додатку Б.**

4.3 Реалізація сервісного шару та збереження даних у JSON

Сервісний шар у розробленому програмному забезпеченні відповідає за виконання основної бізнес-логіки системи. Він є проміжною частиною між графічним інтерфейсом, моделлю даних і механізмом збереження інформації. Завдяки цьому інтерфейс користувача не працює напряму з файлами або складними операціями обробки даних, а звертається до відповідних сервісів.

У проєкті сервісний шар представлений двома основними класами: InventoryService та OrderService. Клас InventoryService відповідає за роботу з товарами магазину, а OrderService реалізує логіку роботи із замовленнями. Такий поділ дозволяє окремо керувати обліком товарів і процесом створення та обробки замовлень.

InventoryService забезпечує отримання списку килимів, додавання нового товару, редагування наявного товару та збереження змін. Під час запуску програми сервіс отримує дані зі сховища, після чого передає їх у графічний інтерфейс для відображення в таблиці товарів. Якщо продавець додає або змінює товар, сервіс оновлює відповідний список і передає зміни до шару збереження.

Клас OrderService відповідає за створення та обробку замовлень. Він

забезпечує створення нового замовлення, додавання позицій, розрахунок загальної суми, зміну статусу, скасування, відновлення та видалення замовлень. Під час додавання позиції сервіс перевіряє, чи достатньо доступної довжини товару. Якщо залишку недостатньо, позиція не додається, що запобігає помилкам у складському обліку.

Після успішного додавання позиції OrderService оновлює суму замовлення та змінює залишок відповідного товару. Для рулонів зменшується доступна довжина, а для шматків використовується їх фіксований розмір. У разі скасування замовлення сервіс повертає товари на склад і змінює статус замовлення на «Повернуто/Скасовано». Відновлення скасованого замовлення виконується лише після повторної перевірки наявності потрібної довжини товару.

Для збереження інформації між запусками програми в системі використовується формат JSON. Такий підхід є зручним для локального десктопного застосунку, оскільки не потребує встановлення бази даних або серверної частини. Дані зберігаються у звичайних файлах, які завантажуються під час запуску програми та оновлюються після виконання дій користувача.

У системі використано два основні файли даних: inventory.json і orders.json. Файл inventory.json призначений для збереження інформації про товари магазину, зокрема назви килимів, їх розмірів, формату, матеріалу, кольору, типу та ціни. Файл orders.json містить інформацію про замовлення, їх позиції, загальну суму та поточний статус.

Робота з файлами винесена в окремий пакет storage. У ньому реалізовано класи InventoryStorage, OrderStorage і JsonStorageService. Клас InventoryStorage відповідає за завантаження та збереження списку товарів, OrderStorage — за роботу зі списком замовлень, а JsonStorageService виконує допоміжні операції серіалізації та десеріалізації даних у форматі JSON.

Для роботи з JSON-файлами використовується бібліотека Gson. Вона дозволяє перетворювати Java-об’єкти у JSON-структуру та виконувати зворотну операцію — відновлювати об’єкти з JSON-файлів. Завдяки цьому об’єкти класів

Carpet, Order, OrderItem та інших можуть бути збережені у файлі без ручного формування текстової структури JSON.

Під час запуску програми дані з файлів зчитуються і передаються до сервісного шару. Після цього вони відображаються в інтерфейсі користувача у вигляді таблиць товарів і замовлень. Якщо продавець додає новий товар, редагує наявний, створює замовлення або змінює його статус, відповідні дані оновлюються в пам'яті програми та зберігаються назад у JSON-файли. Фрагмент JSON-структури для збереження товарів і замовлень наведено на рисунку 4.2.

Фрагмент inventory.json

Фрагмент orders.json

Рисунок 4.2 – Фрагмент JSON-структури для збереження товарів і замовлень

Отже, реалізація сервісного шару та механізму збереження даних забезпечує зв'язок між інтерфейсом, моделлю даних і файловим сховищем. Класи InventoryService та OrderService відповідають за бізнес-логіку, а InventoryStorage, OrderStorage і JsonStorageService забезпечують збереження та завантаження даних у JSON. Такий підхід робить систему зрозумілою, підтримуваною та придатною до подальшого розширення.

Фрагменти реалізації основних класів моделі даних наведено **додатку Б.**

4.4 Реалізація графічного інтерфейсу Swing

Графічний інтерфейс програмного забезпечення реалізовано засобами бібліотеки Swing. Основним класом є GuiApp, у якому створюються головне вікно, таблиці товарів і замовлень, кнопки керування та діалогові вікна.

Головне вікно створюється на основі JFrame. Для розміщення елементів використовується BorderLayout: у верхній частині розташовано кнопки оновлення даних, додавання й редагування товару, створення замовлення та виходу з програми.

Основна частина вікна містить дві таблиці: каталог товарів і список замовлень. Для їх розміщення використано JSplitPane, який поділяє робочу область на частину з товарами та частину із замовленнями.

Таблиця товарів реалізована через JTable і DefaultTableModel. У ній відображаються ідентифікатор, назва, формат, розміри, тип, матеріал, колір і ціна килима. Таблиця замовлень містить ідентифікатор, кількість позицій, загальну суму та статус, який відображається українською мовою.

Фактичний вигляд головного вікна програми наведено на рисунку 4.3.

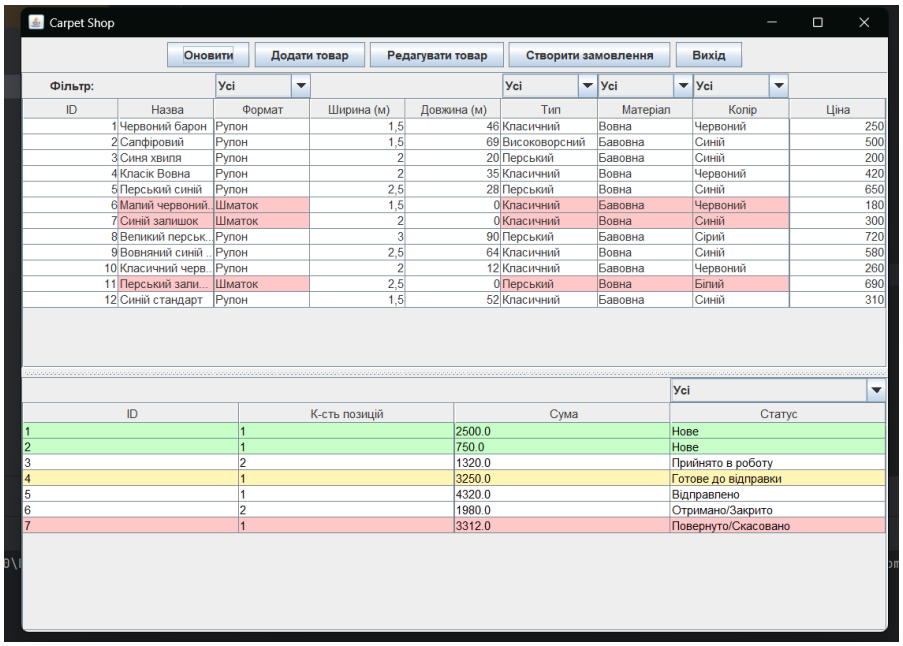


Рисунок 4.3 – Інтерфейс головного вікна програми з таблицями товарів і замовлень

Для додавання нового товару використовується модальне діалогове вікно JDialog. У ньому розміщені поля для введення назви, ширини, довжини та ціни, а також випадаючі списки для вибору матеріалу, кольору й типу килима. Після підтвердження введені дані передаються до InventoryService, додаються до списку товарів і зберігаються у JSON-файлі.

Фактичний вигляд діалогового вікна наведено на рисунку 4.4.

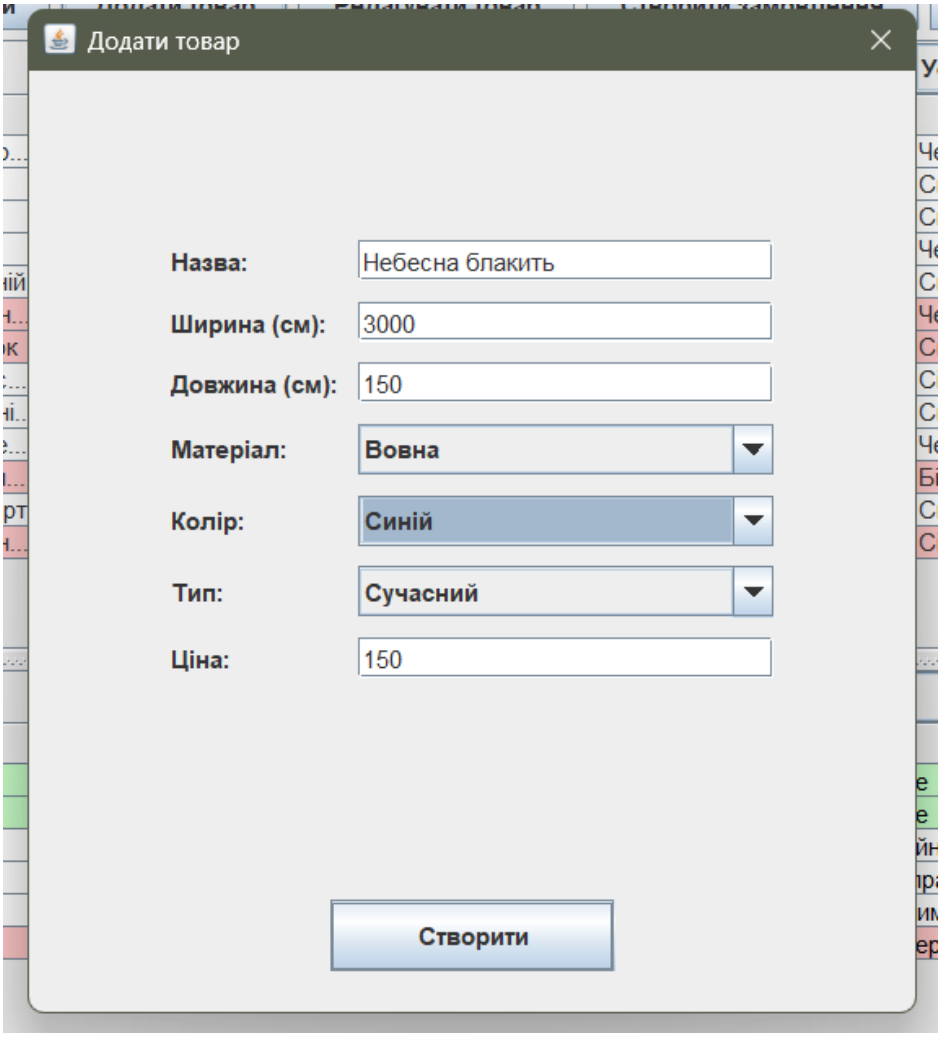


Рисунок 4.4 – Діалогове вікно додавання нового товару

Крім додавання нових товарів, у системі реалізовано функціонал редагування вже існуючих позицій каталогу. Для цього користувач обирає потрібний товар у таблиці та натискає кнопку редагування. Після цього відкривається діалогове вікно, у якому відображаються поточні характеристики

обраного килима. Продавець може змінювати назву, розміри, ціну, матеріал, колір, тип і формат товару. Вигляд вікна наведено на рисунку 4.5.

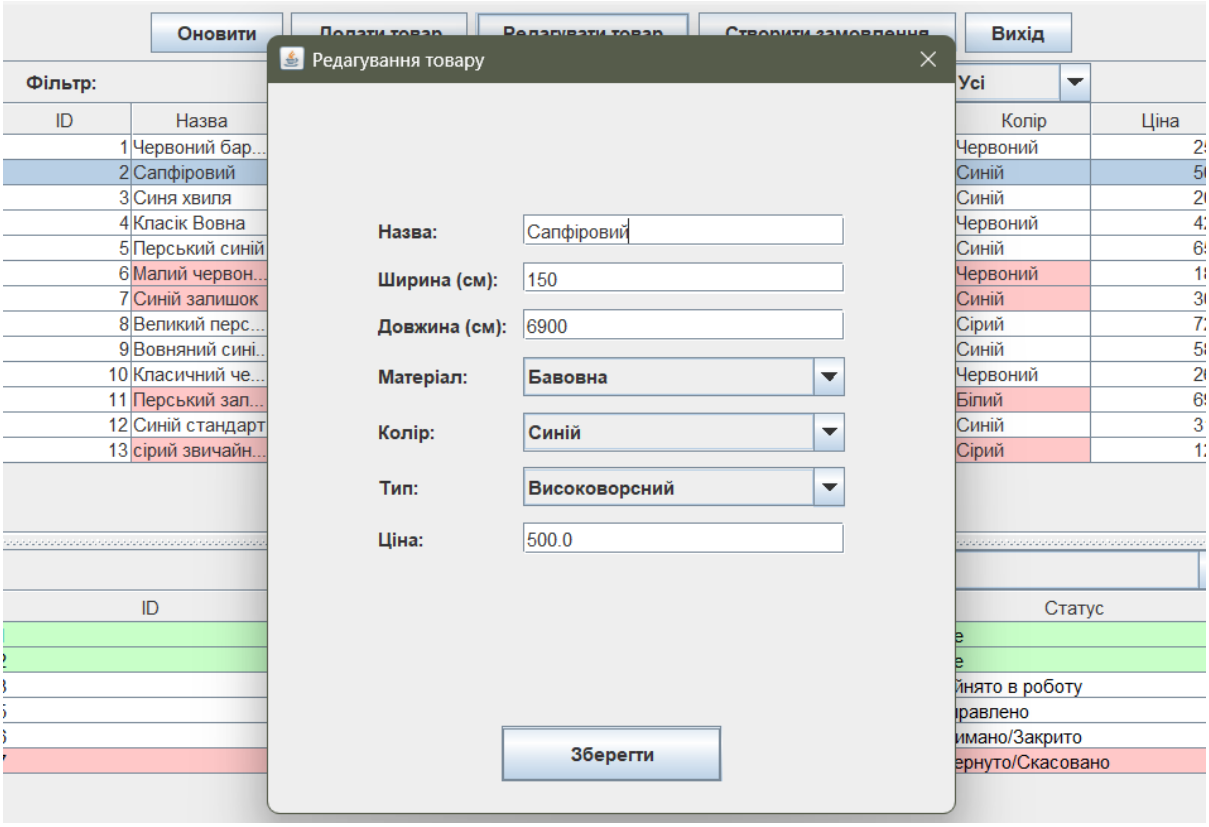


Рисунок 4.5 – Діалогове вікно редагування вже існуючих позицій каталогу

Перед збереженням змін система виконує перевірку коректності введених даних. Не допускається введення від’ємних або нульових значень для розмірів і ціни, а також порожньої назви товару. Після успішного підтвердження змін оновлені дані передаються до сервісного шару, зберігаються у файлі inventory.json і автоматично відображаються в таблиці товарів. Такий підхід забезпечує актуальність інформації про складські позиції та спрощує підтримку каталогу товарів у належному стані.

Окремо реалізовано вікно створення замовлення. У ньому продавець бачить список доступних товарів, обирає потрібний килим, вводить довжину для рулонного товару та додає позицію до замовлення. Якщо вибраний товар є шматком, поле довжини блокується, оскільки такий товар продається як готовий залишок. Вигляд цього вікна наведено на рисунку 4.6.

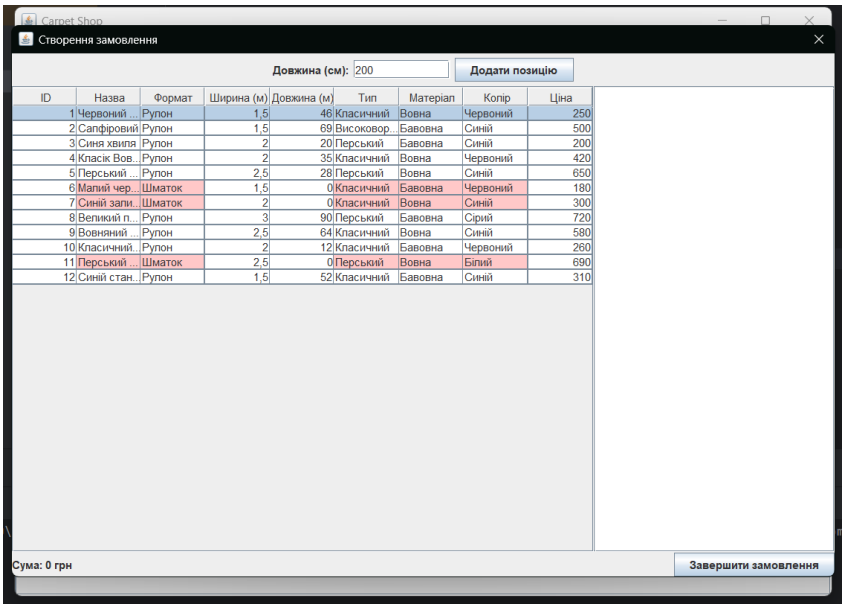


Рисунок 4.6 – Вікно створення замовлення з вибором килима та перевіркою залишку

Також реалізовано вікно деталей замовлення. У ньому відображається список позицій, їх довжина, ціна за метр і сума. Продавець може змінити статус замовлення, відредагувати позицію, видалити її або скасувати замовлення. Для скасованих замовлень додатково доступні дії відновлення та повного видалення. Вигляд вікна деталей замовлення наведено на рисунку 4.7.

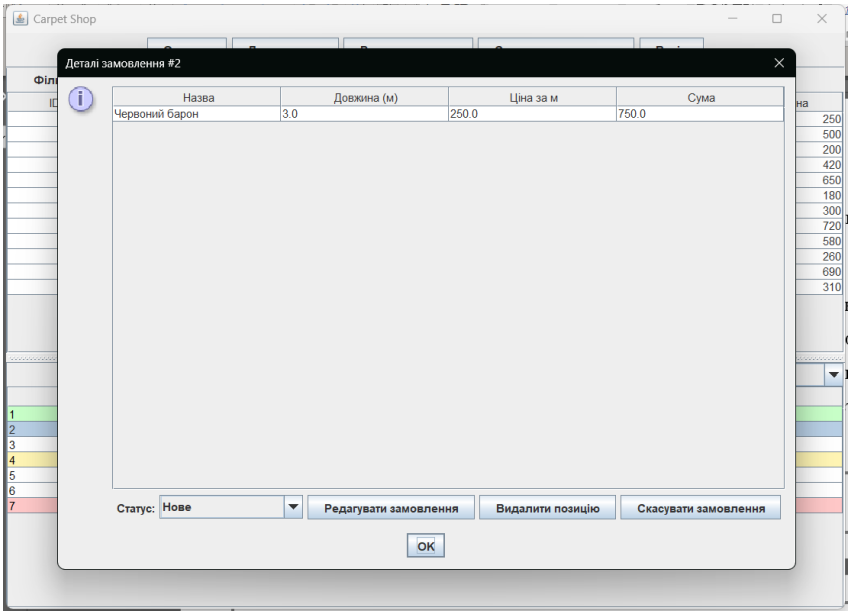


Рисунок 4.7 – Вікно деталей замовлення зі редагуванням та скасуванням замовлення

Важливою частиною інтерфейсу є підтвердження критичних дій. Перед видаленням позиції, скасуванням замовлення, відновленням або повним видаленням система показує діалогове вікно з варіантами підтвердження. Якщо користувач натискає «Скасувати» або закриває повідомлення, дія не виконується. Це зменшує ризик випадкової зміни або втрати даних.

Отже, графічний інтерфейс Swing забезпечує основні можливості взаємодії продавця із системою: перегляд товарів, створення та обробку замовлень, редагування даних, фільтрацію, сортування та підтвердження важливих дій. Реалізація інтерфейсу у вигляді таблиць і діалогових вікон робить програму зручною для щоденного використання в магазині килимів.

4.5 Реалізація створення замовлень і додаткових можливостей інтерфейсу

Створення та редагування замовлень є одним із ключових процесів програмного забезпечення. Цей механізм забезпечує вибір товару, перевірку залишку, додавання позицій, автоматичний розрахунок суми та оновлення складських даних.

Процес створення замовлення реалізовано через окреме діалогове вікно. У ньому продавець бачить таблицю доступних килимів, поле для введення довжини та кнопку додавання позиції. Якщо товар є рулоном, поле довжини доступне для введення. Якщо товар є шматком, довжина встановлюється автоматично, а поле введення блокується.

Під час додавання позиції система перевіряє вибір товару, коректність введеної довжини та доступний залишок. Якщо введена довжина перевищує наявну кількість, позиція не додається, а користувач отримує повідомлення про недостатній залишок. Це запобігає створенню замовлення на товар, якого фактично немає на складі. Вигляд повідомлення про недостатній залишок наведено на рисунку 4.8.

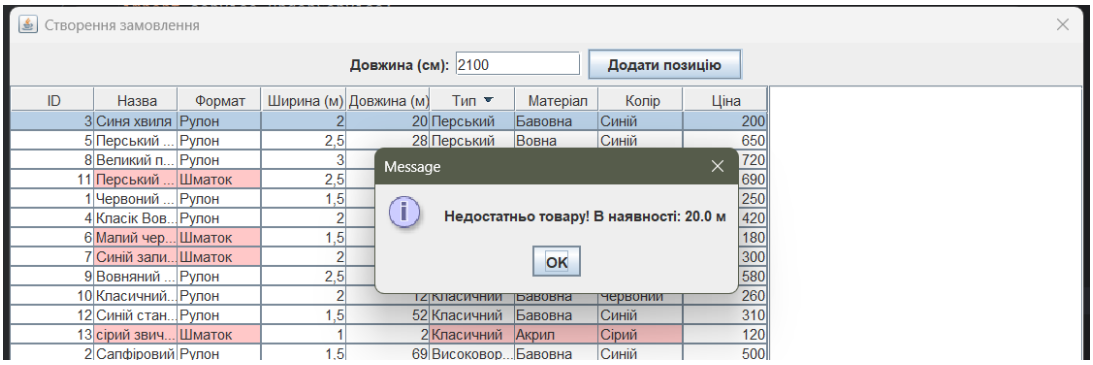


Рисунок 4.8 – Повідомлення про недостатній залишок товару на складі

Замовлення створюється лише після додавання позицій і підтвердження користувачем. Якщо продавець відкрив вікно, але не додав жодної позиції або не завершив створення, замовлення не зберігається. Такий підхід не допускає появи порожніх замовлень у списку.

Після додавання позиції програма розраховує її вартість за формулою:

вартість позиції = довжина у метрах × ціна за метр.

Далі значення додається до загальної суми замовлення, а позиція відображається у списку доданих товарів. Після завершення система зберігає замовлення, оновлює список замовлень і змінює складські залишки відповідних товарів.

Редагування замовлення виконується у вікні деталей. У ньому відображається список позицій із назвою килима, довжиною, ціною за метр і сумою. Продавець може змінити довжину лише для рулонного товару, оскільки шматки продаються як готові залишки фіксованого розміру.

Під час редагування система повторно перевіряє доступний залишок, враховуючи різницю між старою та новою довжиною. Якщо товару недостатньо, зміна не виконується. Якщо перевірка успішна, система оновлює довжину позиції, її суму, загальну суму замовлення та складський залишок.

У вікні деталей також реалізовано видалення окремих позицій. Перед видаленням система виводить діалогове вікно підтвердження. Після підтвердження позиція видаляється, а відповідна довжина товару повертається на склад. Якщо замовлення стає порожнім, воно не залишається у списку активних

замовлень. Вигляд діалогового вікна наведено на рисунку 4.9.

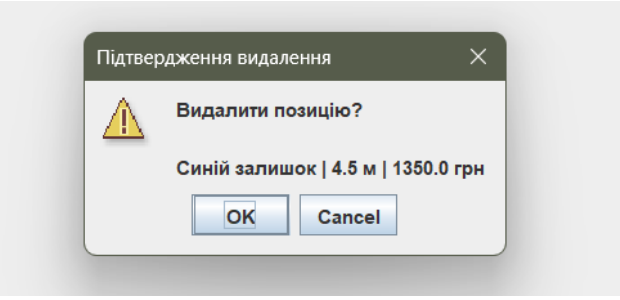


Рисунок 4.9 – Діалогове вікно підтвердження видалення позиції з замовлення

Окремо реалізовано зміну статусу замовлення. Продавець може обрати новий статус зі списку: нове, прийняте в роботу, готове до відправки, відправлене або отримане. У разі скасування статус змінюється на «Повернуто/Скасовано», а всі товари повертаються на склад. Вигляд вікна деталей замовлення зі статусом «Повернуто/Скасовано» наведено на рисунку 4.10. Для скасованих замовлень передбачено відновлення з повторною перевіркою залишків. Вигляд повідомлення про недостатній залишок товару на складі при спробі відновлення скасованого замовлення наведено на рисунку 4.11.

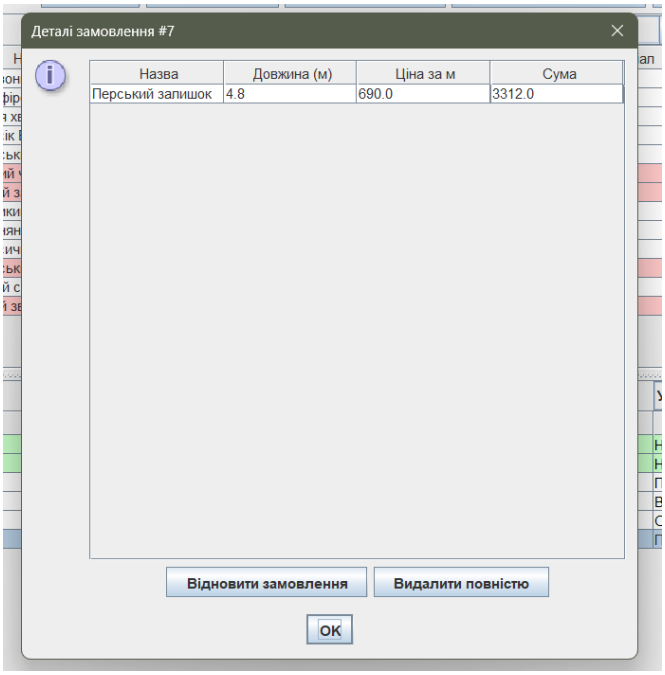


Рисунок 4.10 – Вікно деталей замовлення зі статусом «Повернуто/Скасовано»

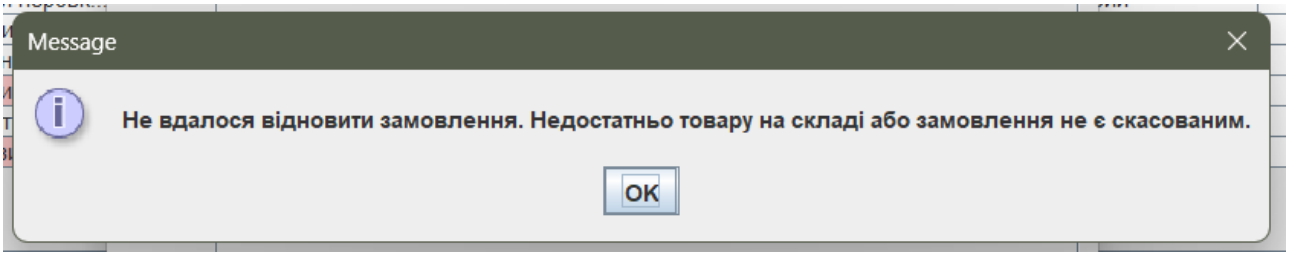
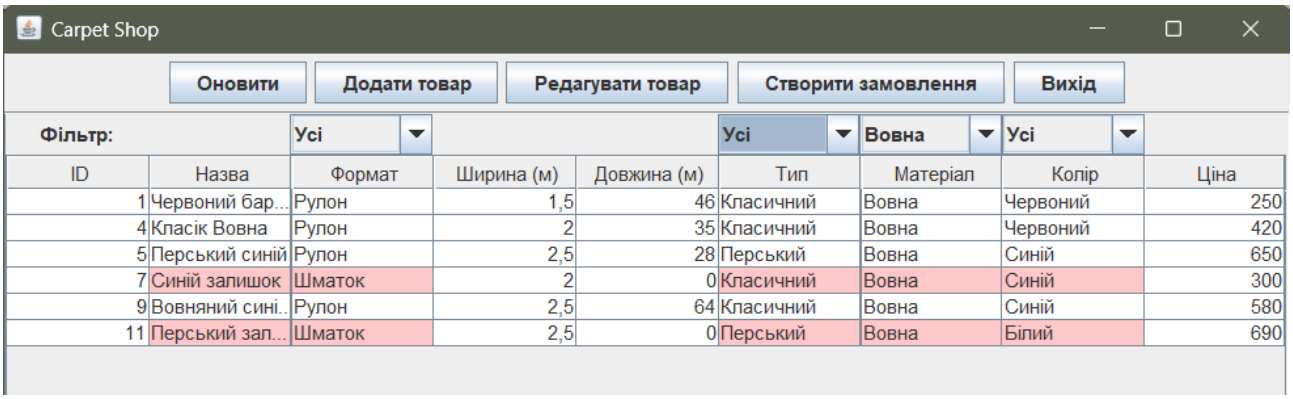


Рисунок 4.11 – Повідомлення про недостатній залишок товару на складі при спробі відновлення скасованого замовлення

Окрім основних функцій роботи із замовленнями, у програмному забезпеченні реалізовано додаткові можливості інтерфейсу: фільтрацію, сортування, кольорове виділення станів, блокування прямого редагування таблиць і підтвердження критичних дій.

Для швидкого пошуку даних реалізовано фільтрацію. У таблиці товарів продавець може відбирати килими за форматом, типом, матеріалом і кольором, а в таблиці замовлень — за статусом. Вигляд реалізації цього функціоналу наведено на рисунку 4.12.



ID	Назва	Формат	Ширина (м)	Довжина (м)	Тип	Матеріал	Колір	Ціна
1	Червоний бар...	Рулон	1,5	46	Класичний	Вовна	Червоний	250
4	Класік Вовна	Рулон	2	35	Класичний	Вовна	Червоний	420
5	Перський синій	Рулон	2,5	28	Перський	Вовна	Синій	650
7	Синій залишок	Шматок	2	0	Класичний	Вовна	Синій	300
9	Вовняний сині..	Рулон	2,5	64	Класичний	Вовна	Синій	580
11	Перський зал...	Шматок	2,5	0	Перський	Вовна	Білий	690

Рисунок 4.12 – Фільтрація даних у таблицях товарів і замовлень

Також у таблицях реалізовано сортування за основними стовпцями. Продавець може впорядковувати товари та замовлення за ідентифікатором, назвою, сумою, кількістю позицій або статусом, що спрощує навігацію в системі.

Для покращення сприйняття даних використовується кольорове виділення рядків. Нові замовлення позначаються зеленим кольором, скасовані — червоним,

а товари з малим залишком можуть виділятися окремо для швидкого контролю. Вигляд замовлень з різними статусами наведено на рисунку 4.13.

ID	К-сть позицій	Сума	Статус
1	1	2500.0	Нове
2	1	750.0	Повернуто/Скасовано
3	2	1320.0	Прийнято в роботу
4	1	3250.0	Готове до відправки

Рисунок 4.13 – Кольорове виділення статусів замовлень у таблиці

Для запобігання випадковому редагуванню даних у таблицях заблоковано зміну значень безпосередньо в комірках. Редагування товарів або позицій замовлення виконується через окремі діалогові вікна, що дозволяє перевіряти дані перед збереженням.

Окремо реалізовано підтвердження дій, які змінюють або видаляють дані. Перед скасуванням замовлення, видаленням позиції, відновленням або повним видаленням скасованого замовлення система показує діалогове вікно з варіантами «ОК» і «Скасувати». Якщо користувач натискає «Скасувати» або закриває вікно, дія не виконується. Вигляд вікна наведено на рисунку 4.14.

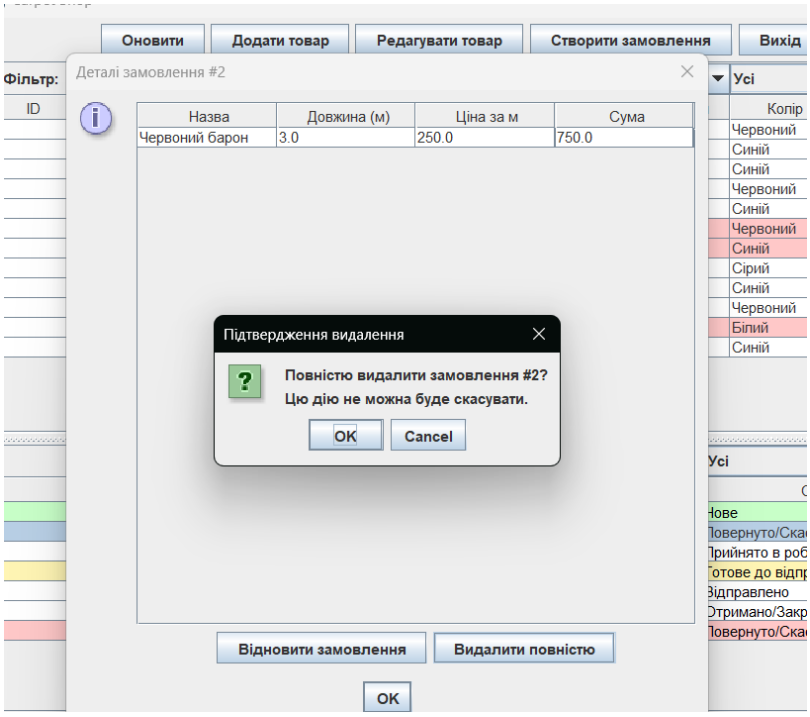


Рисунок 4.14 – Діалогове вікно підтвердження критичної дії

Після додавання товару, створення замовлення, зміни статусу або редагування даних таблиці автоматично оновлюються. Це дозволяє користувачу одразу бачити актуальний стан товарів, замовлень і складських залишків.

Отже, реалізація створення замовлень і додаткових можливостей інтерфейсу забезпечує зручну та безпечну роботу продавця. Перевірка залишків, автоматичний розрахунок вартості, фільтрація, сортування, кольорове виділення, блокування прямого редагування й підтвердження критичних дій зменшують імовірність помилок під час щоденної роботи.

4.6 Висновки по розділу

У четвертому розділі було описано реалізацію програмного забезпечення для магазину килимів. Розглянуто структуру Java-проєкту, основні пакети, класи моделі товарів і замовлень, сервісний шар, механізм збереження даних у JSON-файлах та графічний інтерфейс користувача.

Було реалізовано роботу з каталогом товарів, створення й редагування замовлень, перевірку доступної довжини, автоматичний розрахунок вартості, оновлення складських залишків і зміну статусів замовлень. Для збереження інформації використано файли inventory.json і orders.json, а для взаємодії продавця із системою — інтерфейс Swing.

Також реалізовано додаткові можливості інтерфейсу: фільтрацію, сортування, кольорове виділення станів, блокування прямого редагування таблиць і підтвердження критичних дій. У результаті створено десктопний Java-застосунок, який забезпечує основні процеси обліку товарів і замовлень у магазині килимів.

РОЗДІЛ 5. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ

5.1 Стратегія тестування програмного забезпечення

Тестування програмного забезпечення є важливим етапом розробки, оскільки дозволяє перевірити коректність роботи системи, виявити помилки та переконатися, що реалізовані функції відповідають поставленим вимогам. Для розробленого десктопного застосунку магазину килимів тестування спрямоване на перевірку роботи з товарами, замовленнями, складськими залишками, статусами та збереженням даних у JSON-файлах.

Оскільки система є локальним Java-застосунком з графічним інтерфейсом Swing, основним підходом обрано ручне функціональне тестування. Воно передбачає перевірку основних сценаріїв роботи користувача через інтерфейс програми: додавання товару, редагування характеристик килима, створення замовлення, додавання позицій, зміну статусу, скасування та відновлення замовлення.

Метою тестування є підтвердження того, що програма правильно виконує повсякденні дії продавця та не допускає некоректної роботи з даними. Особливу увагу приділено перевірці ситуацій, які впливають на складські залишки: продаж частини рулону, спроба продажу більшої довжини, ніж є в наявності, скасування замовлення та повторне його відновлення.

Під час тестування враховувалися як звичайні сценарії роботи, так і помилкові дії користувача. Перевірялося додавання товарів із коректними даними, створення замовлень з однією або кількома позиціями, зміна статусів, а також введення порожніх полів, некоректної ціни чи довжини. Це дозволило оцінити не лише правильність основних функцій, а й захист системи від помилок користувача.

Перелік груп функцій що перевірялися під час тестування наведено в таблиці 5.1

Таблиця 5.1

Основні напрями тестування програмного забезпечення

Напрям тестування	**Що перевіряється**	**Очікуваний результат**
Робота з товарами	Додавання, редагування, фільтрація, сортування	Товари коректно відображаються та зберігаються
Створення замовлень	Додавання однієї або кількох позицій	Замовлення створюється без помилок
Перевірка залишків	Введення довжини більшої за доступну	Система забороняє додавання позиції
Статуси замовлень	Зміна статусів замовлення	Статус оновлюється і зберігається
Скасування та відновлення	Повернення і повторне списання товару	Залишки оновлюються коректно
JSON-збереження	Закриття і повторний запуск програми	Дані відновлюються з файлів

Для перевірки системи використовувалися тестові дані, що включали килими різних типів, матеріалів, кольорів і форматів. Частина товарів зберігалася як рулони з великою довжиною, інша частина — як шматки з фіксованим розміром. Це дозволило перевірити роботу програми в різних ситуаціях і виявити можливі помилки під час створення замовлень.

Результати тестування доцільно подати у вигляді таблиць із зазначенням тестового сценарію, вхідних даних, очікуваного результату та фактичного результату. Такий підхід дозволяє систематизувати перевірку й показати, що основні функції програмного забезпечення працюють коректно.

Отже, стратегія тестування розробленого застосунку базується на перевірці основних сценаріїв роботи продавця магазину килимів і контролі можливих помилкових дій. Такий підхід дозволяє оцінити правильність реалізації функціональних вимог, коректність обробки даних і готовність системи до практичного використання.

5.2 Тестування роботи з товарами та створення замовлень

Тестування роботи з товарами та створення замовлень було спрямоване на перевірку основних функцій системи, пов’язаних із каталогом килимів і оформленням покупки. Оскільки каталог товарів є основою складського обліку, важливо переконатися, що всі характеристики килимів зберігаються коректно, а зміни одразу відображаються в інтерфейсі програми.

Під час тестування перевірялося додавання нового товару через діалогове вікно. Для цього вводилися назва килима, ширина, довжина, матеріал, колір, тип і ціна. Після підтвердження створення товар мав з’явитися у таблиці каталогу та зберегтися у файлі inventory.json.

Окремо перевірялося редагування наявного товару. Після вибору килима в таблиці відкривалося вікно редагування, у якому змінювалися окремі характеристики. Після збереження система повинна була оновити дані в таблиці та записати зміни у файл з товарами.

Також тестувалася логіка визначення формату товару. Якщо довжина килима становить 5 метрів або більше, товар повинен відображатися як рулон. Якщо залишок стає меншим за це значення, він повинен розглядатися як шматок. Це важливо для подальшого створення замовлень, оскільки для рулонів дозволяється введення довжини, а для шматків довжина є фіксованою.

Створення замовлення перевірялося через окреме діалогове вікно. Продавець обирає товар із таблиці, вводить необхідну довжину для рулонного килима або використовує фіксовану довжину для шматка. Після додавання позиції система повинна перевірити правильність введених даних і наявність достатньої довжини товару на складі.

Окремо перевірялося створення замовлення з кількома позиціями. У такому випадку система повинна правильно додавати кожну позицію до списку, окремо розраховувати її вартість і формувати загальну суму замовлення. Після завершення замовлення воно має з’явитися у таблиці замовлень зі статусом «Нове».

Також важливим тестовим випадком була перевірка захисту від створення порожнього замовлення. Якщо користувач відкрив вікно створення замовлення, але не додав жодної позиції або не натиснув кнопку завершення, таке замовлення не повинно зберігатися в системі.

Результати тестування роботи з товарами та створення замовлень наведено в таблиці 5.2.

Таблиця 5.2

Тестування роботи з товарами та створення замовлень

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат
1	Додавання нового килима	Назва, ширина, довжина, матеріал, колір, тип, ціна	Товар додається до таблиці каталогу	Виконано
2	Додавання товару з порожніми полями	Не заповнено назву або числові поля	Система виводить повідомлення про некоректні дані	Виконано
3	Додавання товару з некоректною ціною	У поле ціни введено текст	Товар не додається, з’являється повідомлення про помилку	Виконано
4	Редагування характеристик товару	Змінено назву, довжину або ціну	Дані в таблиці оновлюються після збереження	Виконано
5	Визначення товару як рулону	Довжина товару 500 см або більше	Товар відображається як рулон	Виконано
6	Визначення товару як шматка	Довжина товару менше 500 см	Товар відображається як шматок	Виконано
7	Фільтрація товарів	Обрано матеріал, колір, тип або формат	У таблиці залишаються тільки відповідні товари	Виконано
8	Сортування товарів	Натискання на заголовок стовпця	Дані впорядковуються за вибраним стовпцем	Виконано
9	Створення замовлення з однією позицією	Обрано рулонний килим, введено допустиму довжину	Замовлення створюється, позиція додається	Виконано
10	Створення замовлення з кількома позиціями	Обрано кілька різних килимів	Усі позиції додаються до одного замовлення	Виконано
11	Додавання шматка до замовлення	Обрано килим формату «шматок»	Довжина встановлюється автоматично і не редагується	Виконано
12	Введення довжини більшої за залишок	Введено довжину, що перевищує доступну	Позиція не додається, система показує повідомлення	Виконано

Продовження таблиці 5.2

13	Введення некоректної довжини	У поле довжини введено текст або порожнє значення	Позиція не додається, система повідомляє про помилку	Виконано
14	Автоматичний розрахунок суми	Додано позицію з довжиною і ціною за метр	Система правильно обчислює суму позиції та замовлення	Виконано
15	Спроба створення порожнього замовлення	Не додано жодної позиції	Система не створює порожнє замовлення	Виконано

За результатами тестування встановлено, що система коректно виконує основні операції з товарами: додавання, редагування, відображення, фільтрацію, сортування та визначення формату товару. Також підтверджено, що замовлення створюються правильно, система перевіряє доступну довжину, автоматично розраховує суму та не допускає появи порожніх замовлень.

5.3 Тестування залишків, статусів і збереження даних

Тестування залишків, статусів і збереження даних було спрямоване на перевірку коректності складського обліку після створення, скасування та відновлення замовлень. Для системи магазину килимів це є важливим етапом, оскільки після кожної операції кількість доступного товару повинна залишатися актуальною.

Під час створення замовлення система повинна зменшувати залишок відповідного килима на довжину, додану до замовлення. Якщо товар є рулоном, зменшується його доступна довжина. Якщо товар є шматком, він продається як готовий залишок фіксованого розміру. Окремо перевірялося, чи не дозволяє система списати більшу довжину, ніж є в наявності.

Також тестувалася робота зі статусами замовлень. Після створення замовлення воно повинно отримувати статус «Нове». Надалі продавець може змінювати статус на «Прийнято в роботу», «Готове до відправки», «Відправлено» або «Отримано/Закрито». У разі скасування замовлення статус має змінюватися

на «Повернуто/Скасовано».

Окрему увагу приділено скасуванню та відновленню замовлень. Після скасування товари з відповідного замовлення повинні повертатися на склад. При відновленні система повторно перевіряє наявність потрібної довжини товару. Якщо залишку достатньо, замовлення відновлюється, а товари знову списуються зі складу. Якщо залишку недостатньо, відновлення має бути заборонене.

Крім цього, перевірялося збереження та відновлення даних із JSON-файлів. Оскільки в розробленому застосунку дані зберігаються локально у файлах inventory.json та orders.json, важливо переконатися, що після закриття і повторного запуску програми інформація відновлюється коректно.

Під час тестування перевірялося збереження нових товарів, редагування характеристик килимів, створення замовлень, зміна статусів, скасування та відновлення замовлень. Після виконання кожної з цих дій програма закривалася та запускалася повторно. Очікуваним результатом було відновлення всіх змінених даних у тому самому стані, у якому вони були перед закриттям програми.

Окремо перевірялося, чи коректно зберігаються зміни складських залишків. Наприклад, після створення замовлення довжина відповідного рулону повинна зменшитися не лише в таблиці товарів, а й у файлі inventory.json. Після скасування замовлення залишок має збільшитися, а після відновлення — знову зменшитися.

Результати тестування залишків, статусів і збереження даних наведено в таблиці 5.3.

Таблиця 5.3

Тестування залишків, статусів і збереження даних

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат
1	Списання залишку після створення замовлення	Рулон 1000 см, замовлено 300 см	Залишок товару становить 700 см	Виконано
2	Заборона списання більшої довжини	Рулон 500 см, замовлено 800 см	Позиція не додається до замовлення	Виконано

Продовження таблиці 5.3

3	Створення замовлення зі шматком	Обрано товар формату «шматок»	Довжина встановлюється автоматично	Виконано
4	Зміна статусу замовлення	Статус змінено з «Нове» на «Прийнято в роботу»	Новий статус відображається в таблиці	Виконано
5	Послідовна зміна статусів	NEW → ACCEPTED → READY → SENT → RECEIVED	Статуси змінюються коректно	Виконано
6	Скасування замовлення	Активне замовлення з однією або кількома позиціями	Статус змінюється на «Повернуто/Скасовано», товар повертається на склад	Виконано
7	Відновлення скасованого замовлення	На складі достатньо товару	Замовлення відновлюється, товар повторно списується	Виконано
8	Відновлення при недостатньому залишку	На складі недостатньо товару для однієї з позицій	Система забороняє відновлення замовлення	Виконано
9	Повне видалення скасованого замовлення	Замовлення має статус «Повернуто/Скасовано»	Замовлення видаляється зі списку	Виконано
10	Спроба повного видалення активного замовлення	Замовлення має статус, відмінний від «Повернуто/Скасовано»	Повне видалення недоступне	Виконано
11	Збереження нового товару	Додано новий килим	Товар записується у inventory.json	Виконано
12	Відновлення товарів після перезапуску	Програму закрито і відкрито повторно	Список товарів завантажується з inventory.json	Виконано
13	Збереження змін після редагування товару	Змінено назву, довжину або ціну	Оновлені дані записуються у файл товарів	Виконано
14	Збереження нового замовлення	Створено замовлення з позиціями	Замовлення записується у orders.json	Виконано
15	Відновлення замовлень після перезапуску	Програму перезапущено	Список замовлень завантажується з orders.json	Виконано
16	Збереження зміни статусу	Статус замовлення змінено	Новий статус зберігається після перезапуску	Виконано
17	Збереження залишків після створення замовлення	Замовлено частину рулону	Довжина товару зменшується у inventory.json	Виконано

За результатами тестування встановлено, що система коректно змінює складські залишки після створення, скасування та відновлення замовлень. Також

підтверджено правильність роботи зі статусами замовлень, обмеженнями для скасованих та активних записів, а також збереженням інформації у JSON-файлах. Після повторного запуску програми дані відновлюються без втрати інформації.

5.4 Аналіз результатів тестування та можливості подальшого розвитку системи

За результатами проведеного тестування встановлено, що розроблене програмне забезпечення коректно виконує основні функції, визначені у вимогах. Було перевірено роботу з товарами, створення замовлень, контроль складських залишків, зміну статусів, скасування та відновлення замовлень, а також збереження даних у JSON-файлах.

Під час тестування підтверджено, що система дозволяє додавати й редагувати товари, правильно визначає формат килима як рулон або шматок, виконує фільтрацію та сортування даних у таблицях. Також встановлено, що після створення замовлення складські залишки оновлюються коректно, а система не дозволяє додати до замовлення більшу довжину товару, ніж є в наявності.

Особливу увагу було приділено перевірці механізмів валідації введених даних. Результати тестування показали, що система успішно виявляє помилки під час введення довжини, ціни або інших параметрів товару. У разі введення некоректних значень користувач отримує відповідне повідомлення, а операція не виконується до усунення помилки. Це дозволяє підтримувати цілісність даних та зменшує ризик виникнення некоректних записів у системі.

Окремо було перевірено механізм роботи зі статусами замовлень. Система коректно змінює статуси, відображає їх у таблиці, дозволяє скасувати замовлення з поверненням товару на склад і відновити його лише за умови наявності потрібної кількості товару. Це підтверджує правильність реалізації логіки обліку замовлень і залишків.

Також тестування показало, що дані про товари та замовлення зберігаються у файлах inventory.json і orders.json та відновлюються після

повторного запуску програми. Було виконано серію перевірок із багаторазовим додаванням товарів, створенням і редагуванням замовлень та подальшим перезапуском програми. У всіх випадках інформація успішно завантажувалася з файлів і відображалася в інтерфейсі без втрати даних. Це дає змогу використовувати застосунок у локальному режимі без підключення до бази даних або серверної частини.

Важливим результатом тестування стало підтвердження того, що програма запобігає основним помилкам користувача. Зокрема, система не дозволяє створювати порожні замовлення, додавати позиції з некоректною довжиною, продавати товар понад наявний залишок або виконувати критичні дії без підтвердження. Це підвищує надійність роботи застосунку та робить його придатним для щоденного використання продавцем.

Проведене тестування також показало, що графічний інтерфейс забезпечує зручну взаємодію користувача із системою. Табличне відображення товарів і замовлень дозволяє швидко переглядати основні дані, а фільтрація, сортування та кольорове виділення станів допомагають продавцю швидше знаходити потрібну інформацію. Завдяки цьому зменшується кількість зайвих дій під час роботи з каталогом і замовленнями.

Під час практичного використання програми було встановлено, що навіть при значній кількості товарів і замовлень інтерфейс залишається достатньо швидким та зручним. Оновлення таблиць після виконання операцій відбувається автоматично, а більшість дій користувача потребує мінімальної кількості кроків. Це позитивно впливає на продуктивність роботи продавця та скорочує час обслуговування клієнтів.

Аналіз результатів тестування свідчить про відповідність реалізованої системи поставленим функціональним вимогам. Усі основні бізнес-процеси магазину килимів були успішно реалізовані та перевірені на практиці. Отримані результати дозволяють зробити висновок про можливість використання програмного забезпечення як інструмента автоматизації діяльності невеликого торговельного підприємства.

Попри працездатність реалізованої системи, у майбутньому її можна розширити. Одним із напрямів розвитку є підключення бази даних, наприклад MySQL або PostgreSQL, що дозволить ефективніше працювати з більшими обсягами даних. Такий підхід буде доцільним у разі збільшення кількості товарів, замовлень або користувачів системи.

Також доцільним напрямом розвитку є додавання ролей користувачів, наприклад продавця, адміністратора або власника магазину. Це дозволить обмежити доступ до окремих функцій системи. Наприклад, продавець зможе створювати замовлення та переглядати товари, а адміністратор або власник — редагувати каталог, переглядати звіти та керувати налаштуваннями програми.

Ще одним напрямом розвитку є формування звітів про продажі, залишки товарів і скасовані замовлення. Такі звіти можуть допомогти власнику магазину аналізувати популярність товарів, контролювати наявність продукції та планувати закупівлі. Додатково систему можна доповнити можливістю експорту даних у формати Excel, CSV або PDF для подальшої обробки та друку.

Перспективним напрямом удосконалення є інтеграція системи зі сканерами штрихкодів або QR-кодів. Це дозволить прискорити пошук товарів у каталозі, зменшити кількість помилок під час введення інформації та зробити процес оформлення замовлень більш зручним.

Крім цього, у майбутньому можна реалізувати друк чеків, накладних або інших супровідних документів. Це зробить застосунок більш придатним для використання в реальних умовах магазину. Також корисним доповненням може бути резервне копіювання JSON-файлів, щоб запобігти втраті інформації у разі випадкового пошкодження або видалення даних.

Додатково система може бути розширена модулем статистики та аналітики. Такий модуль дозволить відстежувати динаміку продажів, визначати найбільш популярні категорії товарів, аналізувати сезонні зміни попиту та прогнозувати необхідність поповнення запасів. Наявність аналітичних інструментів підвищить ефективність управління магазином і сприятиме прийняттю обґрунтованих управлінських рішень.

У перспективі десктопний застосунок може бути розширений до клієнт-серверної або веб-системи. Це дозволить працювати з кількох робочих місць, синхронізувати дані між користувачами та забезпечити доступ до інформації з різних пристроїв. Такий варіант розвитку буде актуальним, якщо магазин матиме кілька продавців або декілька точок продажу.

Отже, результати тестування підтвердили працездатність розробленого програмного забезпечення, стабільність його роботи та відповідність основним функціональним вимогам. Система успішно виконує завдання автоматизації роботи продавця магазину килимів, а її архітектура забезпечує можливість подальшого розвитку, модернізації та розширення функціональних можливостей відповідно до майбутніх потреб користувачів.

5.5 Висновки по розділу

У п’ятому розділі було проведено тестування розробленого програмного забезпечення для магазину килимів. Перевірено основні сценарії роботи системи: додавання та редагування товарів, створення замовлень, перевірку складських залишків, зміну статусів, скасування й відновлення замовлень.

Результати тестування показали, що система коректно обробляє дані про товари та замовлення, не допускає створення некоректних або порожніх замовлень, правильно оновлює залишки після продажу, скасування чи відновлення замовлення. Також підтверджено працездатність механізму збереження та завантаження даних із JSON-файлів.

Отже, розроблений застосунок відповідає поставленим функціональним вимогам і може використовуватися для автоматизації основних процесів роботи продавця магазину килимів. Крім того, структура системи дозволяє надалі розширювати її функціональність, зокрема шляхом підключення бази даних, додавання звітів, ролей користувачів і можливості друку документів.

ВИСНОВКИ

У ході виконання бакалаврської роботи було розроблено десктопне програмне забезпечення для автоматизації роботи продавця магазину килимів із використанням Java-базованого стеку. Система забезпечує облік товарів, контроль складських залишків, створення та обробку замовлень, зміну статусів і збереження даних у JSON-файлах.

Було проаналізовано предметну область магазину килимів і визначено її основні особливості: продаж товарів за довжиною, поділ килимів на рулони та шматки, необхідність перевірки залишків і точного розрахунку вартості. На основі цього сформовано вимоги до програмного забезпечення та змодельовано основні сценарії роботи продавця.

У процесі проєктування визначено структуру системи, її основні модулі model, service, storage, ui і data, а також класи для опису товарів, замовлень, позицій замовлення, статусів і збереження даних. Практична реалізація виконана мовою Java з використанням Swing для побудови графічного інтерфейсу та Gson для роботи з JSON.

У програмі реалізовано каталог товарів, створення замовлень із кількома позиціями, перевірку доступної довжини, автоматичний розрахунок вартості, оновлення залишків, зміну статусів, скасування та відновлення замовлень. Проведене тестування підтвердило коректність роботи основних функцій системи та збереження даних після повторного запуску програми.

Отже, мету роботи було досягнуто: створено локальний десктопний Java-застосунок, який автоматизує основні процеси роботи продавця магазину килимів і зменшує ризик помилок під час обліку товарів та замовлень. Подальший розвиток системи може передбачати підключення бази даних, формування звітів, друк документів і розширення застосунку до веб-версії.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Піх В. Я., Піх М. М., В.В. Бандура. Методичні рекомендації та вимоги щодо виконання та оформлення бакалаврських робіт для студентів спеціальності 121 “Інженерія програмного забезпечення. В. Я. Піх, М. М. Піх, В.В. Бандура - Івано- Франківськ: ІФНТУНГ, 2024. - 52с.

2. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 2016-07-01. Вид. офіц. Київ : УкрНДНЦ, 2016. 16 с.

3. IBM. What is inventory management? [Електронний ресурс] URL: https://www.ibm.com/think/topics/inventory-management (дата звернення: 27.05.2026).

4. Oracle Fusion Cloud Inventory Management. [Електронний ресурс] URL: https://www.oracle.com/scm/inventory-management/ (дата звернення: 27.05.2026).

5. What Is Automated Inventory Management? [Електронний ресурс] URL: https://www.netsuite.com/portal/resource/articles/inventory-management/automated-inventory-management.shtml (дата звернення: 27.05.2026).

6. Understanding Stock Keeping Units (SKUs): How Retailers Use Them to Track Inventory. [Електронний ресурс] URL: https://www.investopedia.com/terms/s/stock-keeping-unit-sku.asp (дата звернення: 27.05.2026).

7. Package javax.swing. Java Platform SE Documentation. [Електронний ресурс] URL: https://docs.oracle.com/en/java/javase/25/docs/api/java.desktop/javax/swing/package-summary.html (дата звернення: 27.05.2026).

8. Package javax.swing.table. Java Platform SE Documentation. [Електронний ресурс] URL: https://docs.oracle.com/en/java/javase/25/docs/api/java.desktop/javax/swing/table/package-summary.html (дата звернення: 27.05.2026).

9. Inventory Management Software. Square. [Електронний ресурс] URL: https://squareup.com/us/en/point-of-sale/features/inventory-management (дата звернення: 27.05.2026).

10. Cloud-Based iPad & Android Point-of-Sale (POS) System. Poster POS. [Електронний ресурс] URL: https://joinposter.com/en (дата звернення: 27.05.2026).

11. Restaurant Inventory Management Software. Poster POS. [Електронний ресурс] URL: https://joinposter.com/en/tour/inventory (дата звернення: 27.05.2026).

12. Inventory. Odoo 19.0 documentation. [Електронний ресурс] URL: https://www.odoo.com/documentation/19.0/applications/inventory_and_mrp/inventory.html (дата звернення: 27.05.2026).

13. BAS Управління торгівлею. [Електронний ресурс] URL: https://www.bas-soft.eu/soft/bas-mass/bas-trade-management/ (дата звернення: 27.05.2026).

14. Online Inventory Management Software. Zoho Inventory. [Електронний ресурс] URL: https://www.zoho.com/us/inventory/ (дата звернення: 27.05.2026).

15. Free customizable inventory list templates. Microsoft Excel. [Електронний ресурс] URL: https://excel.cloud.microsoft/create/en/inventory-templates/ (дата звернення: 27.05.2026).

16. What are Workflows? Atlassian. [Електронний ресурс] URL: https://www.atlassian.com/work-management/workflows (дата звернення: 27.05.2026).

17. Lesson: Object-Oriented Programming Concepts. Oracle. [Електронний ресурс] URL: https://docs.oracle.com/javase/tutorial/java/concepts/index.html (дата звернення: 27.05.2026).

18. Collection. Java Platform SE Documentation. Oracle. [Електронний ресурс] URL: https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/util/Collection.html (дата звернення: 27.05.2026).

19. Shopify. Inventory management [Електронний ресурс]. – URL: https://help.shopify.com/en/manual/sell-in-person/shopify-pos/inventory-management (дата звернення: 27.05.2026).

20. Oracle. The Java Tutorials: Classes and Objects [Електронний ресурс]. – URL: https://docs.oracle.com/javase/tutorial/java/javaOO/index.html (дата звернення: 27.05.2026).

21. Oracle. JFrame — Java Platform SE API Specification [Електронний ресурс]. – URL: https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html (дата звернення: 27.05.2026).

22. Oracle. JTable — Java Platform SE API Specification [Електронний ресурс]. – URL: https://docs.oracle.com/javase/8/docs/api/javax/swing/JTable.html (дата звернення: 27.05.2026).

23. Oracle. JOptionPane — Java Platform SE API Specification [Електронний ресурс]. – URL: https://docs.oracle.com/javase/8/docs/api/javax/swing/JOptionPane.html (дата звернення: 27.05.2026).

24. Oracle. How to Use Tables. The Java Tutorials [Електронний ресурс]. – URL: https://docs.oracle.com/javase/tutorial/uiswing/components/table.html (дата звернення: 27.05.2026).

25. IETF. RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format [Електронний ресурс]. – URL: https://datatracker.ietf.org/doc/html/rfc8259 (дата звернення: 27.05.2026).

26. ECMA International. ECMA-404: The JSON Data Interchange Syntax [Електронний ресурс]. – URL: https://ecma-international.org/publications-and-standards/standards/ecma-404/ (дата звернення: 27.05.2026).

27. Google. Gson User Guide [Електронний ресурс]. – URL: https://google.github.io/gson/UserGuide.html (дата звернення: 27.05.2026).

28. Oracle. How to Use Split Panes. The Java Tutorials [Електронний ресурс]. – URL: https://docs.oracle.com/javase/tutorial/uiswing/components/splitpane.html (дата звернення: 27.05.2026).

29. Oracle. How to Use Combo Boxes. The Java Tutorials [Електронний ресурс]. – URL: https://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html (дата звернення: 27.05.2026).

30. Oracle. How to Use Dialogs. The Java Tutorials [Електронний ресурс]. – URL: https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html (дата звернення: 27.05.2026).

31. Oracle. How to Write Event Listeners. The Java Tutorials [Електронний ресурс]. – URL: https://docs.oracle.com/javase/tutorial/uiswing/events/index.html (дата звернення: 27.05.2026).

32. Oracle. How to Use Lists. The Java Tutorials [Електронний ресурс]. – URL: https://docs.oracle.com/javase/tutorial/uiswing/components/list.html (дата звернення: 27.05.2026).

33. Oracle. How to Use Buttons, Check Boxes, and Radio Buttons. The Java Tutorials [Електронний ресурс]. – URL: https://docs.oracle.com/javase/tutorial/uiswing/components/button.html (дата звернення: 27.05.2026).

34. Oracle. How to Use Layout Managers. The Java Tutorials [Електронний ресурс]. – URL: https://docs.oracle.com/javase/tutorial/uiswing/layout/index.html (дата звернення: 27.05.2026).

35. Oracle. BorderLayout. Java Platform SE API Specification [Електронний ресурс]. – URL: https://docs.oracle.com/javase/8/docs/api/java/awt/BorderLayout.html (дата звернення: 27.05.2026).

ДОДАТКИ

ДОДАТОК А

Посилання на репозиторій програмного проєкту

Програмний код розробленого застосунку розміщено у GitHub-репозиторії:

https://github.com/Viromin/IP-22-4_Pelekhovych_A_B

ДОДАТОК Б

Фрагменти програмного коду основних класів системи

Клас Carpet

```
public class Carpet {
    private int id;
    private String name;
    private CarpetDetails details;
    private CarpetAttributes attributes;
    private double price;

    public Carpet(int id, String name, CarpetDetails details, CarpetAttributes
attributes,
                  double price) {
        this.id = id;
        this.name = name;
        this.details = details;
        this.attributes = attributes;
        this.price = price;
    }

    public void setId(int id) {
        this.id = id;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setPrice(double price) {
        this.price = price;
    }

    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public CarpetDetails getDetails() {
        return details;
    }
    public CarpetAttributes getAttributes() {
        return attributes;
    }
    public double getPrice() {
        return price;
    }

}
```

Клас CarpetDetails

```
public class CarpetDetails {
    private int widthCm;
    private int lengthCm;
    private boolean isRoll;

    public CarpetDetails(
            int widthCm,
            int lengthCm
    ) {

        this.widthCm = widthCm;

        this.lengthCm = lengthCm;

        this.isRoll = lengthCm >= 500;
    }

    public void setWidthCm(int widthCm) {
        this.widthCm = widthCm;
    }

    public void setLengthCm(int lengthCm) {
        this.lengthCm = lengthCm;
        this.isRoll = lengthCm >= 500;
    }

    public int getWidthCm() {
        return widthCm;
    }
    public int getLengthCm() {
        return lengthCm;
    }
    public boolean isRoll() {
        return isRoll;
    }

}
```

Клас CarpetAttributes

```
public class CarpetAttributes {
    private Material material;
    private CarpetColor carpetColor;
    private CarpetType type;

    public CarpetAttributes(Material material, CarpetColor carpetColor, CarpetType
type) {
        this.material = material;
        this.carpetColor = carpetColor;
        this.type = type;
    }

    public void setMaterial(Material material) {
        this.material = material;
    }
    public void setCarpetColor(CarpetColor carpetColor) {
        this.carpetColor = carpetColor;
    }
```

```
    public void setType(CarpetType type) {
        this.type = type;
    }

    public Material getMaterial() {
        return material;
    }
    public CarpetColor getCarpetColor() {
        return carpetColor;
    }
    public CarpetType getType() {
        return type;
    }
}
```

Приклад enum-класу

```
public enum CarpetType {
    PERSIAN("Перський"),
    TURKISH("Турецький"),
    MODERN("Сучасний"),
    CLASSIC("Класичний"),
    SHAGGY("Високоворсний");

    private final String uaName;

    CarpetType(String uaName) {this.uaName = uaName;}

    public String getUaName() {return uaName;}

    @Override
    public String toString() {return uaName;}
}
```

Клас Inventory

```
public class Inventory {
    private List<Carpet> carpetsList = new ArrayList<>();

    public void addCarpet(Carpet carpet) {
        carpetsList.add(carpet);
    }

    public List<Carpet> getCarpetsList() {
        return carpetsList;
    }

    public void setCarpetsList(List<Carpet> carpetsList) {
        this.carpetsList = carpetsList;
    }
}
```

Клас Order

```
public class Order {
    private int id;
    private List<OrderItem> items = new ArrayList<>();
    private double totalPrice;
    private OrderStatus status;

    public Order(int id) {
        this.id = id;
        this.status = OrderStatus.NEW;
    }

    public void addItem(OrderItem item) {
        items.add(item);
        totalPrice += item.getTotalPrice();
    }

    public int getId() { return id; }
    public List<OrderItem> getItems() { return items; }
    public double getTotalPrice() { return totalPrice; }
    public OrderStatus getStatus() { return status; }

    public void setStatus(OrderStatus status) {
        this.status = status;
    }

    public void recalculateTotalPrice() {

        totalPrice = 0;

        for (OrderItem item : items) {

            totalPrice += item.getTotalPrice();
        }
    }
}
```

Клас OrderItem

```
public class OrderItem {

    private Carpet carpet;
    private int lengthCm;

    public OrderItem(Carpet carpet, int lengthCm) {
        this.carpet = carpet;
        this.lengthCm = lengthCm;
    }

    public void setLengthCm(int lengthCm) {
        this.lengthCm = lengthCm;
    }

    public Carpet getCarpet() {
        return carpet;

    }

    public int getLengthCm() {
```

```
        return lengthCm;
    }

    public double getTotalPrice() {

        return (lengthCm / 100.0)
                * carpet.getPrice();
    }
}
```

Клас OrderStatus

```
public enum OrderStatus {
    NEW("Нове"),
    ACCEPTED("Прийнято в роботу"),
    READY("Готове до відправки"),
    SENT("Відправлено"),
    RECEIVED("Отримано/Закрито"),
    RETURNED("Повернуто/Скасовано");

    private final String uaName;

    OrderStatus(String uaName) {
        this.uaName = uaName;
    }

    @Override
    public String toString() {
        return uaName;
    }
}
```

Клас InventoryService

```
public class InventoryService {

    private Inventory inventory = new Inventory();

    public void load() {
        List<Carpet> carpets = InventoryStorage.load();
        inventory.setCarpetsList(carpets);
    }

    public void save() {
        InventoryStorage.save(inventory.getCarpetsList());
    }

    private int generateNextId() {
        return inventory.getCarpetsList().stream()
                .mapToInt(Carpet::getId)
                .max()
                .orElse(0) + 1;
    }

    public Carpet findById(int id) {
        return inventory.getCarpetsList()
                .stream()
                .filter(carpet -> carpet.getId() == id)
                .findFirst()
```

```
                .orElse(null);
    }



    public void addCarpet(Carpet carpet) {
        carpet.setId(generateNextId());
        inventory.addCarpet(carpet);
    }


    public List<Carpet> getAll() {
        return inventory.getCarpetsList();
    }
}
```

Клас OrderService

```
public class OrderService {

    private List<Order> orders = new ArrayList<>();

    public void load() {
        orders = OrderStorage.load();
    }

    public void save() {
        OrderStorage.save(orders);
    }

    private int generateNextId() {
        return orders.stream()
                .mapToInt(Order::getId)
                .max()
                .orElse(0) + 1;
    }

    public Order createOrder() {
        Order order = new Order(generateNextId());
        orders.add(order);
        return order;
    }

    public Order findById(int orderId) {
        return orders.stream()
                .filter(order -> order.getId() == orderId)
                .findFirst()
                .orElse(null);
    }

    public boolean addItemToOrder(Order order, Carpet carpet, int lengthCm) {

        if (order == null || carpet == null) {
            return false;
        }

        if (lengthCm <= 0) {
            return false;
        }

        int availableLength = carpet.getDetails().getLengthCm();
```

```
    if (!carpet.getDetails().isRoll()) {
        lengthCm = availableLength;
    }

    if (availableLength < lengthCm) {
        return false;
    }

    OrderItem item = new OrderItem(carpet, lengthCm);
    order.addItem(item);

    int newLength = availableLength - lengthCm;
    carpet.getDetails().setLengthCm(newLength);

    return true;
}

public boolean cancelOrder(int orderId, InventoryService inventoryService) {
    Order order = findById(orderId);

    if (order == null) {
        return false;
    }

    if (order.getStatus() == OrderStatus.RETURNED) {
        return false;
    }

    for (OrderItem item : order.getItems()) {
        Carpet carpetInInventory = inventoryService.findById(
                item.getCarpet().getId()
        );

        if (carpetInInventory != null) {
            int newLength = carpetInInventory.getDetails().getLengthCm()
                    + item.getLengthCm();

            carpetInInventory.getDetails().setLengthCm(newLength);
        }
    }

    order.setStatus(OrderStatus.RETURNED);
    return true;
}

public boolean restoreOrder(int orderId, InventoryService inventoryService) {
    Order order = findById(orderId);

    if (order == null) {
        return false;
    }

    if (order.getStatus() != OrderStatus.RETURNED) {
        return false;
    }

    for (OrderItem item : order.getItems()) {
        Carpet carpetInInventory = inventoryService.findById(
                item.getCarpet().getId()
        );
```

```
            if (carpetInInventory == null) {
                return false;
            }


            if (carpetInInventory.getDetails().getLengthCm() < item.getLengthCm()) {
                return false;
            }
        }

        for (OrderItem item : order.getItems()) {
            Carpet carpetInInventory = inventoryService.findById(
                    item.getCarpet().getId()
            );

            int newLength = carpetInInventory.getDetails().getLengthCm()
                    - item.getLengthCm();

            carpetInInventory.getDetails().setLengthCm(newLength);
        }

        order.setStatus(OrderStatus.NEW);
        return true;
    }

    public boolean permanentlyDeleteOrder(int orderId) {
        Order order = findById(orderId);

        if (order == null) {
            return false;
        }

        if (order.getStatus() != OrderStatus.RETURNED) {
            return false;
        }

        return orders.remove(order);
    }

    public List<Order> getAll() {
        return orders;
    }

    public Order createOrderDraft() {
        return new Order(generateNextId());
    }

    public boolean deleteOrderIfEmpty(int orderId) {
        Order order = findById(orderId);

        if (order == null) {
            return false;
        }

        if (!order.getItems().isEmpty()) {
            return false;
        }

        return orders.remove(order);
    }
```

```
    public void addOrder(Order order) {
        if (order != null && !order.getItems().isEmpty()) {
            orders.add(order);
        }


    }

    public void changeStatus(int orderId, OrderStatus newStatus) {
        for (Order order : orders) {
            if (order.getId() == orderId) {
                order.setStatus(newStatus);
                return;
            }
        }
    }
}
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: «Розробка програмного забезпечення для магазину коврів на Java базованих стеках»

Обсяг пояснювальної записки: 75 аркушів

Дата закінчення роботи _____.

Підпис _____