

Черкаський національний університет імені Богдана Хмельницького

Кафедра програмного забезпечення автоматизованих систем

## КУРСОВА РОБОТА

з дисципліни «Об'єктно-орієнтоване програмування»

НА ТЕМУ «Моделювання роботи станції швидкої допомоги»

Студента \_\_\_\_\_ 2 \_\_\_\_\_ курсу, групи \_\_\_\_\_ КС-19

спеціальності \_\_\_\_\_ «Інженерія

\_\_\_\_\_ програмного забезпечення»

\_\_\_\_\_ Джеріхова Івана Олеговича

Керівник \_\_\_\_\_ старший викладач Гребенович  
Ю.Є. \_\_\_\_\_

Національна шкала \_\_\_\_\_

Кількість балів: \_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_

Члени комісії

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(прізвище та ініціали)

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(прізвище та ініціали)

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(прізвище та ініціали)

м. Черкаси – 2021 рік

## Зміст

Вступ.....	4
Розділ 1. Огляд систем масового обслуговування. Вибір технологій для реалізації програми. ....	6
1.1. Теорія масового обслуговування. Системи масового обслуговування (СМО), їх види.....	6
1.2. Розгляд деяких існуючих моделей СМО .....	8
1.3. Вибір системи масового обслуговування для поставленої задачі	9
1.4. Вибір патерну проектування програми .....	10
1.5. Обґрунтування вибору мови програмування та середовища розробки	11
1.6. Висновки до розділу .....	11
Розділ 2. Проектування станції швидкої допомоги .....	13
2.1. Проектування ієрархії класів .....	13
2.2. Спрощений алгоритм роботи станції ШМД.....	14
2.3. Опис алгоритму надходження заявки пацієнта до системи .....	15
2.4. Опис алгоритму призначення бригади на виклик .....	16
2.5. Опис алгоритму емуляції поїздки на виклик .....	16
2.6. Реалізація принципів ООП, діаграма класів.....	17
2.7. Висновки до розділу .....	18
Розділ 3. Програмна реалізація продукту .....	19
3.1. Реалізація контролерів бригад та пацієнтів.....	19
3.2. Реалізація моделей .....	20
3.3. Реалізація файлового менеджера .....	20
3.4. Реалізація та огляд графічного інтерфейсу .....	20

3.5. Тестування програми .....	24
3.6. Висновки стосовно реалізації .....	27
Висновки .....	28
Список використаної літератури .....	29

## Вступ

Здоров'я – найцінніша річ, що є в людини. З давніх часів підтримання свого тіла і духу у тонусі були невід'ємною частиною культури різних народів. Звичайні для наших часів хвороби лікували народними методами. Проте в ситуаціях, коли здоров'ю було завдано серйозної шкоди і потребувалась екстрена допомога, щось зробити самому було майже неможливо. Це і стало головною причиною появи станцій швидкої медичної допомоги.

Перший прототип станції з'явився ще в XI ст. в Єрусалимі ченцями при монастирі св. Іоана, з якого через декілька століть візьме свій початок орден Госпітальєрів, завдяки якому у хрестових походах була опіка над пораненими бійцями.

Першу стаціонарну станцію швидкої медичної допомоги створив віденський професор Яромир Мунді у 1883 році. Уряд Австрії виділив фінанси на цей заклад, оскільки кількома роками раніше через відсутність швидкої допомоги загинуло близько 500 глядачів “Комічної опери” у результаті пожежі.

В Україні перша станція відкрилась у 1902 році. Транспорт закладу складався з трьох пар коней з каретами, через десять років було додано 2 санітарні автомобілі. [1]

З тих часів багато чого змінилось. Зараз станції швидкої допомоги є майже у кожному населеному пункті. Диспетчерська служба викликається за єдиним номером в усій країні (103 або 112 для всіх екстрених служб).

З теоретичної точки зору, робота станції ШМД є одним з прикладів системи масового обслуговування (СМО). Існує декілька видів та критеріїв таких систем, які будуть розглянуті далі.

Тему курсової роботи було обрано через цікавість та перспективність поставленої задачі.

Мета роботи: дослідження існуючих систем масового обслуговування та власна програмна реалізація системи станції швидкої медичної допомоги.

Завдання даної роботи: проаналізувати інформаційні джерела, описати алгоритм роботи станції, написати програму емуляції роботи станції швидкої медичної допомоги, провести тестування продукту.

## Розділ 1. Огляд систем масового обслуговування. Вибір технологій для реалізації програми.

### 1.1. Теорія масового обслуговування. Системи масового обслуговування (СМО), їх види.

Теорія масового обслуговування – це наука, що досліджує вибір структури системи обслуговування та процесів обслуговування, виходячи з параметрів потоків обслуговування, очікування, черг та ін..

Перші проблеми теорії почав розглядати Агнер Ерланг, співробітник телефонної компанії в Копенгагені на початку XX століття, якому доручили завдання упорядкування роботи телефонної станції.[2]

Система масового обслуговування (СМО) - це система, яка обслуговує заявки, що надходять до неї.

Характерні риси СМО (рис. 1.1):

1. Випадкове надходження потоку заявок на обслуговування.
2. Черги заявок.
3. Пристрої, які здійснюють обслуговування цих заявок.

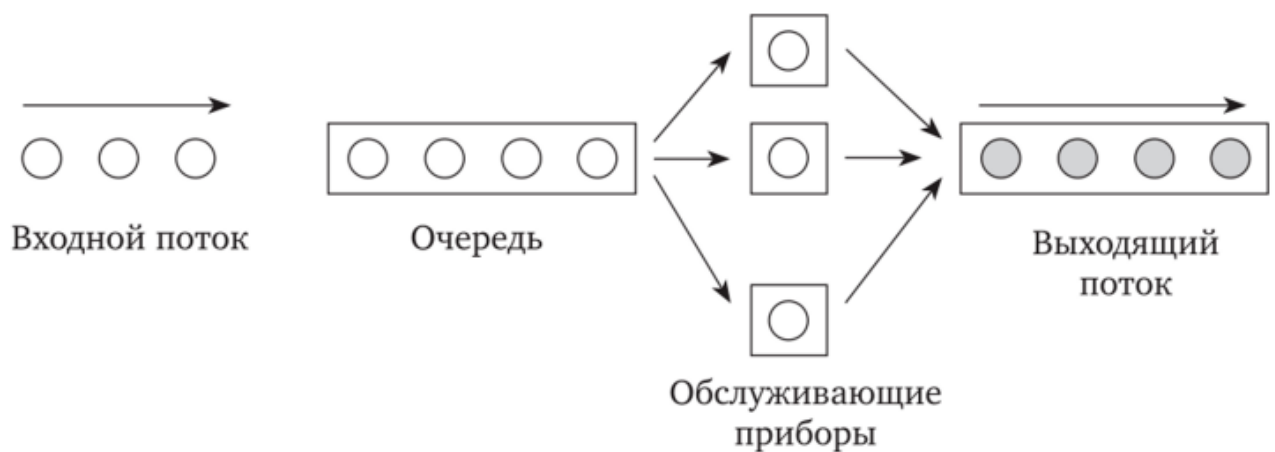


Рис 1.1. Схема роботи системи масового обслуговування

Прикладами систем масового обслуговування є станція швидкої медичної допомоги, телефонна станція, диспетчерська служба таксі.

Існує велика кількість класифікацій систем масового обслуговування за різними ознаками.

1. За кількістю каналів обслуговування:

- a. одноканальна СМО(один канал для обробки заявок).
- b. багатоканальна СМО, яка дозволяє обробляти декілька заявок одночасно.

2. За наявністю черг:

- a. СМО без черги (з відмовами). При надходженні заявки у момент зайнятості усіх каналів, заявка втрачається і не обслуговується.
- b. СМО з чергою (з очікуванням). При надходженні заявки у момент зайнятості усіх каналів, заявка стає в чергу і чекає свого часу для обслуговування.

У той час, за характеристиками черга поділяється на:

- i. за довжиною черги накопичувача: обмежена або безлімітна. При обмеженій довжині черги заявка зникає після досягнення ліміту накопичувача.
- ii. за часом очікування: обмежений або необмежений. При варіанті з обмеженим часом очікування заявка втрачається після певного проміжку, за який вона не була обслугована. Необмежений час передбачає відсутність втрати заявки незалежно від часу очікування.
- iii. за принципом обробки заявок: обслуговування першої заявки в першу чергу (принцип FIFO), обслуговування останньої заявки в першу чергу (LIFO), обслуговування заявок у випадковому порядку (FIRO), обслуговування з пріоритетами.[3]

## 1.2. Розгляд деяких існуючих моделей СМО

**Одноканальна система з відмовами** (рис. 1.2). Прикладом такої черги є, наприклад директор певної компанії, який обслуговує різного виду заявки (розмови з підлеглими, заповнення документів та ін.)

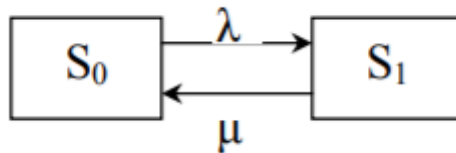


Рис 1.2. Схема роботи одноканальної СМО з відмовами

Кількість можливих станів такої СМО – 2, канал або вільний і готовий прийняти заявку, або зайнятий і заявки відхиляються.

**Одноканальна система з необмеженою чергою** (рис. 1.3). Прикладом такої системи можна назвати телефонну будку з одним апаратом всередині.

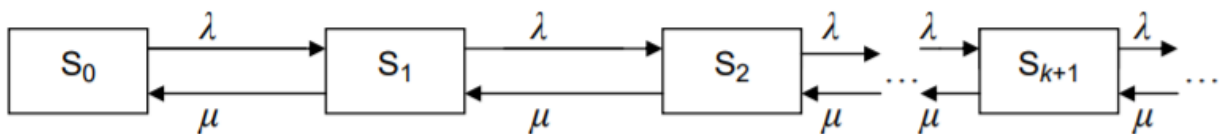


Рис 1.3. Схема роботи одноканальної СМО з необмеженою чергою

Кількість можливих станів такої СМО безліч через необмежену довжину черги.

**Багатоканальна система з відмовами** (рис. 1.4). Прикладом такої системи є телефонна станція (АТС).

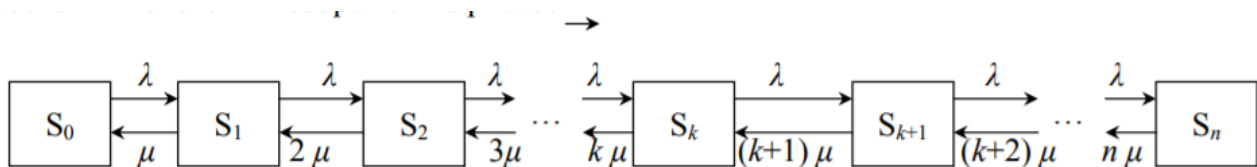


Рис 1.4. Схема роботи багатоканальної СМО з відмовами



Стани системи змінюються від усіх вільних каналів ( $S_0$ ) до стану  $S_n$  з усіма зайнятими каналами.

**Багатоканальна система з обмеженою чергою** (рис. 1.5.). Прикладом також може слугувати АТС.

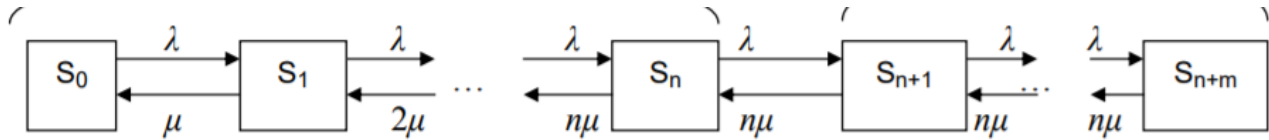


Рис 1.5. Схема роботи багатоканальної СМО з обмеженою чергою

Стани системи змінюються від усіх вільних каналів ( $S_0$ ) до стану  $S_n$  з усіма зайнятими каналами. Потім починається черга при стані  $S_{n+1}$  і закінчується станом  $S_{n+m}$ , де  $m$  – довжина черги.

**Багатоканальна система з необмеженою чергою** (рис. 1.6). Прикладом слугують ті сервіси, у яких відмовити в обслуговуванні заявки не можна.

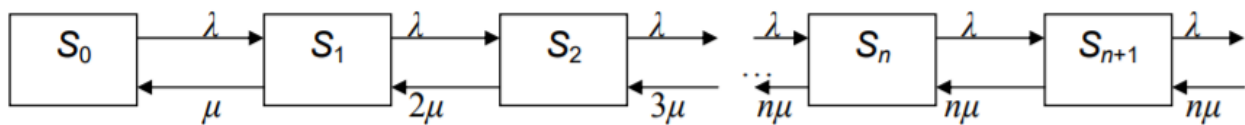


Рис 1.4. Схема роботи багатоканальної СМО з обмеженою чергою

Така система має безліч станів через необмежену довжину черги. змінюються від усіх вільних каналів ( $S_0$ ) до стану  $S_n$  з усіма зайнятими каналами.

### 1.3. Вибір системи масового обслуговування для поставленої задачі

Для вибору СМО для моделювання роботи станції швидкої допомоги слід розглянути описані в розділах 1.1 та 1.2 критерії.

За кількістю каналів обслуговування підходить багатоканальний варіант. Одноканальна система була відкинута через специфіку роботи швидкої допомоги, яка мусить обслуговувати всіх клієнтів в декілька паралельних потоків.

За наявністю черги слід обрати варіант з наявністю такої, оскільки через те, що мова йде про життя людей, відмови в обслуговуванні вимоги не може бути. Це означає, що втрати заявок є недопустимими.

Довжина черги накопичувача має бути умовно безлімітною, знову ж таки, через неможливість відмови.

Був обраний принцип роботи із заявками під назвою FIRO з деякими поправками. Заявка вибирається не стовідсотково випадково, а на розгляд диспетчера системи.

Команда медиків являтиме собою канал СМО, а всі об'єкти бригад, відповідно, складають усю множину каналів. В свою чергу, пацієнт – це заявка, яка поступає до СМО. Його «обробкою», тобто лікуванням, і займається кожна з команд-каналів.

Кількість каналів регулюватиметься зміною кількості бригад медиків.

#### **1.4. Вибір патерну проектування програми**

При проектуванні програми орієнтиром буде патерн MVC (Model-View-Controller). MVC – схема розділу додатку на три частини: модель надає данні та виконує команди контролера, змінюючи свій стан, представлення(вид) – відображення даних моделей користувачу, реагуючи на зміни моделей, контролер – сповіщує модель про зміни при певних діях користувача з частиною представлення[4]. Рис. 1.5. схематично пояснює вище описаний текст.

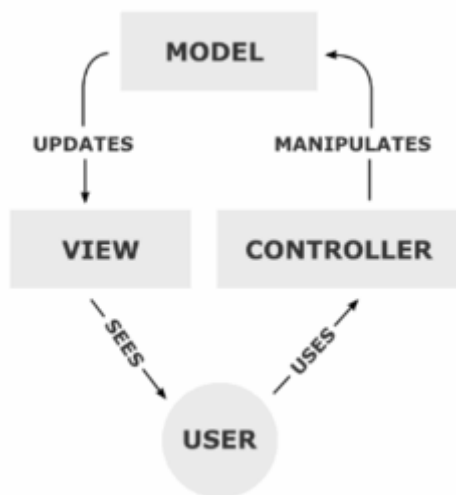


Рис. 1.5. Схема роботи програм з патерном MVC

### 1.5. Обґрунтування вибору мови програмування та середовища розробки

Вибір мови програмування, яку буду використовувати для написання програмного продукту, пав на C#, оскільки інші не вивчав.

Для створення графічного інтерфейсу планується використання бібліотеки Windows Forms. Вона є простою у освоєнні, її ресурсів цілком вистачить для створення MDI-додатку з патерном MVC. Іншим варіантом було використання фреймворку ASP.NET MVC, який призначений для створення проектів з патерном MVC, але більше підходить для веб-додатків.

За середовище розробки було обране Microsoft Visual Studio 2019 Community. Перевагою є безкоштовність, що ідеально підходить для розробки проектів у навчальних цілях.

Середовище JetBrains Rider відпало через відсутність нормальної роботи з Windows Forms.

### 1.6. Висновки до розділу

В даному розділі розглянуті теоретичні відомості щодо систем масового обслуговування. Розглянуто основні моделі систем та обґрунтований вибір однієї з них, а саме багатоканальної СМО з нескінченною чергою. Виконаний вибір патерна проектування програми.

Також обрано та обґрунтовано мову програмування, бібліотеку для створення графічного інтерфейсу та середовище розробки.

## Розділ 2. Проектування станції швидкої допомоги

### 2.1. Проектування ієрархії класів

Для моделювання станції швидкої допомоги за патерном MVC треба спроектувати ієрархію класів в залежності від типу класу.

#### Моделі:

**Клас Brigade.** Являтиме собою абстрактний клас з базовим конструктором властивостей, таких як id бригади, спеціалізація, статус доступності, швидкість карети швидкої допомоги та кількість вилікуваних пацієнтів. Крім того, матимуться абстрактні методи емуляції поїздки та лікування хворого, які поліморфно реалізоватимуться у нащадках.

**Класи HQBrigade та LQBrigade.** Існуватиме два типи спеціалізацій бригад для двох типів заявок: для хвороб високої складності або екстрених випадків – висококваліфікована бригада (далі – HQ), для рядових викликів – звичайна бригада (далі – LQ). Кожна з бригад може обслуговувати лише заявку своєї складності. Класи HQ та LQ-бригад унаслідуються від абстрактного класу Brigade.

**Patient, PatientGenerator.** У першому будуть описані властивості пацієнта, такі як ПІБ, час виклику, дистанція від нього до станції ШМД, хвороба, з якої визначиться пріоритет виклику. Клас PatientGenerator являтиме собою генератор рандомних пацієнтів, у якому випадково будуть надані значення властивостям, що описані в класі Patient. Також тут буде задане часове значення появи нової заявки пацієнта.

Клас **Illness** являтиме собою модель хвороби, що матиме поля назви, пріоритетності та часу лікування цієї хвороби.

Клас **BaseObjectInizializer** відповідатиме за створення базових списків бригад та хвороб, якщо такі відсутні.

#### Контролери:

**BrigadeController** – клас-контролер, який здійснюватиме різні дії над об'єктами бригад, такі як створення або розформування, підвищення рангу бригади, відправлення бригади на виклик.

**PatientController** відповідатиме за надходження заявок пацієнтів до станції, викликаючи метод створення пацієнта з класу **PatientGenerator**.

**Представлення:**

1. Форма зі списком об'єктів пацієнтів
2. Форма зі списком об'єктів бригад
3. Форма призначення бригади на виклик

**Окремі класи:**

Необхідно передбачити серіалізацію списку об'єктів бригад з їх властивостями, оскільки продукт передбачатиме створення нових об'єктів, інформація про які не повинна зникнути. За це відповідатиме клас **FileManager**, у якому будуть методи збереження та загрузки списку бригад з файлу.

## **2.2. Спрощений алгоритм роботи станції ШМД**

На рис. 2.1. зображений спрощений алгоритм роботи станції ШМД. На блок-схемі не враховані тупикові ситуації, як-то нестача бригад, але в програмі буде реалізована черга очікування бригади.

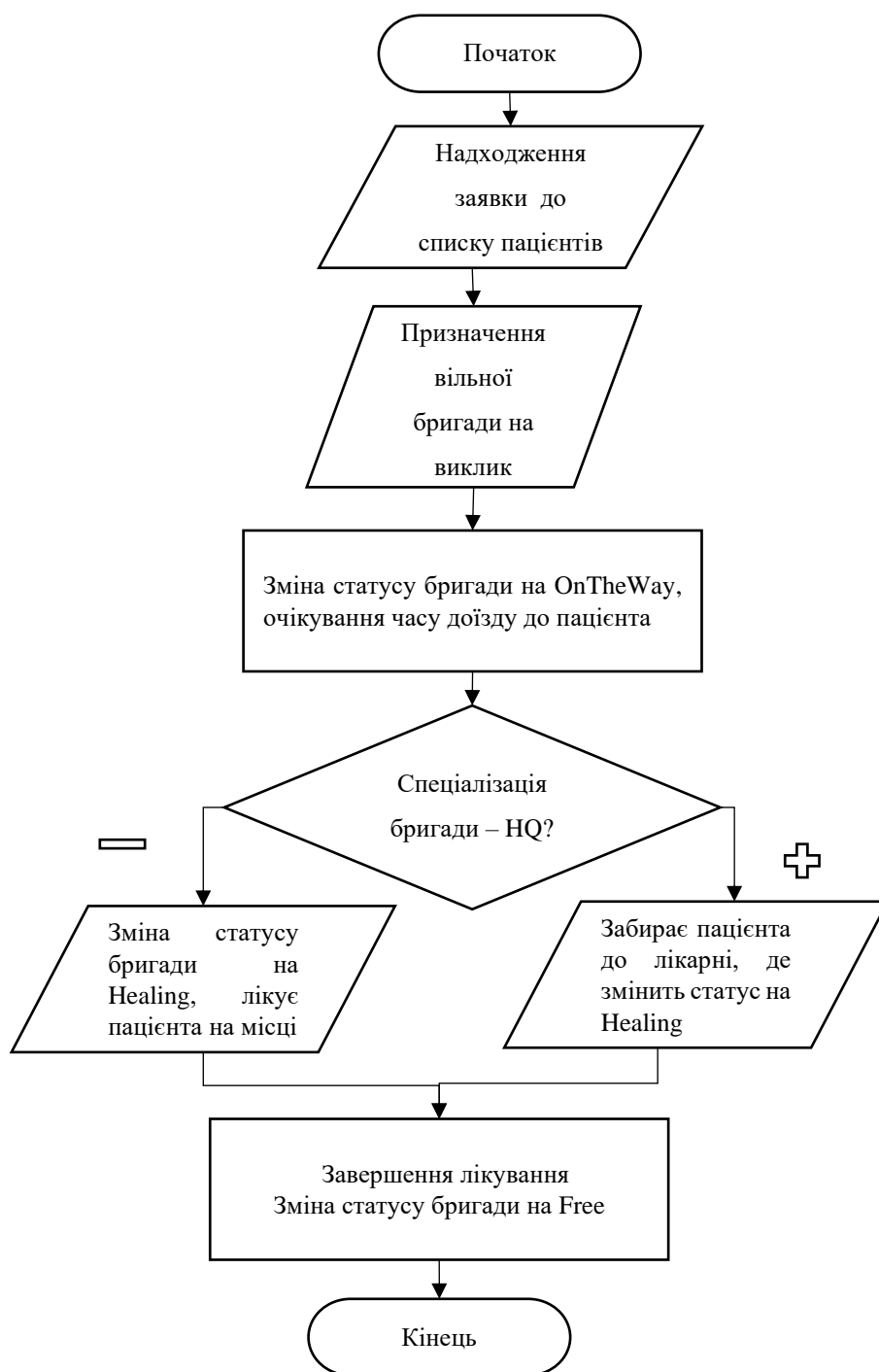


Рис 2.1. Блок-схема спрощеного алгоритму роботи станції

### 2.3. Опис алгоритму надходження заявки пацієнта до системи

Для емуляції надходження заявки пацієнта до станції ШМД було придумано такий алгоритм:

1. З заздалегідь створеного списку перелічень імен та прізвищ випадковим чином обирається по одному.
2. Випадковим чином задається хвороба пацієнта та дистанція до нього.
3. Задається інтервал появи нового пацієнта методом випадкового вибору (від 15 до 30 секунд).
4. Після закінчення часового інтервалу очікування заявка потрапляє до списку пацієнтів.

Блок-схема алгоритму знаходиться в додатку А.

## **2.4. Опис алгоритму призначення бригади на виклик**

Після того, як до станції надійшла заявка пацієнта, треба призначити бригаду на виклик.

Для цього був розроблений алгоритм, який обирає бригаду в залежності від хвороби пацієнта.

Відсилається запит на виведення списку доступних бригад. Якщо зараз вільних бригад потрібної спеціалізації немає, діалогове вікно повідомить про це.

При наявності екіпажів відкривається форма зі списком вільних команд. Після підтвердження вибору бригади в окремому потоці запускається метод емуляції поїздки до пацієнта.

Блок-схема алгоритму знаходиться в додатку А.

## **2.5. Опис алгоритму емуляції поїздки на виклик**

Абстрактний метод емуляції поїздки заданий в абстрактному класі *Brigade*, його реалізація в класах *HQBrigade* та *LQBrigade* дещо відрізняється.

Бригада призначена на виклик, починається емуляція поїздки екіпажа до пацієнта. Спочатку вираховується час подорожі екіпажу в одну та дві сторони, після чого у *MessageBox* для *LQBrigade* виводиться повідомлення про приблизний час зайнятості бригади.

Потім викликається затримка виконання асинхронного методу лікування пацієнта. Часовий інтервал затримки дорівнює часу подорожі до пацієнта для



об'єкту LQBrigade та часу подорожі туди-назад для об'єкту HQBrigade. Після цього для об'єкту HQBrigade виводиться MessageBox з приблизним часом лікування пацієнта. Після завершення лікування пацієнта виводиться відповідне повідомлення, змінюється статус бригади на Free, а об'єкту пацієнта присвоюється значення null.

Різниця методів об'єктів HQBrigade та LQBrigade полягає у тому, що у першому випадку лікування пацієнта відбувається у лікарні, а для другого об'єкта процес лікування відбувається вдома у пацієнта. Відповідно, затримки для емуляції подорожі або лікування у цих методах викликаються в різні моменти.

Блок-схеми для об'єкту HQBrigade розташована в додатку А.

## 2.6. Реалізація принципів ООП, діаграма класів

Одним з прикладів інкапсуляції (приховання деталей реалізації методу) у даному програмному продукті є надання полям класу FileManager модифікатора доступу private (рис. 2.2.)

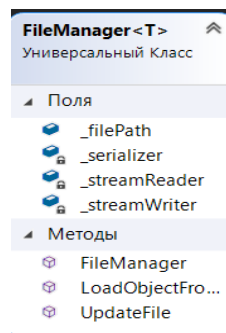


Рис. 2.2. Приклад інкапсуляції

Наслідування (успадкування класами методів чи полів іншого класу) в цьому проекті представлено абстрактним батьківським класом Brigade та його нащадками HQBrigade та LQBrigade, які використовують базовий конструктор.

Поліморфізм в цій програмі використовується для перевизначення методів класу Brigade в бригадах-нащадках.

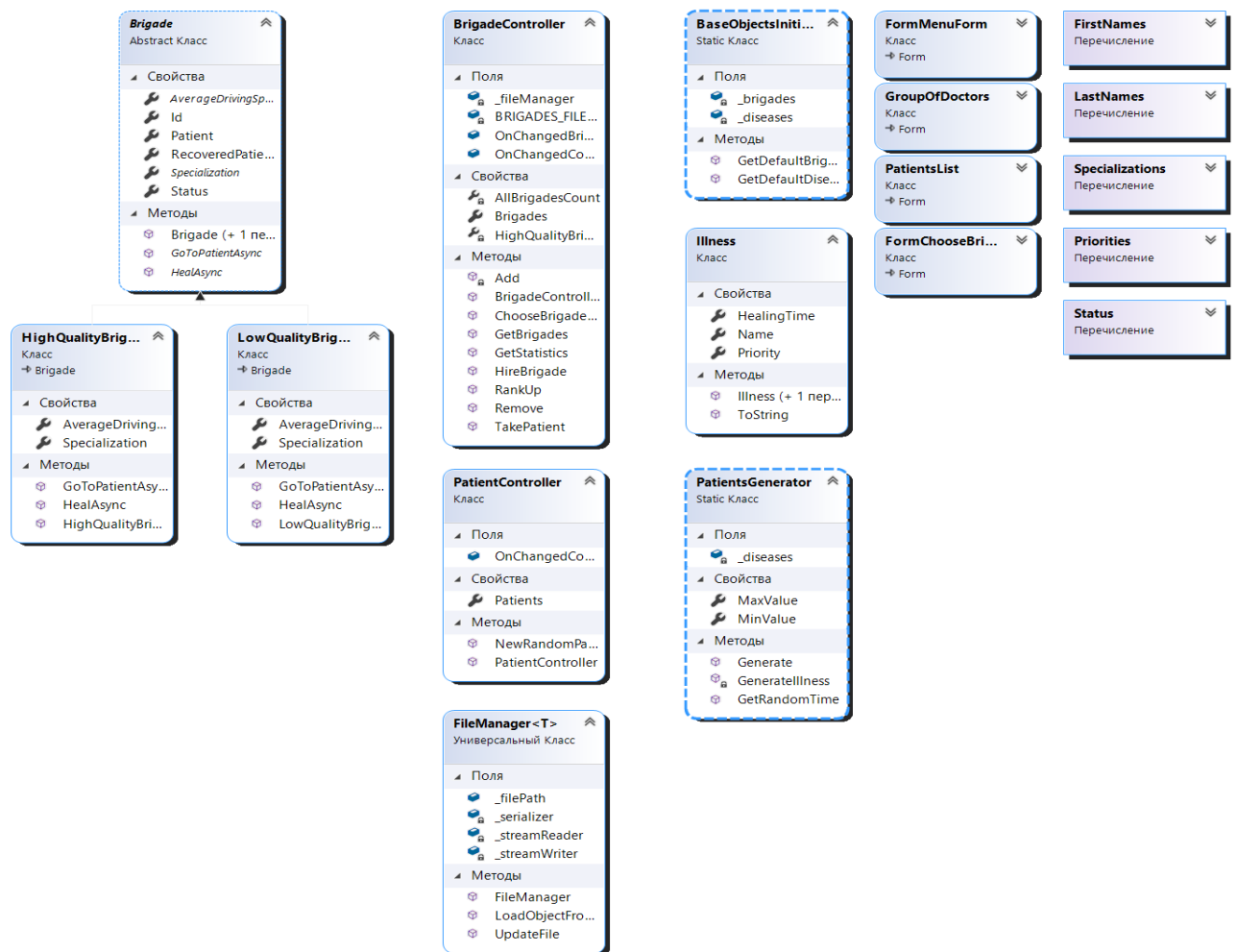


Рис. 2.3. Діаграма класів проекту

Всю ієрархію класів можна побачити на діаграмі класів (рис. 2.3.)

## 2.7. Висновки до розділу

В даному розділі було спроектовано ієрархію класів програми, описані основні алгоритми, пояснена реалізація принципів ООП та побудована діаграма класів.

## Розділ 3. Програмна реалізація продукту

### 3.1. Реалізація контролерів бригад та пацієнтів

У контролерах прописана логіка керування об'єктами-моделями.

Клас `BrigadeController` дозволяє виконувати різні операції над об'єктами бригад. Наявні методи підвищення рангу бригади, створення та розформування бригади, метод призначення бригади на виклик, метод емуляції поїздки бригади. Розглянемо детальніше деякі з них.

Метод `Add`, що відповідає за додання нової бригади до списку, в якості параметра приймає будь-яку кількість об'єктів бригад. Також в якості параметра можна передати як `HQBrigade`, так і `LQBrigade`, які зміняться на `Brigade`. У методі додається нова бригада до списку, оновлюється файл зі списком та викликається делегат `OnChangedCount`, який сповіщає усіх підписників про зміни в кількості бригад.

Асинхронний метод `TakePatient` відповідає за емуляцію поїздки бригади на виклик. Всередині нього викликаються методи бригад `GoToPatientAsync` та `HealAsync`, описані у розділі 2.5.

Лістинг класу в додатку Б.1.

Контролер пацієнтів `PatientController` відповідає за дії з об'єктами пацієнтів, а саме на рандомне надходження заявок та додання до списку. Розглянемо метод `NewRandomPatient`. Відбувається звернення до генератора рандомних пацієнтів, робота якого була описана в розділі 2.3., після додавання пацієнту до списку видається звуковий сигнал, що сповіщує про нову заявку, а делегат `OnChangedCount` сповіщає підписників про зміну кількості пацієнтів.

Лістинг класу в додатку Б.2.

### 3.2. Реалізація моделей

Класи моделей (Brigade, HQBrigade, LQBrigade, Patient, PatientGenerator, BaseObjectGenerator) були реалізовані за проектувальним описом з розділу 2.1, деякі з них також і за описами алгоритмів з розділів 2.3-2.5.

Лістинги цих класів розташовані у додатку Б.3-9.

### 3.3. Реалізація файлового менеджера

FileManager являє собою параметризований клас, який відповідає за збереження та подальше зчитування списку бригад з XML-файлу. Метод UpdateFile оновлює об'єкти у файлі за допомогою серіалізатора, а метод LoadObjectFromFile десеріалізує об'єкти з файлу. До файлу серіалізуються такі атрибути бригад, як id, спеціалізація, статус та кількість вилікуваних пацієнтів. Вигляд запису об'єкту бригади у файлі зображено на рис. 2.1.

```
<Brigade xsi:type="HighQualityBrigade">  
  <Id>1</Id>  
  <Specialization>HighQualityBrigade</Specialization>  
  <Status>Free</Status>  
  <RecoveredPatients>3</RecoveredPatients>  
  <AverageDrivingSpeed>100</AverageDrivingSpeed>  
</Brigade>
```

Рис. 2.1. Вигляд запису бригади в XML-файлі.

Лістинг класу можна побачити у додатку Б.10.

### 3.4. Реалізація та огляд графічного інтерфейсу

Користувацький інтерфейс продукту складається з чотирьох форм.

Стартове вікно – форма FormMenuForm (рис. 3.2).

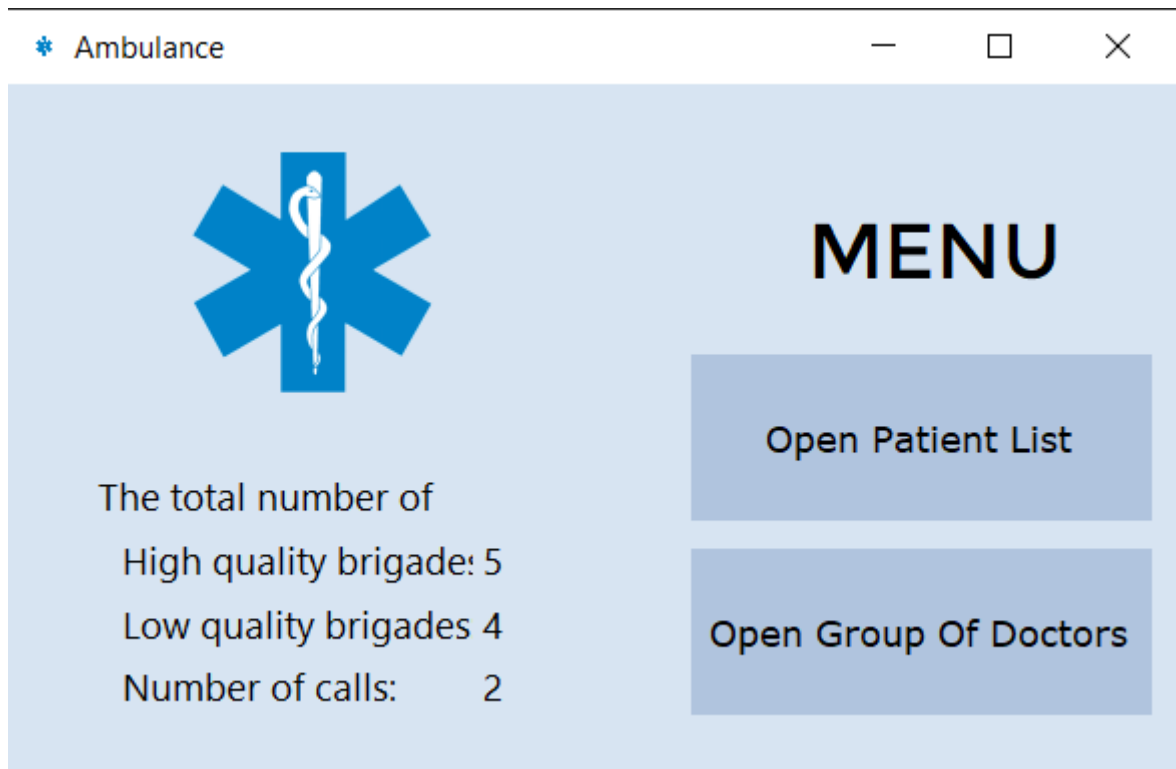


Рис. 3.2. Стартова форма програми

Вікно містить кнопки з написами «Open Patient List» та «Open Group Of Doctors», які відповідають за відкриття форм зі списками пацієнтів та бригад. Крім того, наявне так зване інформаційне табло, на якому виводиться загальна кількість бригад кожного типу, а також кількість непризначених заявок. Табло працює за рахунок змін значень делегату OnChangedCount, які відбуваються у контролерах бригад та пацієнтів при створенні або видаленні нових об'єктів.

Форма FormPatientList зображає інформацію щодо заявок пацієнтів, що надходять до програми (рис. 3.3):

List of receipts					Set Brigade
CallTime	FullName	Distance	Illness	Priority	
25.05 16:38:22	Олійник Богдан	9	Грип	High	
25.05 16:38:22	Савченко Давид	34	Онко	Low	

Рис. 3.3. Форма заявок пацієнтів

Список заявок реалізований за допомогою вбудованого до Windows Forms класу DataGridView, що дозволяє виводити дані зі списку пацієнтів BindingList у кастомізовану таблицю.

Сам BindingList – це не простий список, а список, до якого прив'язаний DataGridView, що дозволяє при кожній зміні в BindingList автоматично додавати\видаляти об'єкти.

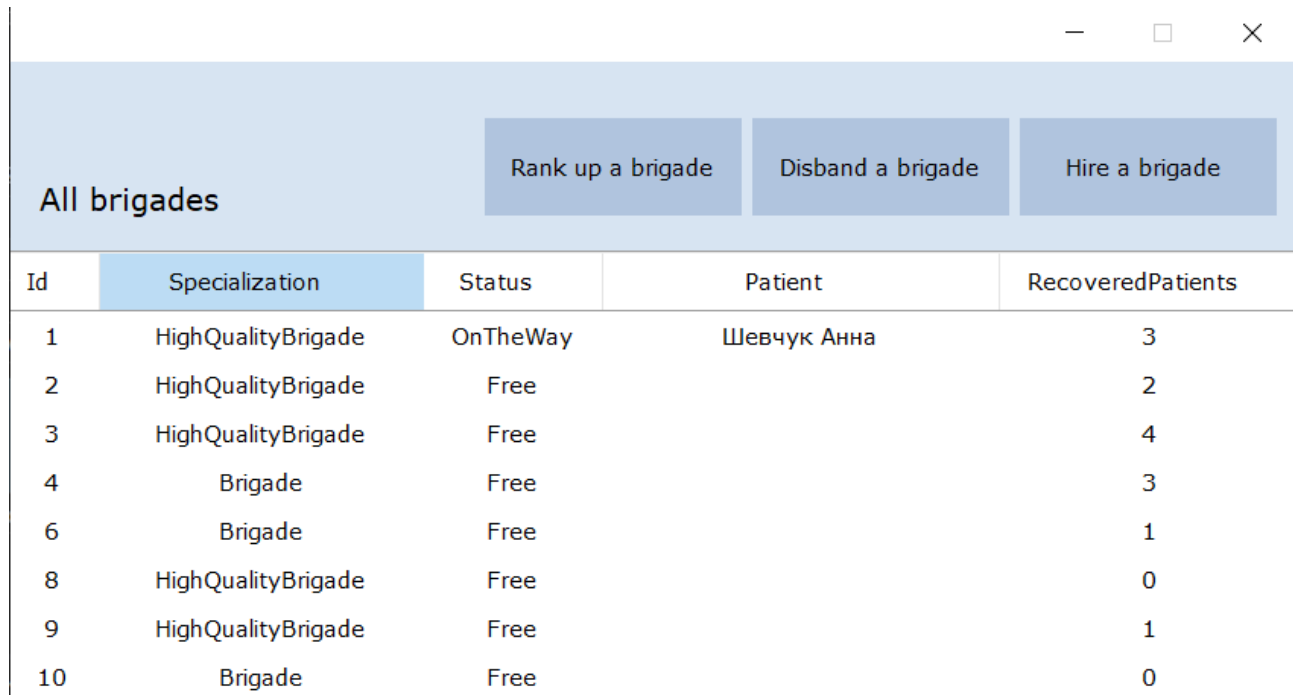
При відкритті форми ініціалізуються контролери бригад та пацієнтів, визначається джерело даних для таблиці, а також оформлюється підписка на подію SelectionChanged. Ця подія відповідає за виклик методу, що буде показувати або приховувати кнопку Set Brigade, при зміні виділення рядку таблиці. Також задається параметр CancellationTokenSource, який відповідає за призупинення потоку пацієнтів при закритті даної форми. Після відкриття форми запускається асинхронний метод, який починає генерацію пацієнтів через контролер. Лістинг методів класу зображений в додатку Б.11.

При натисненні на кнопку Set Brigade постає форма вибору бригади (рис. 3.4), на якій у вигляді таблиці зображений список вільних на цей момент бригад і кнопка призначення бригади на виклик, яка повертає до форми з пацієнтами сповістку про те, що бригада була призначена на виклик, відповідно, дані про пацієнта можна прибирати з таблиці.

Choose a free brigade:			To appoint
Id	Specialization	Status	RecoveredPatients
1	HighQualityBrigade	Free	3
2	HighQualityBrigade	Free	2
3	HighQualityBrigade	Free	4
8	HighQualityBrigade	Free	0
9	HighQualityBrigade	Free	1

Рис 3.4. Форма з вибором бригади для призначення на виклик

Остання форма (рис. 3.5.) GroupOfDoctors являє собою вікно контролю над бригадами. Вже згаданий вище клас DataGridView дозволив вивести з файлу список бригад у вигляді таблиці. Метод UpdateGrid задає параметри вигляду таблиці та пропонує створити першу бригаду, якщо вони взагалі відсутні.



All brigades				
Rank up a brigade Disband a brigade Hire a brigade				
Id	Specialization	Status	Patient	RecoveredPatients
1	HighQualityBrigade	OnTheWay	Шевчук Анна	3
2	HighQualityBrigade	Free		2
3	HighQualityBrigade	Free		4
4	Brigade	Free		3
6	Brigade	Free		1
8	HighQualityBrigade	Free		0
9	HighQualityBrigade	Free		1
10	Brigade	Free		0

Рис. 3.5. Вигляд форми контролю за бригадами

Над таблицею розташовані три кнопки, що дозволяють виконувати різні дії над бригадами. Кнопка Rank up викликає метод підвищення кваліфікації бригади, який перевіряє, чи є нинішня кваліфікація бригади звичайною, а також статус команди. Для підвищення бригада повинна бути вільною в цей момент, інакше виведуться відповідні MessageBox з помилкою виконання операції.

Кнопка Disband a brigade активується при вибраному рядку певної бригади в таблиці, при натисненні звертається до контролера бригад та видаляє її зі списку.

Кнопка Hire a brigade також звертається до контролера, але з протилежною причиною – створення і занесення до списку нової бригади.

Лістинги кнопок розташовані у додатку Б.12.

### 3.5. Тестування програми

Протестуємо програму (рис. 3.6 – рис. 3.14) у режимі роботи диспетчера станції ШМД: прийняти заявки пацієнтів, виконати різні дії над бригадами.

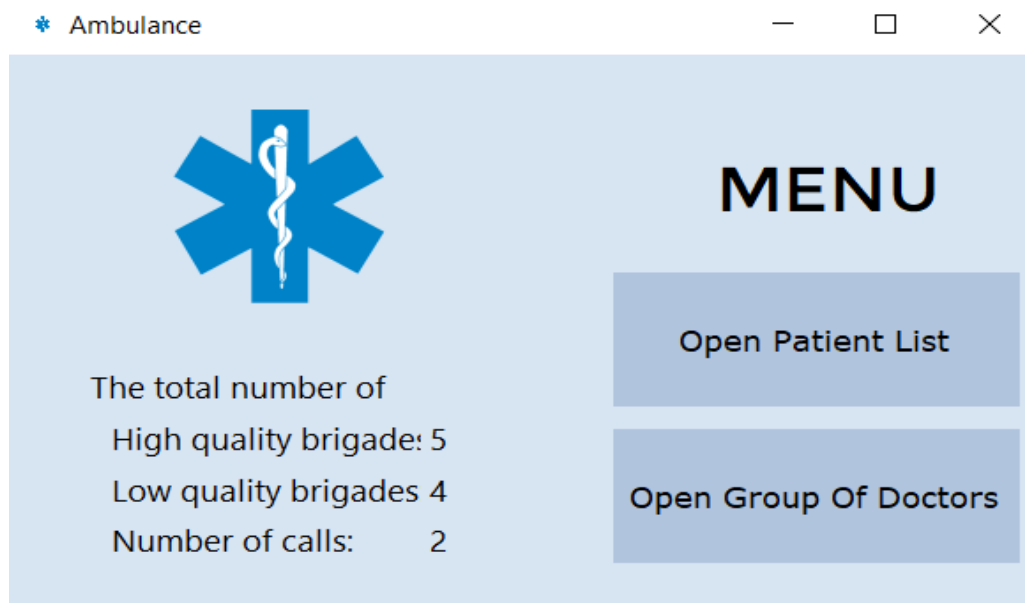


Рис. 3.6. З головного меню перехід до списку пацієнтів

List of receipts					Set Brigade
CallTime	FullName	Distance	Illness	Priority	
31.05 3:44:05	Ковальчук Василь	29	Опik	Low	
31.05 3:44:05	Шевчук Марія	36	Перелом	High	

Рис. 3.7. Вибір пацієнта та натиснення кнопки призначення бригади



Choose a free brigade:			To appoint
Id	Specialization	Status	RecoveredPatient
4	Brigade	Free	3
6	Brigade	Free	1
10	Brigade	Free	0
11	Brigade	Free	0

Рис. 3.8. Вибір будь-якої вільної бригади, натиснення кнопки призначення бригади

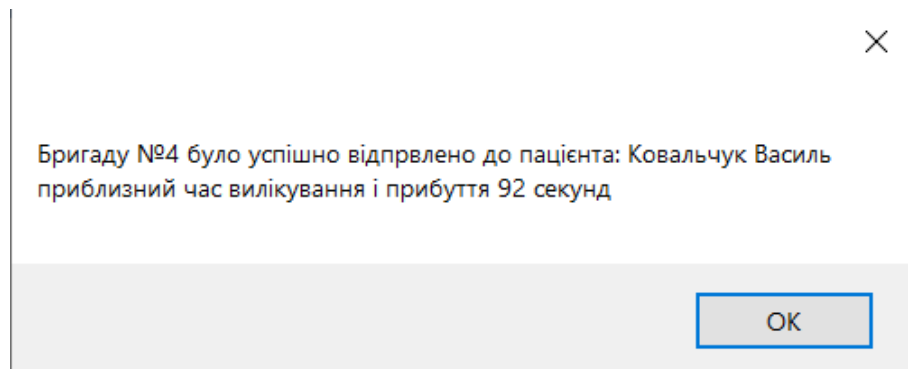


Рис. 3.9. Діалогове вікно підтвердження відправки бригади

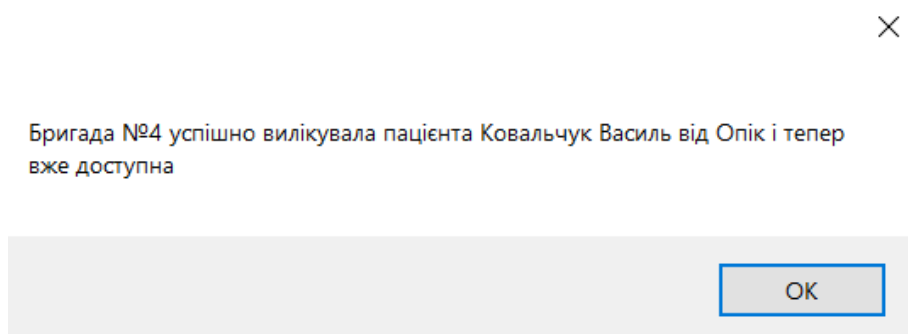


Рис. 3.10. Виведення діалогового вікна завершення процесу лікування

All brigades					Rank up a brigade	Disband a brigade	Hire a brigade
Id	Specialization	Status	Patient		RecoveredPatients		
1	HighQualityBrigade	Free			3		
2	HighQualityBrigade	Free			2		
3	HighQualityBrigade	Free			4		
4	Brigade	HealsThePati...	Ковальчук Василь		3		
6	Brigade	Free			1		
8	HighQualityBrigade	Free			0		
9	HighQualityBrigade	Free			1		
10	Brigade	Free			0		
11	Brigade	Free			0		

Рис. 3.11. Вибір бригади при натисненні на рядок

6	Brigade	Free
8	HighQualityBrigade	Free
9	HighQualityBrigade	Free
10	Brigade	Free
11	HighQualityBrigade	Free

Бригада була підвищена

OK

Рис. 3.12. Підвищення бригади з id 11

6	Brigade	Free
8	HighQualityBrigade	Free
9	HighQualityBrigade	Free
11	HighQualityBrigade	Free

Видалення бригади

Бригаду №10 розформовано

OK

Рис. 3.13. Розформування бригади з id 10

6	Brigade	Free
8	HighQualityBrigade	Free
9	HighQualityBrigade	Free
11	HighQualityBrigade	Free
12	Brigade	Free

Увага!

Нову бригаду створено

OK

Рис. 3.14. Створення нової бригади з id 12

### **3.6. Висновки стосовно реалізації**

У цьому розділі була описана реалізація контролерів і моделей продукту, проведений ретельний розбір реалізації графічного інтерфейсу з використанням Windows Forms, а також проведене тестування програми за сценарієм роботи диспетчера станції ШМД.

## Висновки

Після аналізу технічного завдання, були проведені пошуки інформації щодо поняття систем масового обслуговування та їх видів, був обраний підходящий варіант.

При проектуванні застосунку була розроблена ієрархія класів, яка відповідає схемі обраного патерну проектування (MVC). Наявність ієрархії класів полегшила створення як загальних, так і конкретних алгоритмів реалізації деяких методів.

При розробці курсового проекту було покращене розуміння принципів об'єктно-орієнтованого програмування, у результаті чого вони були успішно інтегровані до програмного продукту.

Придумані алгоритми були реалізовані з використанням мови C# та бібліотеки Windows Forms, що дозволило створити швидкий, зручний і зрозумілий MDI-додаток.

Продукт має місце для подальшого розвитку його можливостей. Варіанти розширень функціоналу:

- розширена кастомізація бригад;
- звільнення або призначення кожного члена бригади окремо;
- реалізація відмов пацієнтів від обслуговування;
- створення рейтингової системи для кожної з бригад та ін.

## Список використаної літератури

1. Стаття на тему «Швидка медична допомога». [Електронний ресурс] - Режим доступу: [https://uk.wikipedia.org/wiki/Швидка\\_медична\\_допомога](https://uk.wikipedia.org/wiki/Швидка_медична_допомога) Дата доступу 22.05.2021
2. Стаття на тему «Теорія масового обслуговування». [Електронний ресурс] - Режим доступу: [https://uk.wikipedia.org/wiki/Теорія\\_масового\\_обслуговування](https://uk.wikipedia.org/wiki/Теорія_масового_обслуговування) Дата доступу 22.05.2021
3. Лекція на тему «Основні поняття теорії масового обслуговування». - [Електронний ресурс] Режим доступу: [https://er.nau.edu.ua/bitstream/NAU/21000/23/Лекція\\_CA\\_12.pdf](https://er.nau.edu.ua/bitstream/NAU/21000/23/Лекція_CA_12.pdf) Дата доступу 23.05.2021
4. Стаття на тему «Model-View- Controller» . - [Електронний ресурс] Режим доступу: <https://ru.wikipedia.org/wiki/Model-View-Controller> Дата доступу 24.05.2021
5. Супруненко О. О. Методичні вказівки до виконання та оформлення курсової роботи з дисципліни «Об'єктно-орієнтоване програмування» для студентів, які навчаються за напрямом підготовки 050101 – «Комп'ютерні науки», 050103 – «Програмна інженерія» та 040303 – «Системний аналіз» усіх форм навчання. / О. О. Супруненко, Ю. Є. Гребенович. – Черкаси: ЧНУ імені Богдана Хмельницького, 2013. – 40 с.