МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІФОРМАЦІЙНОЇ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**КУРСОВА РОБОТА**

# з дисципліни «Програмування - 2. Структури даних та алгоритми»

Варіант №12

|  |  |  |
| --- | --- | --- |
| **Керівник :**  доц. каф. АСОІУ,  к.т.н. Вітковська І.I |  | **Виконав:**  студент гр. ІС-02, АСОІУ  Лієнко Д.К  залікова книжка № ІС-0212 |
| Допущено до захисту |  |
| І\_\_\_І \_\_\_\_\_\_\_\_\_ 2021 \_\_\_\_\_\_\_\_\_  підпис |  |
| Захищено з оцінкою  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_  оцінка підпис  І\_\_\_І \_\_\_\_\_\_\_\_\_ 2021 |  |

Київ-2021

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Факультет | ***Інформаційної та обчислювальної техніки*** | | | | |
|  | (назва факультету, інституту) | | | | |
| Кафедра | **Кафедра автоматизованих систем обробки інформації та управління** | | | | |
|  | (назва кафедри) | | | | |
| Дисципліна | ***Програмування - 2. Структури даних та алгоритми*** | | | | |
|  | (назва) | | | | |
| Курс | ***1*** | Група | ***ІС-02*** | Семестр | ***2*** |

**ЗАВДАННЯ**

**на курсовий проект (роботу) студента**

|  |
| --- |
| ***Лієнко Дениса Костянтиновича*** |
| (прізвище, ім’я, по батькові) |

.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1. Тема роботи: | | ***Розробка програмного забезпечення з використанням*** | | | |
| ***об’єктно-орієнтованого підходу.*** | | | | | |
|  | | | | | |
|  | |  | | | |
| 2. Строк здачі студентом закінченого проекту (роботи) | | | | |  |
|  | |  | | | |
| 3. Вихідні дані до проекту (роботи): | | | | ***Варіант №12*** | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | |  | | | |
| 4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають | | | | | |
| розробці): |  | | | | |
|  | | | | | |
|  | |  | | | |
| 5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): | | | | | |
| ***діаграми класів, послідовності та об’єктів в нотації UML*** | | | | | |
|  | | | | | |
|  | |  | | | |
| 6. Дата видачі завдання: | | |  | | |

Зміст

Оглавление

**РОЗДІЛ 1 аналіз предметної області3**

опис предметної області 3  
сценарій роботи програми 3

окремі сценарії роботи.формування тз 3

**РОЗДІЛ 2. кодування8**

опис вибору мови програмування 8

**РОЗДІЛ 3 опис програмного забезпечення 9**

3.1 Опис структури програми 9

3.1.1 Структура бібліотек 9

3.1.2 Взаємодія бібліотек 9

3.2 Опис класів та їх взаємодії 10

**РОЗДІЛ 4 тестування** 16

4.1 План тестування програмної системи 16

4.2 Тестування бізнес логіки 16

4.3 Приклади тестів 16

**Висновок** 24

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 25

**ДОДАТКОВІ ПОСИЛАННЯ** 25

ВСТУП

Призначення КР: Вивчити типи відношень між класами в ООП.

Ціль роботи: Спроектувати об’єктно-орієнтовану модель предметної галузі згідно з варіантом, визначивши необхідні для цього класи та їх структуру. При об’єктно-орієнтованому проектуванні продемонструвати застосування всіх видів відношень. Написати програми мовами С++ та С#, у яких реалізувати попередньо спроектовану об’єктно-орієнтовану модель. Предметну галузь реалізувати окремим проектом – динамічною бібліотекою (С#) чи окремими класами з відсутнім в них введенням-виведенням (С++). Інтерфейс користувача, наприклад, введення\виведення з консолі, реалізовувати окремим проектом (С#) чи класами з операціями введення\виведення (С++). Код інтерфейсу має бути простим (демонструється використання класів предметної галузі шляхом створення об’єктів та їх застосування, вимагається лише нескладна перевірка коректності вводу, введення з консолі мінімальне).

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області  
В клубі надається інформація про країни, що цікаві для туризму, тематики турів, туристичні тури по певних місцях, туристичні агенції, що надають послуги таких турів. Користувач сервісу клубу може обрати собі тур та агенцію згідно його вподобань та запланованої дати подорожі.

1.2 Сценарій роботи програми.  
Аналізуючи опис предметної області, можна так інтерпретувати поставлену задачу:

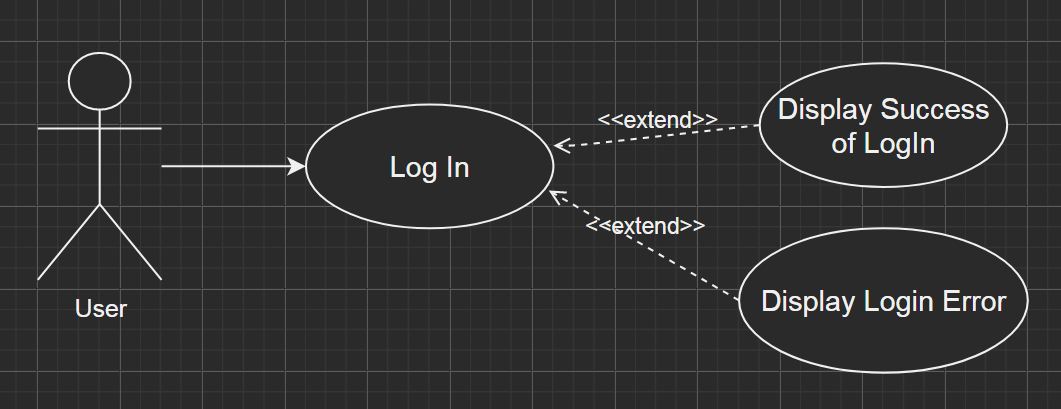
Актори: користувач

Дії користувача:

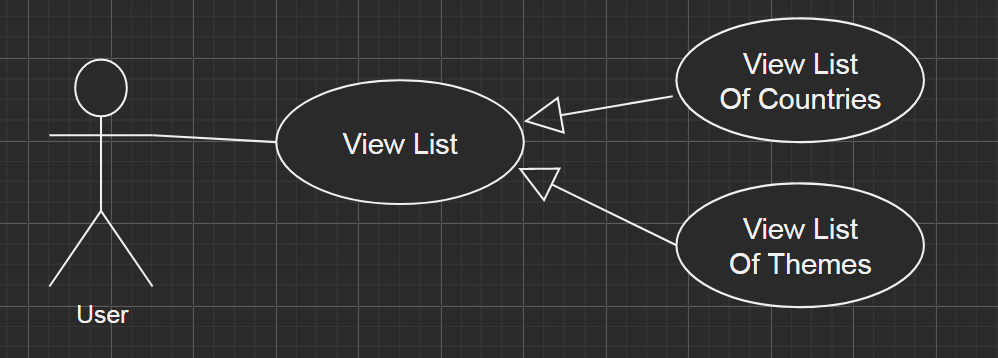
1. Отримати інформацію про доступні країни, тематики.
2. Поняття «цікаві тури» слід ідентифікувати, як суб’єктивні вподобання користувача, які він буде вводити на обробку в програми. Тому виокремимо таку дію, як введення вподобань.
3. Отримання інформації про тури, що підходять вподобанням користувача.
4. Замовлення користувачем туру, що надається певною агенцією

1.3 Окремі сценарії взаємодії користувача із програмою.

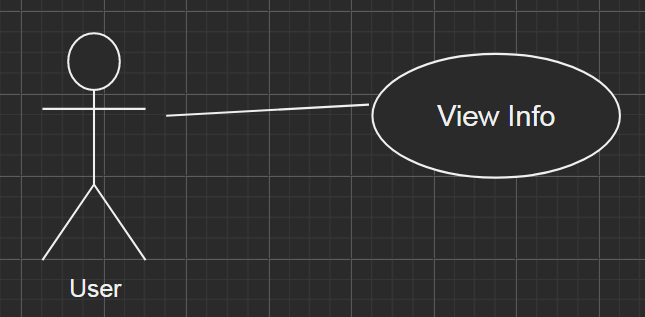
Зображення окремих сценаріїв взаємодії Use-case нотаціїєю:

1) Авторизація. Користувач вводить свій ідентифікатор(email, телефон або унікальне ім’я) та пароль. Авторизація розширюється 2 підподіями: отримання інформації про успішну авторизацію або інформацію про помилку вводу пароля або ім’я.   


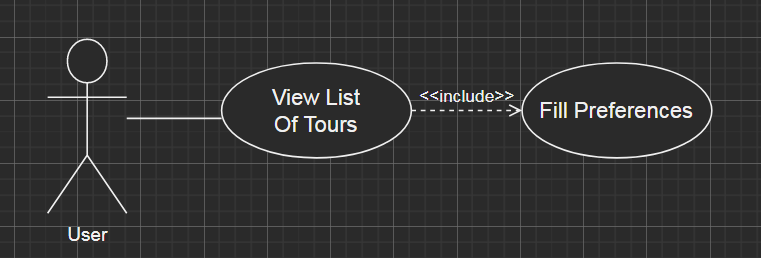
2) Отримання інформації про доступні доступні країни, тематики. Користувач отримує інформацію про доступні доступні країни або тематики як списки. Сценарії «подивитися список країни» або «подивитися список турів» наслідуються від базового сценарію «подивитися список»



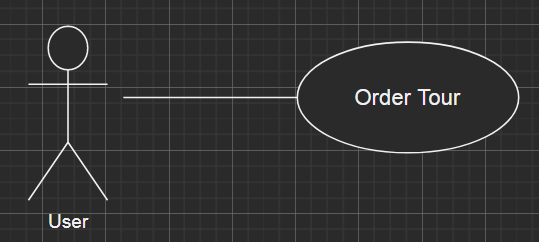
3) Ознайомившись зі списками доступних варіантів країн або тематик, користувач може отримати інформацію про конкретну опцію.



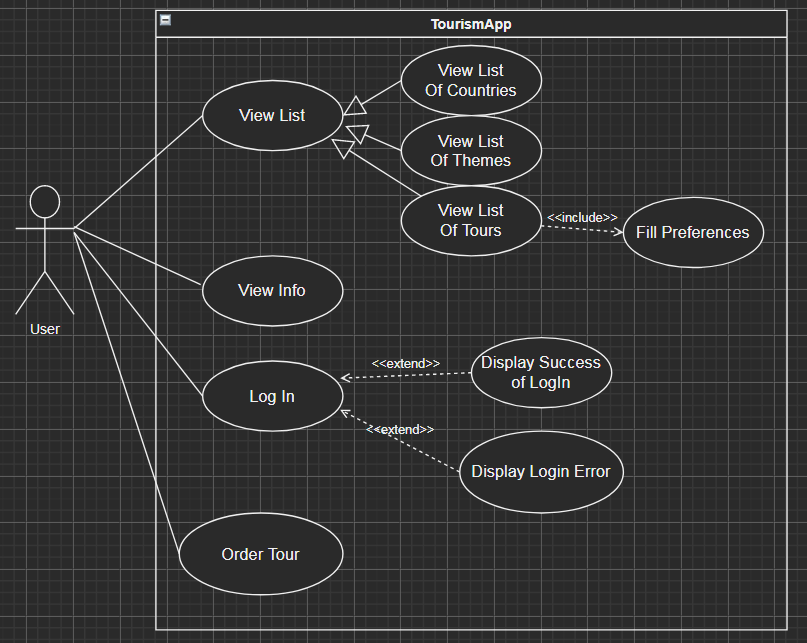
4) Оскільки перегляд турів, що відповідають вподобанням користувача, вимагає вказання вподобань користувачем, то сценарій «перегляд турів» включає в себе сценарій «заповнення вподобань».



5) Ознайомившись із турами, користувач може обрати конкретний і замовити його



Зкомбінувавший окремі сценарії, створимо повноцінну Use-case діаграму:



Розбивши вхідну задачу на конкретні під задачі, а також зобразивши механізм взаємодії програми з користувачем у вигяді UML Use-Case діаграми, можна конкретизувати зміст питання та сформувати вимоги до програмного продукту.

Програма повинна містити дані про

* доступні країни
* тури
* тематики турів
* туристичні агенції
* користувачів
* замовлення, що робляться користувачами

Цими даними є сенс користуватися у вигляді списків. Їх легко заповнювати, змінювати. Також буде сенс звертатися до окремих об’єктів данних через індекси.

Для зберігання даних скористаємося файлами формату .txt. Списки можна легко створювати, читаючи по рядкам поля .txt файлу, де кожне поле буде представляти собою конкретний об’єкт класу.

Програма буде надавати користувачу функціонал доступу до цих даних. Наприклад для перегляду доступних країн. Або буде давати доступ до перегляду даних дозовано. Наприклад показуючи лише ті тури, що відповідають вподобанням користувача.

Вищеперерахованому опису підходить архітектура MVC (Model View Control).



Зображення взято з Wikipedia:   
https://ru.wikipedia.org/wiki/Model-View-Controller

Де View буде представлено у вигляді консольного додатку. Controller – у вигляді бібліотеки, що буде містити головний однойменний клас, а також класи доступу до txt файлів. Клас Controller при ініціалізації буде завантажувати ресурси даних із файлів у окремі змінні, ці змінні будуть підпорядковуватися бізнес логіці. Бізнес логіку (тобто реалізацію предметного домену засобами ООП) винесемо в окрему бібліотеку. Model являє собою txt файли з даними.

РОЗДІЛ 2

КОДУВАННЯ

2.1 Опис вибору мови програмування

Для виконання даної курсової роботи була обрана мова С# на платформі .NET Core 3.1.

Вибір мови зумовлений її відносною простотою (відносно С++). В основному її зручність полягає у автоматизованому очищенні пам’яті, у той час, як розробка на мові С++ вимагає від програміста слідкування за звільненням ресурсів.

Але робота з пам’яттю в С# не повністю автоматична – програміст повинен слідкувати за пам’яттю, що виділяється під об’єкти некерованих ресурсів: з’єднань з файлами, файловими дескрипторами, базами даних, а також потоками і т.д.

Окрім цього, в мові С# присутні зручні синтаксичні конструкції. Наприклад: зрізи масивів або колекцій, блок using, в якому проводяться операції над некерованими об’єктами. Якщо даний об’єкт реалізує інтерфейс IDisposable, то після виходу із блоку для нього виконається метод Dispose – аналог деструктора. Також слід зазначити про бібліотеку LinQ, яка сильно спрощує роботу з колекціями.

Платформа .NET Core 3.1. була обрана для можливості розробки на комп’ютерах з різними операційними системами (Linux та Windows). Як відомо, програми на базі .NET Core виконуються на всіх найпопулярніших ОС: Windows, Linux, IOS. У той час, як .NET використовується тільки на Windows-машинах.

РОЗДІЛ 3

ОПИС СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Опис структури рішення

3.1.1 Структура проектів та бібліотек

Як було описано раніше, компоненти програмного рішення (проекти та бібліотеки) будуть побудовані за принципом MVC. І будуть представлені у вигляді консольного додатку, контролера, бібліотеки класів бізнес логіки та файлів, що міститимуть дані.

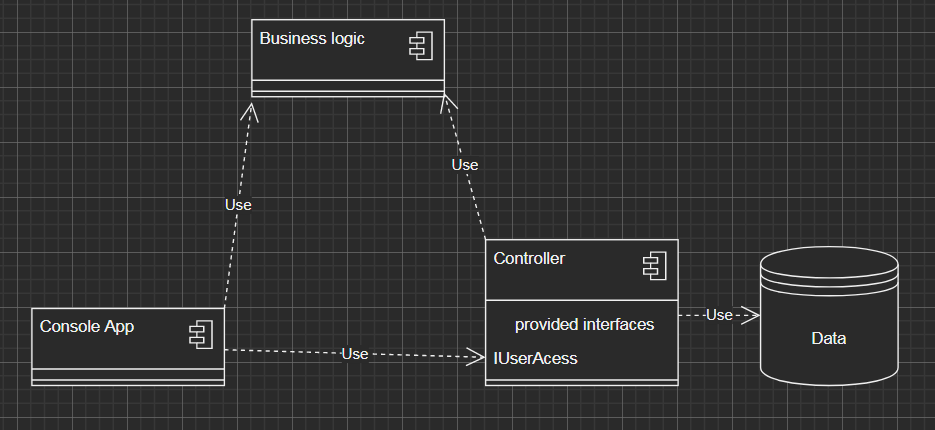
3.1.2 Взаємодія бібліотек

Проект консольного додатку буде використовувати класи бізнес логіки

Та реалізувати методи виведення на екран об’єктів цих класів. У той же час він буде використовувати контролер для доступу до ресурсів. Тому доцільно створити інтерфейс користувацького доступу, який буде реалізовувати клас Controller.

Клас Controller у свою чергу теж використовує класи бізнес логіки, але для керування та абстракції даних, які він отримує із файлів.

При першому наближені система має наступний вигляд:



3.2 Опис класів та їх взаємодії

3.1.1 Опис класів бізнес логіки

1. Class Person.  
   Клас створено як абстракцію. При масштабуванні проекту, можуть з’явитися нові класи, окрім користувача, що будуть інкапсулювати людину. Окрім того, в класах-нащадках успадкуються властивості класу Person, що дозволить наглядніше бачити специфіку нащадка.

|  |  |  |
| --- | --- | --- |
| Властивості | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | string | Name |
| public | string | Surname |
| public | string | Email |
| public | string | Phone |

1. Class User  
   Клас користувача, який безпосередньо «контактує» з іншими класами: заповнює вподобання, створює замовлення.   
   Має унікальний ідентифікатор, а також пароль.   
   Містить список ідентифікаторів замовлень, зроблених ним.   
   Метод IsPasswordCorrect() приймає значення введеного з консолі паролю та порівнює із значенням поля password. Метод приймає в якості аргументів об’єкт класу тур, кількість туристів, за якими бронюється тур та опис, залишений користувачем під час бронювання туру. Властивість OrdersID безпечно інкапсулює поле \_ordersID, запобігаючи тим самим доступ за посиланням до поля класу.

|  |  |  |
| --- | --- | --- |
| Властивості | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | string | Password |
| public | int | UserID |

|  |  |  |
| --- | --- | --- |
| Методи | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | bool | IsPasswordCorrect() |
| public | void | OrderTour |

1. Class Order  
   Клас, що описує заказ – бронювання туру. Містить властивості   
   OrderID – ідентифікатор заказу, Tour – тур замовлення, OrderOwner –користувач, що зробив замовлення, Desription - опис, залишений користувачем, NumOfTourists – кількість туристів, за якими заброньовано тур, TotalPrice – остаточна ціна (підраховуються при ініціалізації об’єкта при зчитуванні з файлу)

|  |  |  |
| --- | --- | --- |
| Властивості | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | int | OrderID |
| public | Tour | Tour |
| public | User | OrderOwner |
| public | string | Description |
| public | int | NumOfTourists |
| public | Float | TotalPrice |

1. Class Tour  
   Agency – агенція, яка надає послуги туру.  
   StartTime – дата та час початку туру   
   EndTime – дата та час закінчення туру

|  |  |  |
| --- | --- | --- |
| Властивості | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | int | TourID |
| public | int | Agency |
| public | string | Name |
| public | TourAgency | Agency |
| public | Theme | TourTheme |
| public | DateTime | StartTime |
| public | DateTime | EndTime |
| public | float | PricePerPerson |

1. Class TourAgency  
   Туристичне агентство

|  |  |  |
| --- | --- | --- |
| Властивості | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | int | AgencyID |
| public | string | Name |
| public | string | Email |
| public | string | WebPage |

1. Interface ITourFilter  
   Інтерфейс, що реалізується класами Theme, Preferences, Country Запроваджує для цих класів спільний метод – IsTourMatches(). Метод повертає true якщо об’єкт туру має спільні з ними властивості, false якщо різні . Інакше кажучи – класи, що реалізують інтерфейс можуть використовуватися як фільтр під пошуку турів за певною характеристикою.
2. Class Theme   
   Тематика тура  
   Name – назва теми, Description - опис теми, Equals – перевизначений від object метод, повертає true якщо рівні і назва, і опис тем. GetHashCode() Є необхідним для перевизначеня, коли перевизначається і IsEqual () , IsTourMatches(Tour tour) – повертає true, якщо тур має таку ж саму тематику.

|  |  |  |
| --- | --- | --- |
| Властивості | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | string | Name |
| public | string | Description |

|  |  |  |
| --- | --- | --- |
| Методи | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | bool | Equals |
| public | bool | IsTourMatches |

1. Class Preferences  
   Вподобання користувача.  
   СountryOfTour - бажана країна, Theme – бажана тематика туру, ArrivalTimeFrom – мінімальна бажана дата туру, ArrivalTimeTo - максимальна, PriceFrom – мінімальна бажана ціна, PriceTo – максимальна бажана ціна.

|  |  |  |
| --- | --- | --- |
| Властивості | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | Country | CountryOfTour |
| public | Theme | Theme |
| public | DateTime | ArrivalTimeFrom |
| public | DateTime | ArrivalTimeTo |
| public | PriceFrom | PriceFrom |
| public | PriceTo | PriceTo |

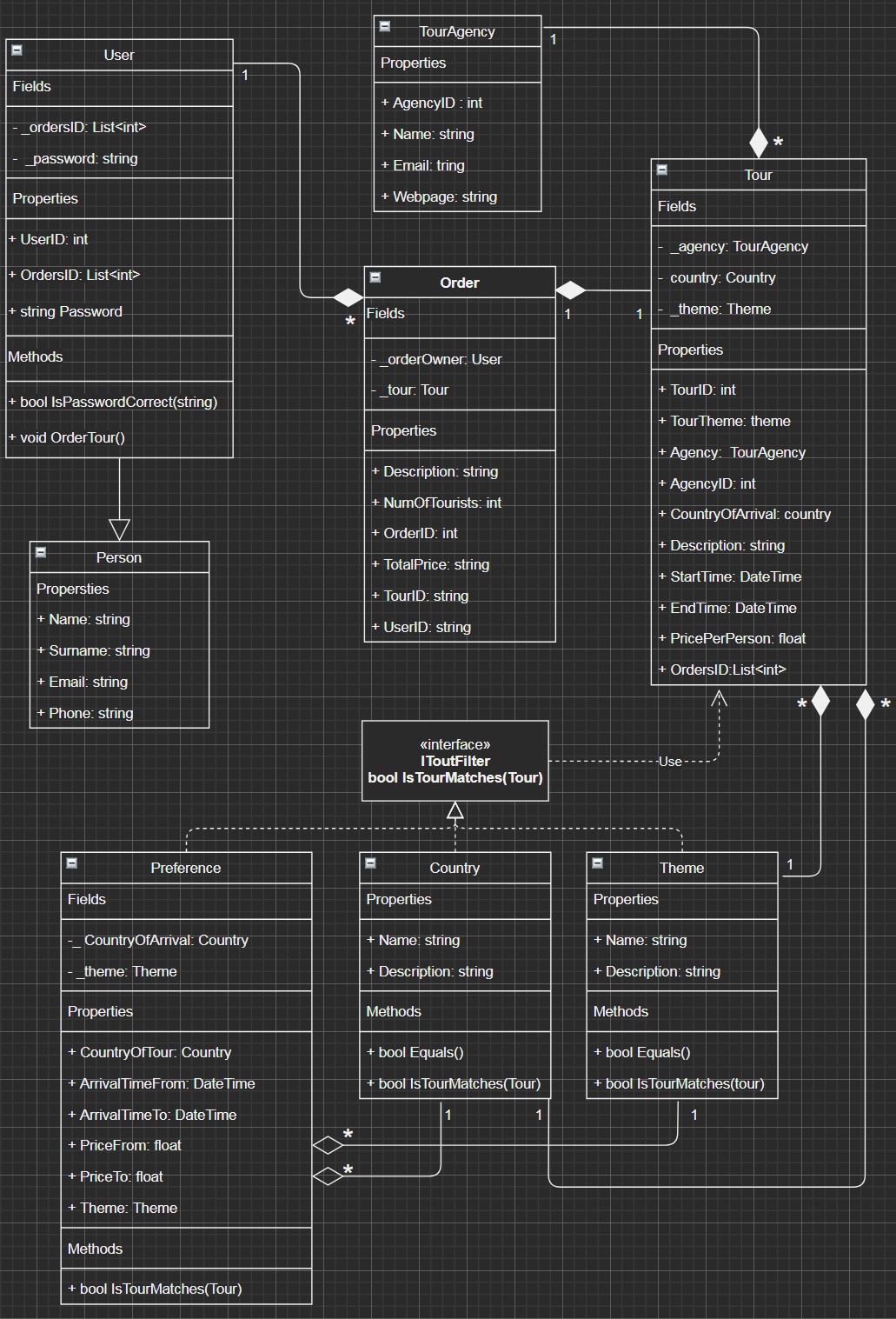
|  |  |  |
| --- | --- | --- |
| Методи | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | bool | IsTourMatches(Tour tour) |

1. Class Country  
   Тематика тура  
   Name – назва країни, Description - опис країни, Equals – перевизначений від object метод, повертає true якщо рівні і назва, і опис країни. GetHashCode() Є необхідним для перевизначеня, коли перевизначається і IsEqual () , IsTourMatches(Tour tour) – повертає true, якщо тур проводиться в тій ж країні.

|  |  |  |
| --- | --- | --- |
| Властивості | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | String | Name |
| public | string | Description |

|  |  |  |
| --- | --- | --- |
| Методи | | |
| **Модифікатор доступу** | **тип** | **Назва** |
| public | bool | IsTourMatches(Tour tour) |

Проаналізувавши класи, створимо UML-діаграму класів



Опис зв’язків між компонентами діаграми:

1. Користувач - Людина(User-Person). Тип відношення - наслідування.
2. Користувач - Замовлення(User-Order). Копмпозиція. Клас замовлення інкапсулює користувача, який і створив його. Чисельне відношення: 1..\* (1 користувач може мати багато замовлень, а замовлення – тільки одного користувача)
3. Замовлення – Тур(Order - Tour). Композиція. Замовлення повинно обов’язково відноситися до туру.
4. Тур агенція – Тур(TourAgency - Tour). Композиція. Тур агенція надає тури. Чисельне відношення: 1..\*.
5. Тематика – Тур(Theme - Tour). Композиція .Тур має певну тематику, але одну і туж саму тематику можуть мати різні тури. Відношення \*..1
6. Країна – Тур(Country - Tour). Композиція. Кожна країна може мати декілька турів,
7. Вподобання – Країна(Preferences - Country). Агрегація.
8. Вподобання - Тематика(Preferences - Theme). Агрегація
9. Ітнерфейс ITourFilter – (Вподобання, країна, тематика). Реалізація
10. Інтерфейс ItourFilter – Тур. Асоціація(використання)

РОЗДІЛ 4

ТЕСТУВАННЯ

4.1 План тестування

1. Тестування методів бізнес логіки (Unit-тести)
2. Тестування інтерфейсу  
   2.1 Протестувати введення даних при авторизації  
   1. Протестувати всі варіанти переміщення по інтерфейсу програми(з коректними значеннями вибору опцій та введення додаткової інформації)
   2. Протестувати неправильне введення даних на запитах додаткової інформації(включаючи заповнення форм)

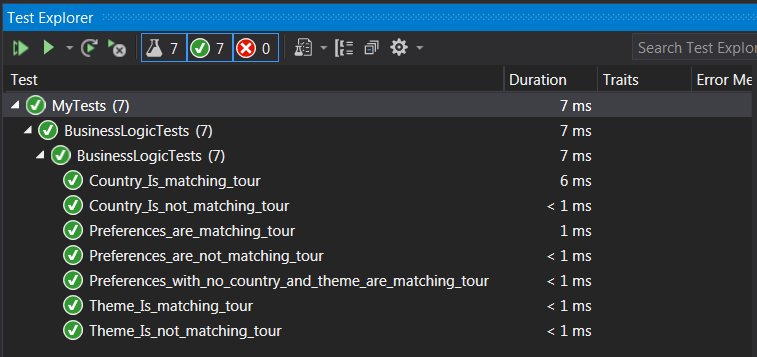
4.2 Приклади тестів

**4.2.1 Тестування методів бізнес логіки (Unit-тести)**

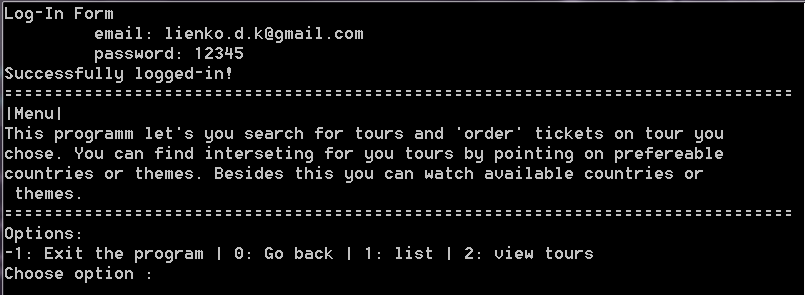
Тестування методу .IsTourMatches (див. MyTests -> BusinessLogicTests.cs)

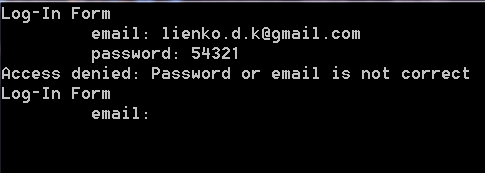
Тестування буде проводитися для реалізації методу в класах Preferences, Country, Theme в наступних умовах: тур підходить або не підходить: країні або тематиці, вподобанням(за умови, коли поле Theme Country дорівнюють або не дорівнюють null).

Результати тестувань:

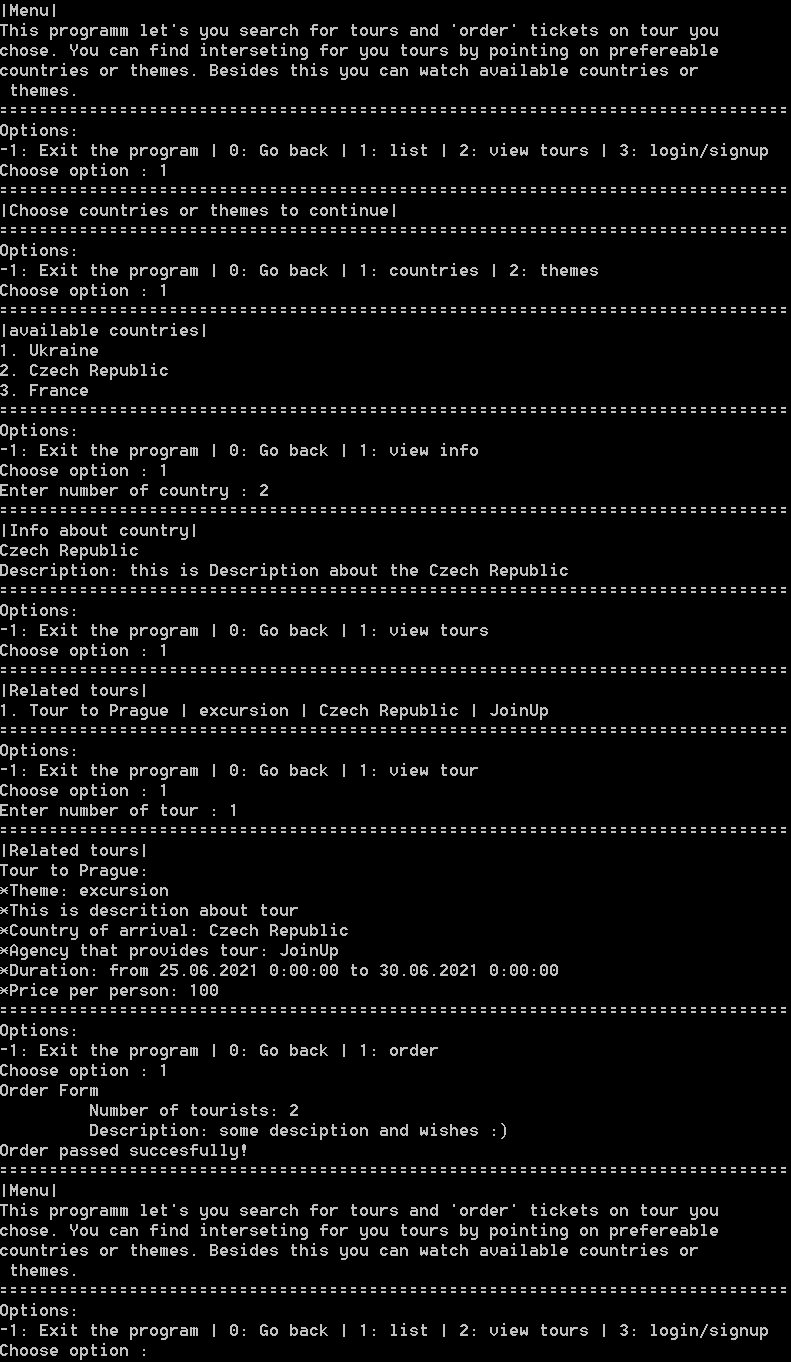
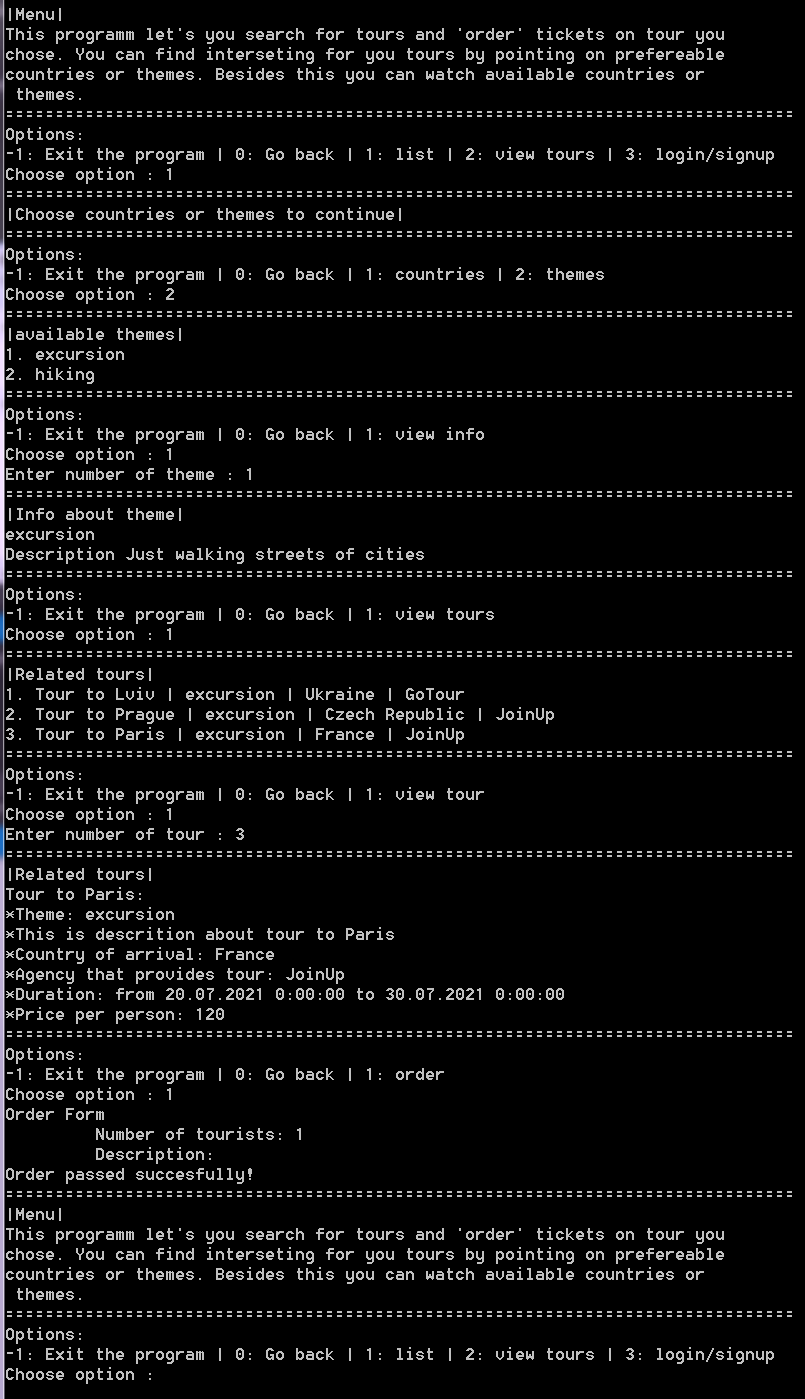


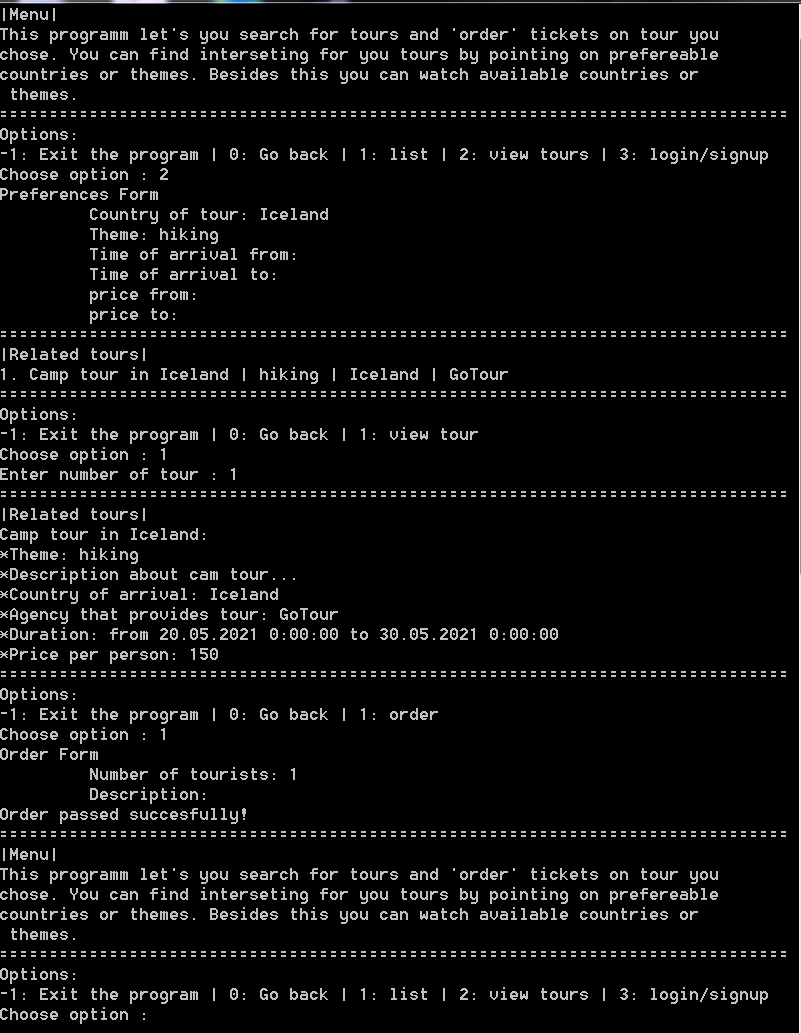
**4.2.1 Правильне і неправильне введення даних при авторизаці**ї





**4.2.2 всі варіанти переміщення по інтерфейсу програми**

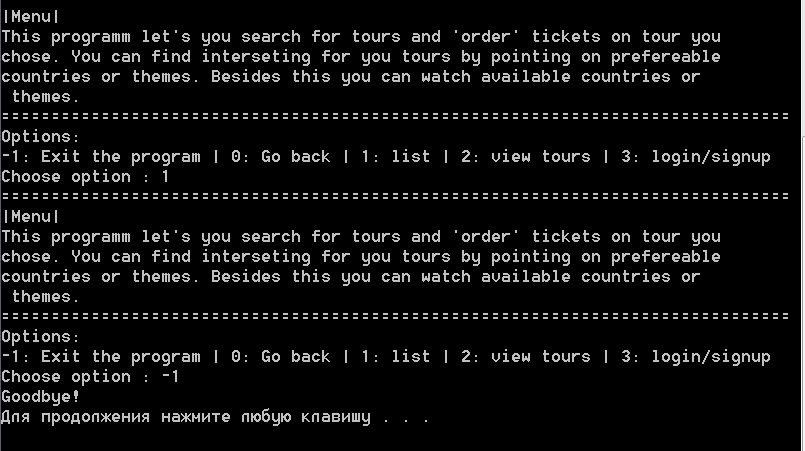
1. Показати список –> список країн –> подивитися тури по країні–>вибрати тур –> замовити тур –> автоматичне повернення до меню  
   
2. Схожий сценарій, але з переглядом доступних тематик  
   
3. Замовити тур –> заповнити вподобання –> перегляд турів



1. Повернення назад за допомогою опції 0.

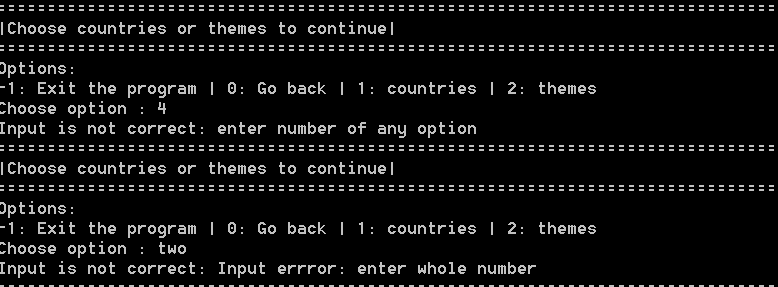


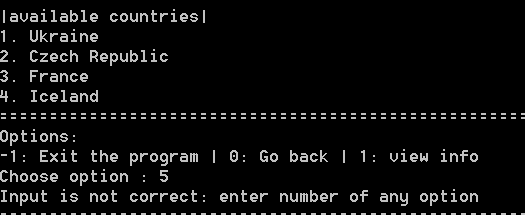
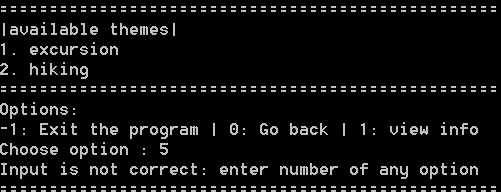
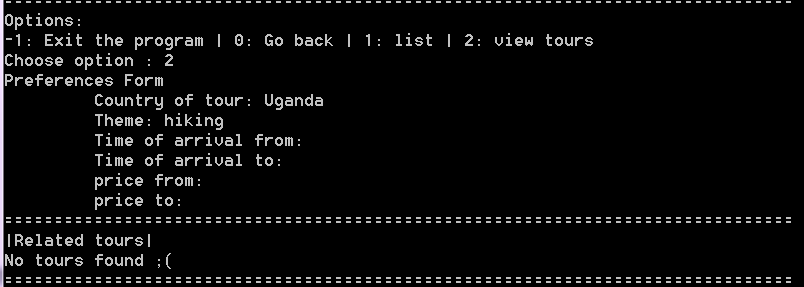
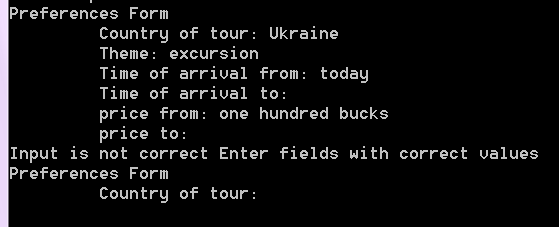
1. Вихід із програми (працює на всіх сторінках)



**4.2.3 Протестувати неправильне введення даних**

1. Неправильний вибір опцій(працює на всіх сторінках)



1. Неправильний вибір при запитувані номера (країни, тематики, туру)   
   
2. Країну або тематику не знайдено  
   
3. Заповнення некоректними значеннями 

Висновок  
У рамках цієї курсової роботи я створив програму, що надає користувачам туристичного клубу інформацію про доступні для подорожей країни та тематики, а також дає можливість шукати тури згідно вподобань користувача. Для вирішення цієї задачі було застосовану мову програмування С#, парадигму обєктно орієнтованого програмування, а також підходи щодо проектування комплексної програми. У результаті було створено програмний продукт, що відповідає поставленій задачі. Періодичні тестування допомагали знаходити помилки в програмі та виправляти їх. Фінальне тестування помилок не виявило.

Списки джерел:

* H.Schildt C# 4.0 The Complete Reference
* <https://metanit.com>
* https://docs.microsoft.com/en-us/dotnet/csharp/

Додаткові посилання:

* Репозиторій з проектом на GitHub: