НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕЇНІКИ

ЛАБОРАТОРНА РОБОТА 4

з дисципліни:

«Сучасні методології і технології розробки програмного забезпечення»

на тему:

«Багатошарова архітектура програмних додатків. Піраміда тестування. Інтеграційні тести.»

Студента 3 курсу групи ІТ-81

Венделовського Івана Сергійовича

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

Викладач:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_к.т.н. Штифурак Юрій Михайлович

(оцінка)

Київ – 2021

ЗМІСТ

[1 Постанока задачі 3](#_Toc67166363)

[1 Покриття змістовної логіки модульними тестами 4](#_Toc67166364)

[1.1 Реалізація тестів для класу «BillSevice» 4](#_Toc67166365)

[1.1.1 Реалізація тестів методу «InitializeBill» 4](#_Toc67166366)

[1.1.2 Реалізація тестів методу «GetBillsToPayByUserName» 6](#_Toc67166367)

[1.1.3 Реалізація тестів методу «PayForDelivery» 7](#_Toc67166368)

[1.2 Реалізація тестів для класу «DeliveryService» 8](#_Toc67166369)

[1.2.1 Реалізація тестів методу «GetDeliveryInfoToGet» 8](#_Toc67166370)

[1.2.2 Реалізація тестів методу «ConfirmGettingDelivery» 9](#_Toc67166371)

[1.2.3 Реалізація тестів методу «GetDeliveryCostAndTimeDto» 9](#_Toc67166372)

[1.3 Реалізація тестів для класу «UserService» **Ошибка! Закладка не определена.**](#_Toc67166373)

[1.3.1 Реалізація тестів методу «FindByName» **Ошибка! Закладка не определена.**](#_Toc67166374)

[1.3.2 Реалізація тестів методу «ReplenishAccountBalance» **Ошибка! Закладка не определена.**](#_Toc67166375)

[Висновки 11](#_Toc67166376)

[Список використаних джерел 12](#_Toc67166377)

# 1 Постанока задачі

Мета: Розуміти суть піраміди тестування. Навчитися реалізовувати інтеграційні тести.

Завдання:

1. За відсутності механізмів впровадження інтеграційних тестів (наприклад, зазорів) провести відповідний рефакторинг коду.

2. Реалізувати інтеграційні тести.

3. За умови наявності складної логіки, що не можливо покрити автономними тестами, провести рефакторинг коду, який забезпечить можливість написання unit-тестів для змістовної логіки розробленого ПЗ.

# 1 Покриття змістовної логіки модульними тестами

## Реалізація тестів для класу «BillSevice»



Рисунок 1.1 – Тестовий клас, необхідні фікстури та їх ініціалізація.

### 1.1.1 Реалізація тестів методу «InitializeBill»



Рисунок 1.2 – Перевірка коректності створення рахунку при коректних вхідних даних.



Рисунок 1.3 – Перевірка виникнення «NoSuchUserException» виключення у випадку якщо не існує користувача отриманого із вхідних даних.



Рисунок 1.4 – Перевірка виникнення «NoSuchWayException» виключення у випадку якщо не існує маршруту отриманого із вхідних даних.

### 1.1.2 Реалізація тестів методу «GetBillsToPayByUserName»



Рисунок 1.5 – Перевірка коректності отримання платежів за ім’ям користувача при коректних вхідних даних.

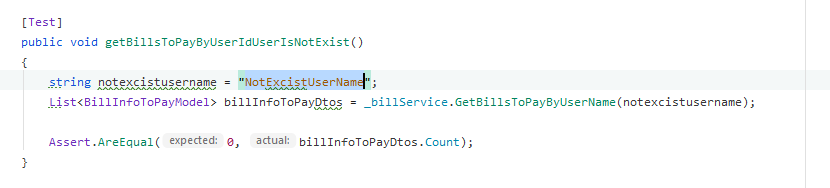


Рисунок 1.6 – Перевірка відсутності платежів у випадку якщо не існує зданого користувача.

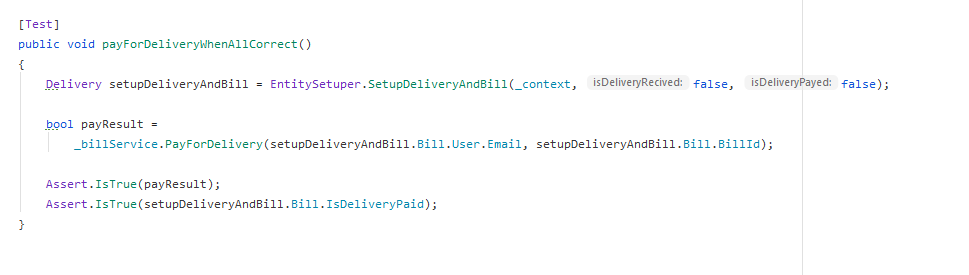


Рисунок 1.7 – Перевірка виникнення «NoSuchWayException» виключення у випадку якщо не існує маршруту отриманого із вхідних даних.

### 1.1.3 Реалізація тестів методу «PayForDelivery»

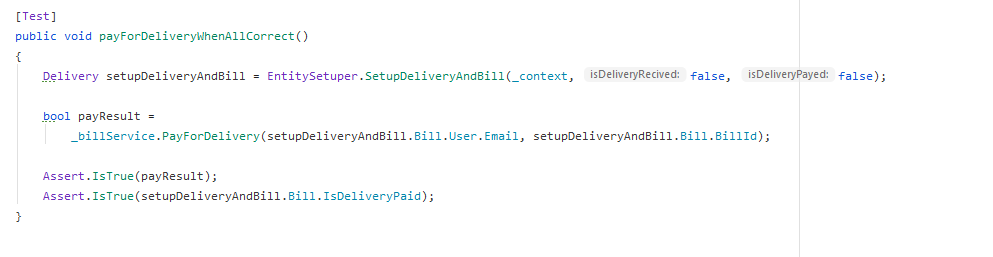


Рисунок 1.8 – Перевірка коректності оплати при коректних вхідних даних.

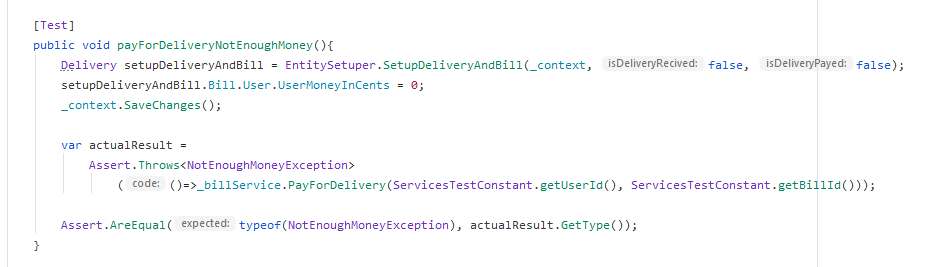


Рисунок 1.9 – Перевірка виникнення «NotEnoughMoneyException» виключення у випадку якщо у користувача не достатньо коштів для виконання оплати.

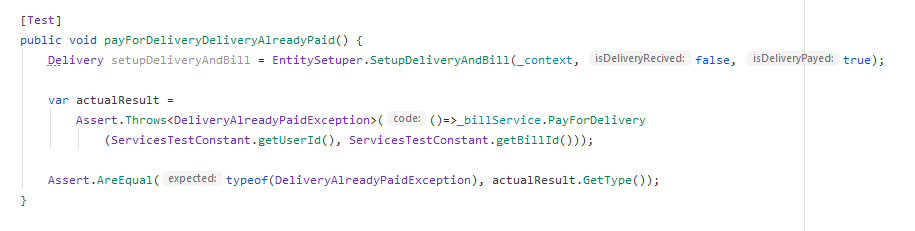


Рисунок 1.10 – Перевірка виникнення «DeliveryAlreadyPaidException» виключення у випадку якщо рахунок вже оплачено.

## Реалізація тестів для класу «DeliveryService»



Рисунок 1.11 – Тестовий клас, необхідні фікстури та їх ініціалізація.

### 1.2.1 Реалізація тестів методу «GetDeliveryInfoToGet»



Рисунок 1.12 – Перевірка коректності списку доставок на отримання при коректних вхідних даних.

### 1.2.2 Реалізація тестів методу «ConfirmGettingDelivery»



Рисунок 1.13 – Перевірка коректності підтвердження отримання доставки при коректних вхідних даних.



Рисунок 1.14 – Перевірка виникнення виключення «AskedDataIsNotExist» у випадку якщо заданої доставки не існує.

### 1.2.3 Реалізація тестів методу «GetDeliveryCostAndTimeDto»

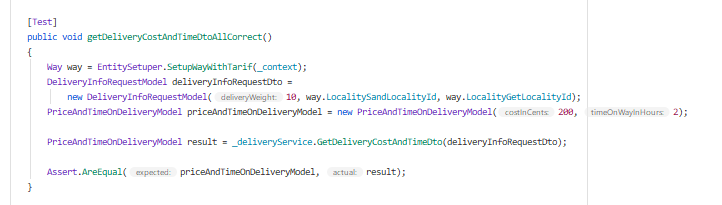


Рисунок 1.15 – Перевірка коректності отримання даних про час та вартість доставки при коректних вхідних даних.



Рисунок 1.16 – Перевірка виникнення «NoSuchWayException» виключення у випадку якщо заданого маршруту не існує.

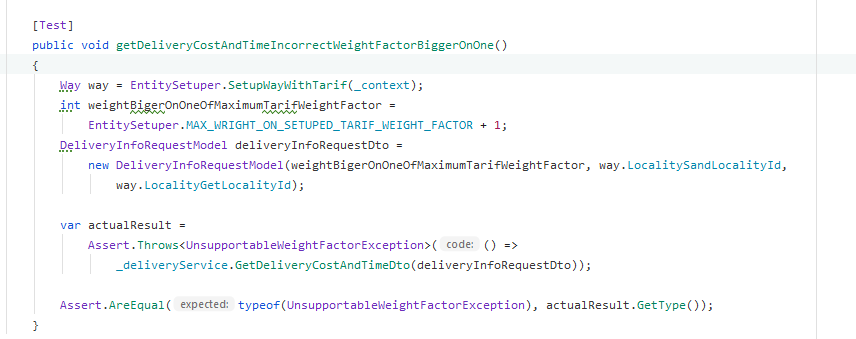


Рисунок 1.17 – Перевірка виникнення «UnsupportableWeightFactorException» виключення у випадку вантаж занадто важки для заданого маршруту.

# Висновки

В дані роботі було протестовано логіку додатку з допомогою інтеграційних тестів. Додаток було спроектовано коректно, з повним дотриманням принципів солід та модульності це дозволило уникнути необхідності в рефакторингу коду перед та у процесі тестування.

В звіті було наведено тести ключових методів що реалізують основний функціонал додатку. Всі тести можна знайти у репозиторії проекту за адресою «https://github.com/VINIPOOH/delivery\_dot\_net».

# Список використаних джерел

1. Ноубл, Дж., Андерсон, Т., Брэйтуэйт, Г., Казарио, М., Третола, Р. Flex 4. Рецепты программирования. — БХВ-Петербург, 2011. — С. 548. — 720 с
2. Самоучитель UML 2. – СПб.: БХВ-Петербург, 2007. – 567 с.: ил. ISBN 978-5-94157-878-8
3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. П75 Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 368 с.: ил. (Серия «Библиотека программиста») ISBN 5-272-00355-1
4. Мартин Фаулер., Чистий код: создание, анализ и рефакторинг. – СПб.: Питер, 2019. – 464 с.: ил.