Міністерство освіти та науки України

Національний університет «Одеська політехніка»

Інститут комп’ютерних систем

Кафедра інженерії програмного забезпечення

Курсова робота

з дисципліни:

«Об’єктно-орієнтоване програмування»

Виконав:

студент

групи АС-221

Воробйов В. П.

Перевірила:

ст. викладач

Шаманіна Т.В.

Одеса – 2023

АНОТАЦІЯ

Ця курсова робота присвячена проектуванню та розробці системи автоматизації пункту друкарні з використанням сучасних технологій та програмного забезпечення. Метою цього проекту є оптимізація процесів, пов'язаних із збором інформації, замовленнями, та поліпшення продуктивності друкарні. У роботі детально розглядаються наступні аспекти: аналіз поточного стану замовлення автоматизації, проектування системи автоматизації, розробка програмного забезпечення, тестування, валідація проекту та висновки.

До аналізу поточного стану входить: оцінка існуючих процесів, обладнання та програмного забезпечення, ідентифікуються основні проблеми та обмеження. До проектування системи автоматизації: визначаються вимоги до системи, розроблюється архітектура та концепція роботи. До розробки програмного забезпечення: створення програмна частина системи, яка включає в себе представлення потрібної інформації. До тестування та валідація: перевірка роботи системи в реальних умовах та визначення її можливості та обмеження. До висновку відноситься узагальнення результатів проекту.

# ЗМІСТ

[АНОТАЦІЯ 0](#_Toc152749609)

[1 ЗАВДАННЯ ДО КУРСОВОЇ 2](#_Toc152749611)

[2 ФОРМУЛЮВАННЯ ВИМОГ 2](#_Toc152749612)

[2.1 Прецеденти: 2](#_Toc152749613)

[2.2 Опис прецедентів 3](#_Toc152749614)

[3 СТРУКТУРИ КЛАСІВ 4](#_Toc152749615)

[3.1 Опис прецедентів з назвою можливих класів для реалізації кожного пункту сценарію 4](#_Toc152749616)

[3.2 Загальна діаграма концептуальних класів 5](#_Toc152749617)

[3.3 Діаграма системних операцій для «важливого» прецедентна 7](#_Toc152749618)

[3.4 Діаграми взаємодії для кожного пункту сценарію «важливого» прецеденту 7](#_Toc152749619)

[3.4.1 Проектне рішення viewSpecializations 8](#_Toc152749620)

[3.4.2 Проектне рішення viewListStaff 8](#_Toc152749621)

[3.4.3 Проектне рішення viewListClients 8](#_Toc152749622)

[3.4.4 Проектне рішення viewCountClientsWithTime 9](#_Toc152749623)

[3.4.5 Проектне рішення viewMaxCostOrder 9](#_Toc152749624)

[3.4.6 Проектне рішення viewMiddleCost 9](#_Toc152749625)

[4 КОДИ КЛАСІВ 11](#_Toc152749626)

[5 СТРУКТУРИ ФАЙЛІВ 12](#_Toc152749627)

[6 ІНСТРУКЦІЯ КОРИСТУВАЧА 13](#_Toc152749628)

[6.1 Загальні умови перед використанням 13](#_Toc152749629)

[6.2 Інструкція використання програми 13](#_Toc152749630)

[7 РЕЗУЛЬТАТ ТЕСТУВАННЯ 20](#_Toc152749631)

[8 ВИСНОВКИ 22](#_Toc152749632)

[9 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 23](#_Toc152749633)

[ДОДАТОК А – КОД ПРОГРАМИ 24](#_Toc152749634)

1 ЗАВДАННЯ ДО КУРСОВОЇ

**1.1 Загальне положення**

Метою курсової роботи є поглиблення та закріплення знань, одержаних при вивченні дисципліни «Об’єктно-орієнтоване програмування», та набуття практичних навичок у проектуванні та налагодженні програм, що застосовують класи та об’єкти.

Я маю розробити консольне застосування для роботи зі списками об'єктів. Інформація для списків може зберігатися в файлах. Кожен зі списків повинен містити не менше 3 об'єктів класу.

Структура сутностей приведена в описі кожного індивідуального завдання. Студент може додавати в клас якості, але їх не може бути менше, ніж зазначено в завданні.

Застосування повинне обробляти не менше шести запитів.

Для реалізації успадкування в роботі повинен використовуватися абстрактний клас і породжені від нього класи (не менше двох). Для реалізації композиції повинні бути створені класи, які не входять в ієрархію і являють собою деякі властивості (як мінімум один клас). Бажано деякі властивості реалізувати статичними даними.

В роботі треба навести опис і реалізацію інтерфейсу користувача.

Розроблена програма має забезпечити виконання, як мінімум 6 запитів наведених у кожному індивідуальному завданні.

Таблиця 1.1 - Завдання №6 Пункт ксерокопії

|  |  |  |  |
| --- | --- | --- | --- |
| Властивості | Класи | | |
| Майстер | Клієнт | Послуга |
| Назва пункту | + | + |  |

Продовження таблиці 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| ПІБ | + | + |  |
| Адреса | + |  |  |
| № телефону | + | + |  |
| Спеціалізація | + |  |  |
| Id послуги |  | + | + |
| Тип |  |  | + |
| Назва |  |  | + |
| Вартість |  |  | + |
| Призначений  час |  |  | + |

Запити до системи.

1) Список майстрів.

2) Список спеціалізацій майстрів.

3) Список клієнтів.

4) Кількість клієнтів, яким призначена послуга на вказаний період.

5) Найдорожча послуга.

6) Середня вартість послуги.

2 ФОРМУЛЮВАННЯ ВИМОГ

Назва проекту: Автоматизація демонстрації інформації для пункту ксерокопії.

Пункти ксерокопії компанії «ООО рога й папір « працюють на багатьох вулицях Миколаєва. Вони вже мають програму для автоматичного внесення даних у файлову базу даних, але не мають клієнта, якій може видавати тільки потрібну інформацію з цих даних. Продавець послуг здійснює прийом замовлень та видачу. Оператор друкарні виконує замовлення. Затрати часу на збір даних і їх розрахунок знижує ефективність компанії та її масштабування.

Потрібно розробити автоматизацію, що дозволяє прискорити збір і обробку даних, а також прискорити повідомлення о створені нового замовлення і його розрахунок.

2.1 Прецеденти:На рис. 2.1 зображена діаграма прецедентів.



Рисунок 2.1 - UML діаграма прецедентів.

На рисунку 2.1 зображено взаємодія користувачів з програмою і взаємодія програми з користувачами.

2.2 Опис прецедентів**:**

а) Представлення існуючих працівників та часу

Основна діюча особа: продавець.

Учасники та інтереси Продавець – отримати інформацію щодо доступних професій, працівників та часу для замовлення.

Клієнт – замовити послугу.

Передумова: Клієнт замолює послугу.

Мінімальна гарантія: Касир звертається за інформацією у системі.

Гарантія успіху: Касир отримує відповідну інформацію.

Тригер: Касир звертається за інформацією.

Основний сценарій:

1. Клієнт замовляє послугу, касир створює замовлення. Касир дивиться у системі, які зараз існують доступні спеціальності, працівників та клієнтів, для створення замовлення.

2. Касир запитує наявність інших замовлень на заданий час для кінцевого створення замовлення. Система повертає замовлення на даний період.

б) Отримання існуючих замовлень та інформації для визначення пріоритету.

Основна дієва особа: Оператор.

Учасники та інтереси: Оператор – отримати існуючі замовлення та інформації для визначення пріоритет.

Передумова: Працюючий у даний момент оператор, існуюче замовлення.

Мінімальна гарантія: .

Гарантія успіху: Клієнт отримує замовлену послугу.

Тригер: Перевірка оператором існуючі замовлення.

Основний сценарій:

1. Оператор переглядає саме найдорожче замовлення та перевіряє наявність інших замовлень на даний термін у системі.

2. Оператор виявляє більш пріоритетне замовлення. Оператор дивиться у системі середню ціну замовлення та всі замовлення у системі.

3 СТРУКТУРИ КЛАСІВ

3.1 Опис прецедентів з назвою можливих класів для реалізації кожного пункту сценарію**:**

Через аналіз двох най важливих варіантів використання «Представлення існуючих працівників та часу» і «Отримання існуючих замовлень та інформації для визначення пріоритету» ми отримаємо відповідні класи до кожного пункту їх сценарію з таблиці 3.1.1 та 3.1.2.

Таблиця 3.1.1  - Виявлення концептуальних класів на підставі ВВ «Представлення існуючих працівників та часу»

|  |  |  |
| --- | --- | --- |
| № пункту | Зміст пункту основного сценарію | Можливий клас |
| 1 | Клієнт замовляє послугу, касир створює замовлення. Касир дивиться у системі, які зараз існують доступні спеціальності, працівників та клієнтів, для створення замовлення. | Staff, StaffService, CrudRepository, ObjectToFile, Client, ClientService, DefaultService |
| 2 | Касир запитує наявність інших замовлень на заданий час для кінцевого створення замовлення. Система повертає замовлення на даний період. | Order, OrderService, ProductType |

Таблиця 3.1.2  - Виявлення концептуальних класів на підставі ВВ «Отримання існуючих замовлень та інформації для визначення пріоритету»

|  |  |  |
| --- | --- | --- |
| № пункту | Зміст пункту основного сценарію | Можливий клас |
| 1 | Оператор переглядає саме найдорожче замовлення та перевіряє наявність інших замовлень на даний термін у системі. | Order, OrderService, ProductType, Client, ClientService, CrudRepository, ObjectToFile, DefaultService |
| 2 | Оператор виявляє більш пріоритетне замовлення. Оператор дивиться у системі середню ціну замовлення та всі замовлення у системі. | Order, OrderService |

З даних таблиць можливо створити діаграми концептуальних класів.

## 3.2 Загальна діаграма концептуальних класів

Через аналіз відношень класів у змісту пунктів першого та другого прецеденту, з таблиці 3.1.1 і 3.1.2 відповідно, була сформульована наступна діаграма на рисунку 3.2.1.

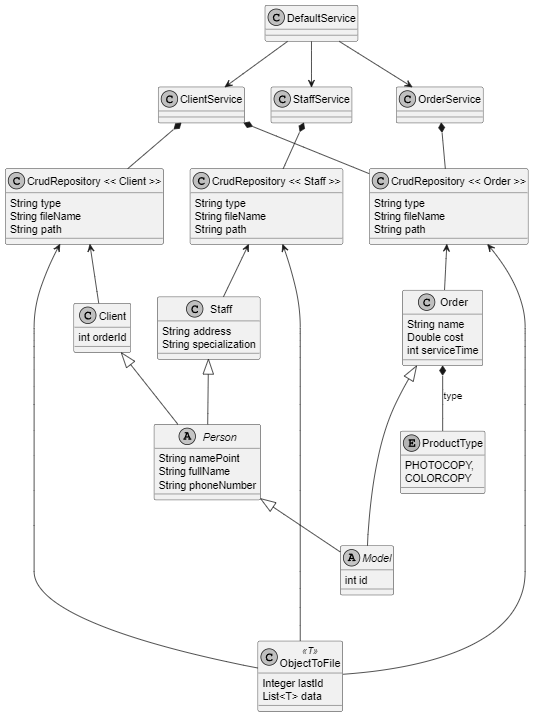


Рисунок 3.2.1 - Діаграма концептуальних класів прецеденту «Представлення існуючих працівників та часу» і «Отримання існуючих замовлень та інформації для визначення пріоритету», тобто загальна

З рисунку 3.2.1 можливо знайти системні операції.

## 3.3 Діаграма системних операцій для «важливого» прецедентна

Через аналіз пунктів прецеденту Представлення існуючих працівників та часу» і «Отримання існуючих замовлень та інформації для визначення пріоритету» було отримано наступні системні операції з рисунку 3.3.1.

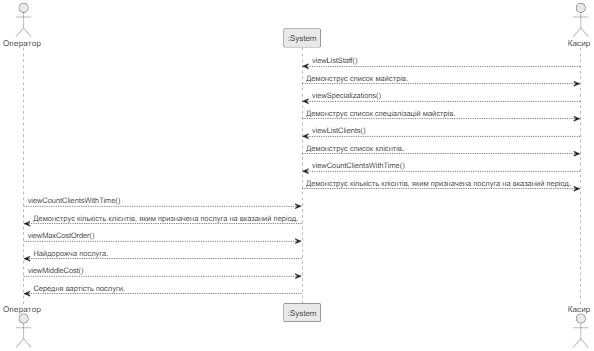


Рисунок 3.3.1  - Системні операції на підставі ВВ «Замовлення послуги»

З рисунку 3.3.1 виходить діаграм взаємодії.

## 3.4 Діаграми взаємодії для кожного пункту сценарію «важливого» прецеденту

Через аналіз відношень класів та самих змісту пунктів першого та другого прецеденту, були сформульовані наступні діаграма на рисунку 3.4.1, 3.4.2, 3.4.3, 3.4.4, 3.4.5 та 3.4.6.

### 3.4.1 Проектне рішення viewSpecializations

Клієнт замовляє послугу, касир створює замовлення. Касир дивиться у системі, які зараз існують доступні спеціальності



Рисунок 3.4.1  - Проектне рішення viewSpecializations

На рисунку 3.4.1 зображено діаграма проектного рішення.

### 3.4.2 Проектне рішення viewListStaff

Касир дивиться у системі, які зараз існують доступні працівників.



Рисунок 3.4.2  - Проектне рішення viewListStaff

На рисунку 3.4.2 зображено діаграма проектного рішення.

### 3.4.3 Проектне рішення viewListClients

Касир дивиться у системі, які зараз існують клієнти.



Рисунок 3.4.3  - Проектне рішення viewListClients

На рисунку 3.4.3 зображено діаграма проектного рішення.

### 3.4.4 Проектне рішення viewCountClientsWithTime

Касир запитує наявність інших замовлень на заданий час для кінцевого створення замовлення. Система повертає замовлення на даний період.



Рисунок 3.4.4  - Проектне рішення viewCountClientsWithTime

На рисунку 3.4.4 зображено діаграма проектного рішення.

### 3.4.5 Проектне рішення viewMaxCostOrder

Оператор переглядає саме найдорожче замовлення.



Рисунок 3.4.5  - Проектне рішення viewMaxCostOrder

На рисунку 3.4.5 зображено діаграма проектного рішення.

### 3.4.6 Проектне рішення viewMiddleCost

Оператор виявляє більш пріоритетне замовлення. Оператор дивиться у системі середню ціну замовлення.



Рисунок 3.4.6  - Проектне рішення viewMiddleCost

З наданої інформації можливо почати написання коду програми.

4 КОДИ КЛАСІВ

Коди класів, як і весь код програми знаходиться у додатку А. Відео з зображенням праці програми у додатку Б.

5 СТРУКТУРИ ФАЙЛІВ

Структура файлів має наступний вигляд на рисунку 5.

src:

Main.java

Config:

Config.java

Models:

Order:

Order.java

ProductType.java

Client.java

Model.java

Person.java

Staff.java

Repository:

CrudRepository.java

ObjectToFile.java

Services:

ClientService.java

DefaultService.java

OrderService.java

StaffService.java

Рисунок 5. Структура файлів проекту.

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

## 6.1 Загальні умови перед використанням

Для використання програмного забезпечення потрібно мати встановлену JRE (Java Runtime Environment), файл типу «jar» «Photocopy point.jar» та папку з назвою "data" з файлами розширенні «data», такими як: «ClientObject.data», «OrderObject.data», «StaffObject.data» – вони повинні мати дані з відповідною структурою щодо об’єктів класів з програми.

Програма запускається простим запуском файлу «Photocopy point.jar» або через команду у консолі «java -jar './Photocopy point.jar'«.

## 6.2 Інструкція використання програми

На рисунку 6.2.1 зображене головне меню, на якому зображено всі можливі варіанти отримання даних. При натисканні одної з неї буде виведена інша сторінка на програмному вікні відповідно, щодо натиканої кнопці. Якщо буде невистачити місця у вікні для інформації, то з’явиться можливість прокрутки.

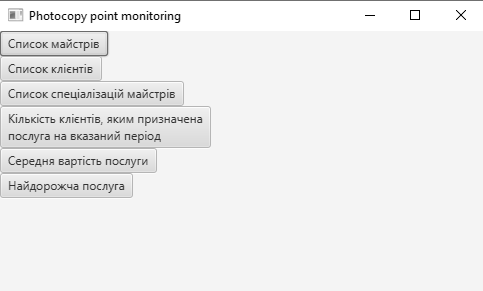


Рисунок 6.2.1 Головне меню програми.

На рисунку 6.2.2 зображене сторінка при відсутності даних у папці «data» даних чи самої папці. При цьому воно відображається при нажаті будь якої кнопки у головному меню. Кнопка «Повернутись» при нажаті повертає користувача до головного меню.

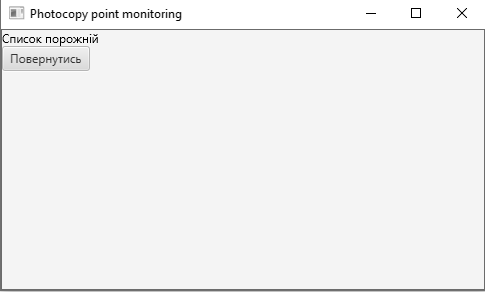


Рисунок 6.2.2 Повідомлення о неможливості отримати дані

Якщо пака «data», відповідні файли з цим розширенням та коректні дані у цих файлів існують. На рисунку 6.2.3 зображено те, що ми отримуємо при нажаті кнопці «Список майстрів». А саме отримуємо лист, який складається з усіх працюючих майстрів у компанії з їх ПІБ, посадою, назвою пункту де він працює, її адреса та телефонний номер. Кнопка «Повернутись» при нажаті повертає користувача до головного меню. Якщо буде невистачити місця у вікні для інформації, то з’явиться можливість прокрутки.

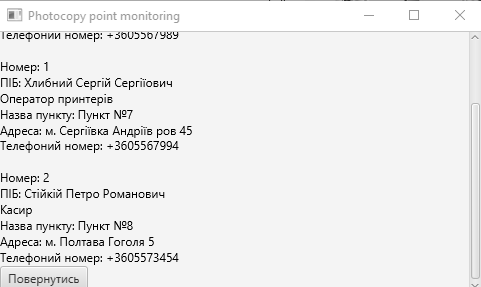


Рисунок 6.2.3 Зображення програмою список майстрів

Якщо пака «data», відповідні файли з цим розширенням та коректні дані у цих файлів існують. На рисунку 6.2.4 зображено те, що ми отримуємо при нажаті кнопці «Список клієнтів». А саме отримуємо лист, який складається з усіх клієнтів, які чекають отримання замовлення, з їх ПІБ, назвою пункту де він зумовлює, телефонний номер та номер замовлення. Кнопка «Повернутись» при нажаті повертає користувача до головного меню. Якщо буде невистачити місця у вікні для інформації, то з’явиться можливість прокрутки.

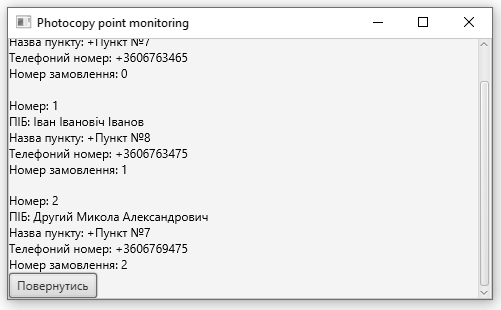


Рисунок 6.2.4 Зображення програмою список клієнтів

Якщо пака «data», відповідні файли з цим розширенням та коректні дані у цих файлів існують. На рисунку 6.2.5 зображено те, що ми отримуємо при нажаті кнопці «Список спеціалізацій майстрів». А саме отримуємо лист усіх можливих спеціалізація, які можуть мати співробітники у пункті для творення замовлення. Кнопка «Повернутись» при нажаті повертає користувача до головного меню. Якщо буде невистачити місця у вікні для інформації, то з’явиться можливість прокрутки.

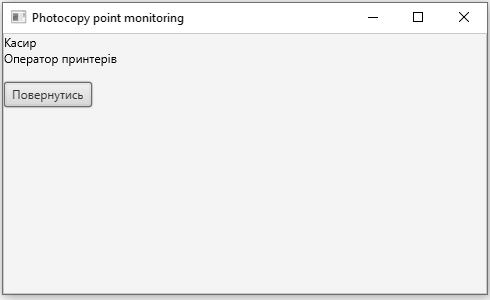


Рисунок 6.2.5 Зображення програмою список можливих спеціалізацій для майстрів

Якщо пака «data», відповідні файли з цим розширенням та коректні дані у цих файлів існують. На рисунку 6.2.6 зображено те, що ми отримуємо при нажаті кнопці «Кількість клієнтів, яким призначена послуга на вказаний період», вводі 1 у поле та нажаті кнопці «Знайти». А саме кількість клієнтів за позначеним періодом. Період водиться користувачем тільки у ціло-чисельному форматі. При неправильному вводі видається помилка. Кнопка «Повернутись» при нажаті повертає користувача до головного меню. Якщо буде невистачити місця у вікні для інформації, то з’явиться можливість прокрутки.

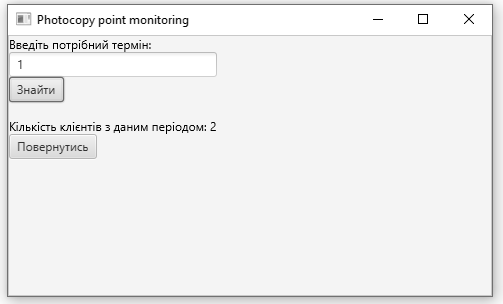


Рисунок 6.2.6 Зображення програмою кількість клієнтів та заданий період

Якщо пака «data», відповідні файли з цим розширенням та коректні дані у цих файлів існують. На рисунку 6.2.7 зображено те, що ми отримуємо при нажаті кнопці «Середня вартість послуг». А саме середня вартість послуг. Кнопка «Повернутись» при нажаті повертає користувача до головного меню. Якщо буде невистачити місця у вікні для інформації, то з’явиться можливість прокрутки.

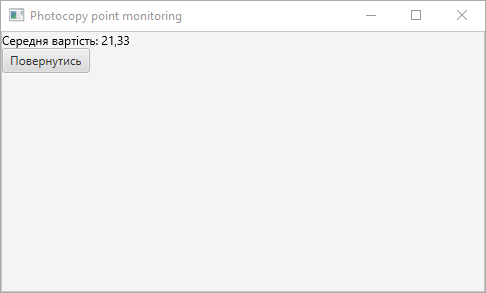


Рисунок 6.2.7 Зображення програмою середня вартість послуг

Якщо пака «data», відповідні файли з цим розширенням та коректні дані у цих файлів існують. На рисунку 6.2.8 зображено те, що ми отримуємо при нажаті кнопці «Найдорожча послуга». А саме інформацію щодо найдорожчій послуги, у порівняння з іншими. Кнопка «Повернутись» при нажаті повертає користувача до головного меню. Якщо буде невистачити місця у вікні для інформації, то з’явиться можливість прокрутки.

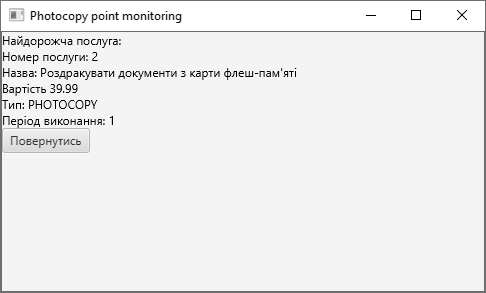


Рисунок 6.2.8 Зображення програмою інформацію щодо найдорожчої послуги

7 РЕЗУЛЬТАТ ТЕСТУВАННЯ

Наступні тест-кейси, на таблиці 7, перевіряють відповідність підсумкової програми щодо вимог.

Таблиця 7  - Опис, потрібний та фактичний результат тест-кейсів

|  |  |  |
| --- | --- | --- |
| Опис тест-кейсу | Потрібний вивід | Фактичний вивід |
| Можливо отримати список майстрів? | так | так |
| Можливо отримати список спецілізацій майстрів? | так | так |
| Можливо отримати список клієнтів? | так | так |
| Можливо отримати кількість клієнтів за вказаний період? | так | так |
| Можливо отримати найдорожчу послугу? | так | так |
| Можливо отримати середню вартість послуги? | так | так |
| Майстер має наступні властивості: назва пункту, ПІБ, Адреса, № телефону, спеціалізація? | так | так |
| Клієнт має наступні властивості: назва пункту, ПІБ, № телефону, ID послуги? | так | так |
| Послуга має наступні властивості: ID послуги, тип, назва, вартість, призначений час? | так | так |
| Об'єкти типу Майстер, Клієнт, Послуга не менш 3? | так | так |
| Абстрактний клас використовується у коді програми? | так | так |

Продовження таблиці 7

|  |  |  |
| --- | --- | --- |
| Існує композиція хоча б у одному класі? | так | так |
| Описано та реалізовано інтерфейс користувача? | так | так |

З результату таблиці 7 виводиться, що програма відповідає встановленим умовам, так як проходиться всі данні тест-кейси.

8 ВИСНОВКИ

У даній курсової роботі були поглибленні та закріпленні знання, одержаних при вивченні дисципліни «Об’єктно-орієнтоване програмування», та набуття практичних навичок у проектуванні та налагодженні програм, що застосовують класи та об’єкти.

Було розроблене програма з графічним користувацьким інтерфейсом, яка працює зі списками об'єктів. Інформація для списків зберігається в файлах. Кожен зі списків містить 3 об'єкту класу.

Були створені сутності «Майстер», «Клієнт», «Послуга» з відповідними та потрібними, через вимоги, властивостями.

Застосування оброблює шість запитів у вигляді методів та відповідних для них кнопці.

Реалізована композиція та успадкування. У роботі описано та реалізовано інтерфейс користувача.

Посилання на відео з демонстрацією роботи програми: https://drive.google.com/file/d/1a7Y\_u\_g3AVkThHo-KnzSDU4ibLKonfrg/view?usp=sharing

9 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Кунгурцев О.Б. Основи програмування на мові Java. Середовище Net Beans./О.Б. Кунгурцев – Одеса: ВМВ, 2006. – 182 с

2 Методичні вказівки до курсової роботи з дисципліни «Об’єктно – орієнтоване програмування» для студентів першого рівня вищої освіти (бакалавр). Галузь знань - 12 Інформаційні технології. Спеціальність 121 Інженерія програмного забезпечення / Укл.: О. Б. Кунгурцев. – Одеса: «Одеська політехніка», 2022. –24 с.

3 “Руководство по языку программирования Java” // metanit.com [2023]. Дата відновлення: 20.09.2023. URL: https://metanit.com/java/tutorial/ (дата звернення: 05.12.2023).

4 “Руководство по JavaFX” // metanit.com [2023]. Дата відновлення: 23.09.2021. URL: https://metanit.com/java/javafx/ (дата звернення: 05.12.2023).

ДОДАТОК А – КОД ПРОГРАМИ

Main.java

import Config.Config;

import Services.ClientService;

import Services.DefaultService;

import Services.OrderService;

import Services.StaffService;

public class Main {

public static void main(String[] args) {

Config.value = new Config("data/");

DefaultService.viewDefault();

}

}

Config.java

package Config;

public class Config {

public static Config value;

public String pathToDate;

public Config(String pathToDate) {

this.pathToDate = pathToDate;

value = this;

}

}

Order.java

package Models.Order;

import Models.Model;

public class Order extends Model {

private ProductType type;

private String name;

private Double cost;

private int serviceTime;

public Order(int id, ProductType type, String name, Double cost, int serviceTime) {

super(id);

this.type = type;

this.name = name;

this.cost = cost;

this.serviceTime = serviceTime;

}

public ProductType getType() {

return type;

}

public void setType(ProductType type) {

this.type = type;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Double getCost() {

return cost;

}

public void setCost(Double cost) {

this.cost = cost;

}

public int getServiceTime() {

return serviceTime;

}

public void setServiceTime(int serviceTime) {

this.serviceTime = serviceTime;

}

}

ProductType.java

package Models.Order;

public enum ProductType {

PHOTOCOPY,

COLORCOPY

}

Client.java

package Models;

public class Client extends Person{

private int orderId;

public Client(int id, String namePoint, String fullName, String phoneNumber, int orderId) {

super(id, namePoint, fullName, phoneNumber);

this.orderId = orderId;

}

public int getOrderId() {

return orderId;

}

}

Model.java

package Models;

import java.io.Serializable;

public abstract class Model implements Serializable {

private final int id;

public Model(int id) {

this.id = id;

}

public int getId() {

return this.id;

};

}

Person.java

package Models;

public abstract class Person extends Model{

private String namePoint;

private String fullName;

private String phoneNumber;

public Person(int id, String namePoint, String fullName, String phoneNumber) {

super(id);

this.namePoint = namePoint;

this.fullName = fullName;

this.phoneNumber = phoneNumber;

}

public String getNamePoint() {

return namePoint;

}

public void setNamePoint(String namePoint) {

this.namePoint = namePoint;

}

public String getFullName() {

return fullName;

}

public void setFullName(String fullName) {

this.fullName = fullName;

}

public String getPhoneNumber() {

return phoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

this.phoneNumber = phoneNumber;

}

}

Staff.java

package Models;

import Models.Person;

public class Staff extends Person {

private String address;

private String specialization;

public Staff(int id, String namePoint, String fullName, String phoneNumber, String address, String specialization) {

super(id, namePoint, fullName, phoneNumber);

this.address = address;

this.specialization = specialization;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getSpecialization() {

return specialization;

}

public void setSpecialization(String specialization) {

this.specialization = specialization;

}

}

CrudRepository.java

package Repository;

import Config.Config;

import Models.Model;

import java.io.\*;

import java.util.Objects;

public class CrudRepository<T extends Model>{

private final String type;

private final String fileName;

public CrudRepository(String type) {

this.type = type;

this.fileName = path + type + "Object.data";

}

private final String path = Config.value.pathToDate;

public void save(T entity) {

try {

System.out.println(fileName);

ObjectToFile<T> emp;

if (!new File(fileName).isFile()) {

File file = new File(fileName);

boolean created = file.createNewFile();

System.out.println(created);

emp = new ObjectToFile<T>();

} else {

FileInputStream is = new FileInputStream(fileName);

ObjectInputStream ois = new ObjectInputStream(is);

emp = (ObjectToFile<T>) ois.readObject();

}

emp.addData(entity);

FileOutputStream fos = new FileOutputStream(fileName);

ObjectOutputStream oos = new ObjectOutputStream(fos);

// write object to file

oos.writeObject(emp);

System.out.println("Done");

// closing resources

oos.close();

fos.close();

} catch (IOException e) {

e.printStackTrace();

} catch (ClassNotFoundException e) {

throw new RuntimeException(e);

}

}

public T findOne(Integer primaryKey) {

for (T object :

findAll().getData()) {

if(Objects.equals(object.getId(), primaryKey))

return object;

}

return null;

}

public ObjectToFile<T> findAll() {

try {

if (!new File(fileName).isFile()) {

return new ObjectToFile<T>();

}

FileInputStream is = new FileInputStream(path + type + "Object.data");

ObjectInputStream ois = new ObjectInputStream(is);

ObjectToFile<T> emp = (ObjectToFile<T>) ois.readObject();

ois.close();

is.close();

return emp;

} catch (IOException e) {

e.printStackTrace();

} catch (ClassNotFoundException e) {

throw new RuntimeException(e);

}

return null;

}

public int count() {

return findAll().getData().size();

}

public void delete(T entity) {

try {

ObjectToFile<T> emp;

if (!new File(fileName).isFile()) {

throw new RuntimeException("Нечего удолять.");

}

FileInputStream is = new FileInputStream(fileName);

ObjectInputStream ois = new ObjectInputStream(is);

emp = (ObjectToFile<T>) ois.readObject();

emp.getData().remove(entity.getId());

FileOutputStream fos = new FileOutputStream(fileName);

ObjectOutputStream oos = new ObjectOutputStream(fos);

// write object to file

oos.writeObject(emp);

System.out.println("Done");

// closing resources

oos.close();

fos.close();

} catch (IOException e) {

e.printStackTrace();

} catch (ClassNotFoundException e) {

throw new RuntimeException(e);

}

}

public boolean exists(Integer primaryKey) {

return findOne(primaryKey) != null;

}

}

ObjectToFile.java

package Repository;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

public class ObjectToFile<T> implements Serializable {

private Integer lastId = 0;

private final List<T> data;

public ObjectToFile() {

this.data = new ArrayList<>();

lastId = 0;

}

public List<T> getData() {

return data;

}

public void setLastId(Integer lastId) {

this.lastId = lastId;

}

public void addData(T data) {

this.setLastId(lastId);

lastId++;

this.data.add(data);

}

}

ClientService.java

package Services;

import Models.Client;

import Models.Order.Order;

import Repository.CrudRepository;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ScrollPane;

import javafx.scene.control.TextField;

import javafx.scene.layout.VBox;

import javafx.scene.text.Text;

import javafx.stage.Stage;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.atomic.AtomicInteger;

public class ClientService {

protected static final CrudRepository<Client> crudRepository = new CrudRepository<Client>(Client.class.getSimpleName());

protected static final CrudRepository<Order> crudRepositoryOrder = new CrudRepository<Order>(Order.class.getSimpleName());

public static void viewListClients(Stage stage) {

List<Client> clients = crudRepository.findAll().getData();

if(clients.size() == 0) {

String text = "Список порожній";

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(new Text(text.toString()), btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

return;

}

Client first = clients.get(0);

StringBuilder text = new StringBuilder("Номер: " + first.getId()

+ "\nПІБ: " + DefaultService.convertWinToUtf(first.getFullName())

+ "\nНазва пункту: " + DefaultService.convertWinToUtf(first.getNamePoint())

+ "\nТелефоний номер: " + DefaultService.convertWinToUtf(first.getPhoneNumber())

+ "\nНомер замовлення: " + first.getOrderId()

);

clients.stream().skip(1).forEach(o -> text.append("\n\nНомер: " + o.getId()

+ "\nПІБ: " + DefaultService.convertWinToUtf(o.getFullName())

+ "\nНазва пункту: " + DefaultService.convertWinToUtf(o.getNamePoint())

+ "\nТелефоний номер: " + DefaultService.convertWinToUtf(o.getPhoneNumber())

+ "\nНомер замовлення: " + o.getOrderId()

));

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(new Text(text.toString()), btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

}

public static void viewCountClientsWithTime(Stage stage) {

List<Client> clients = crudRepository.findAll().getData();

List<Order> orders = crudRepositoryOrder.findAll().getData();

if(clients.size() == 0 || orders.size() == 0 ) {

String text = "Список порожній";

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(new Text(text.toString()), btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

return;

}

Text text1 = new Text("Введіть потрібний термін:");

TextField textField = new TextField();

AtomicInteger serviceTime = new AtomicInteger();

ArrayList<Client> clientsWithThisTimes = new ArrayList<>();

Button accept = new Button("Знайти");

Text text2 = new Text("\nКількість клієнтів з даним періодом: 0");

accept.setOnAction(actionEvent -> {

clientsWithThisTimes.clear();

try {

serviceTime.set(Integer.parseInt(textField.getText()));

for (Client client:

clients) {

for (Order order :

orders) {

if(order.getServiceTime() == serviceTime.get() &&

client.getOrderId() == order.getId())

clientsWithThisTimes.add(client);

}

}

text2.setText("\nКількість клієнтів з даним періодом: "

+ clientsWithThisTimes.size());

} catch (NumberFormatException e) {

text2.setText("\nВведіть тільки ціле числове значення!\n\nКількість клієнтів з даним періодом:"

+ clientsWithThisTimes.size());

}

});

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(text1, textField, accept, text2, btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

}

}

DefaultService.java

package Services;

import java.util.Scanner;

public class DefaultService {

public static void viewDefault() {

System.out.println("""

Автоматизація пункту ксерокпоії.

Команди (Нажміть відповідну цифру та Enter):

1)Список майстрів.

2)Список спеціалізацій майстрів.

3)Список клієнтів.

4)Кількість клієнтів, яким призначена послуга на вказаний період.

5)Найдорожча послуга.

6)Середня вартість послуги.

""");

Scanner scanner = new Scanner(System.in);

switch (scanner.nextShort()) {

case(1):

StaffService.viewListStaff();

case(2):

StaffService.viewSpecializations();

case(3):

ClientService.viewListClients();

case(4):

ClientService.viewCountClientsWithTime();

case(5):

OrderService.viewMaxCostOrder();

case(6):

OrderService.viewMiddleCost();

}

}

}

OrderService.java

package Services;

import Models.Order.Order;

import Repository.CrudRepository;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ScrollPane;

import javafx.scene.layout.VBox;

import javafx.scene.text.Text;

import javafx.stage.Stage;

import java.util.List;

public class OrderService {

protected static final CrudRepository<Order> crudRepository = new CrudRepository<Order>(Order.class.getSimpleName());

public static void viewMaxCostOrder(Stage stage) {

List<Order> orders = crudRepository.findAll().getData();

if(orders.size() == 0) {

String text = "Список порожній";

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(new Text(text.toString()), btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

return;

}

Order max = orders.get(0);

for (Order order :

orders) {

if (order.getCost() > max.getCost())

max = order;

}

String text = "Найдорожча послуга:" +

"\nНомер послуги: " + max.getId()

+ "\nНазва: " + DefaultService.convertWinToUtf(max.getName())

+ "\nВартість " + max.getCost()

+ "\nТип: " + max.getType().name()

+ "\nПеріод виконання: " + max.getServiceTime();

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(new Text(text), btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

}

public static void viewMiddleCost(Stage stage) {

List<Order> orders = crudRepository.findAll().getData();

if(orders.size() == 0) {

String text = "Список порожній";

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(new Text(text.toString()), btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

return;

}

Double middleCost = 0.0;

for (Order order:

orders) {

middleCost += order.getCost();

}

middleCost = middleCost / orders.size();

String text = String.format("Середня вартість: %.2f", middleCost);

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(new Text(text), btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

}

}

StaffService.java

package Services;

import Models.Staff;

import Repository.CrudRepository;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ScrollPane;

import javafx.scene.layout.VBox;

import javafx.scene.text.Text;

import javafx.stage.Stage;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

public class StaffService {

protected static final CrudRepository<Staff> crudRepository = new CrudRepository<Staff>(Staff.class.getSimpleName());

public static void viewListStaff(Stage stage) {

List<Staff> staffs = crudRepository.findAll().getData();

if(staffs.size() == 0) {

String text = "Список порожній";

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(new Text(text.toString()), btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

return;

}

Staff first = staffs.get(0);

StringBuilder text = new StringBuilder("Номер: " + first.getId()

+ "\nПІБ: " + DefaultService.convertWinToUtf(first.getFullName())

+ "\n" + DefaultService.convertWinToUtf(first.getSpecialization())

+ "\nНазва пункту: " + DefaultService.convertWinToUtf(first.getNamePoint())

+ "\nАдреса: " + DefaultService.convertWinToUtf(first.getAddress())

+ "\nТелефоний номер: " + DefaultService.convertWinToUtf(first.getPhoneNumber())

);

staffs.stream().skip(1).forEach(o -> text.append("\n\nНомер: " + o.getId()

+ "\nПІБ: " + DefaultService.convertWinToUtf(o.getFullName())

+ "\n" + DefaultService.convertWinToUtf(o.getSpecialization())

+ "\nНазва пункту: " + DefaultService.convertWinToUtf(o.getNamePoint())

+ "\nАдреса: " + DefaultService.convertWinToUtf(o.getAddress())

+ "\nТелефоний номер: " + DefaultService.convertWinToUtf(o.getPhoneNumber())

));

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(new Text(text.toString()), btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

}

public static void viewSpecializations(Stage stage) {

List<Staff> staffs = crudRepository.findAll().getData();

if(staffs.size() == 0) {

String text = "Список порожній";

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(new Text(text.toString()), btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

return;

}

ArrayList<String> specializations = new ArrayList<>();

sta: for (Staff staff:

staffs) {

for (String specialization :

specializations) {

if (Objects.equals(specialization, staff.getSpecialization())) {

continue sta;

}

}

specializations.add(staff.getSpecialization());

}

StringBuilder text = new StringBuilder();

specializations.forEach(s -> text.append(DefaultService.convertWinToUtf(s)).append("\n"));

Button btn = new Button("Повернутись");

btn.setOnAction(actionEvent -> DefaultService.viewDefault(stage));

VBox vbox = new VBox(new Text(text.toString()), btn);

ScrollPane scrollPane = new ScrollPane(vbox);

Scene scene = new Scene(scrollPane);

stage.setScene(scene);

stage.show();

}