**Міністерство освіти і науки України**

**Національний авіаційний університет**

**Кафедра інженерії програмного забезпечення**



**Об'єктно-орієнтоване програмування**

**Курсова робота**

**Варіант 8**

**Виконав:студент групи ПІ-225**

**Казьмін Дмитро Юрійович**

**(ПІБ студента)**

**Прийняв:Дишлевий О.П.**

**(ПІБ викладача)**

Київ 2020

Зміст

Опис завдання ………………………………………………………………………………………………………….1

Опис шарів проекту та загальна архітектура проекту …………………………………………….2

Опис інших компонентів проекту ……………………………………………………………………………3

Діаграма класів з усіми зв’язками та поясненнями ……………………………………………….4

Описання деяких особливостей реалізації …………………………………………………………….5

Описання функціоналу та використання ………………………………………………………………..5

Використані джерела ………………………………………………………………………………………………6

Опис завдання

1. В якості типу застосування обрати консольне застосування з командним рядком, як

виняток, win forms, wpf.

2. Спроектувати та реалізувати систему класів, в основу якої покладено логічну структуру

даних, наведену у варіанті, для накопичення та обробки даних домену відповідно

варіанту курсової роботи.

3. Структура програмної системи курсового проекту:

Код програмної системи має складатись не менш як з трьох частин (окремих проектів)

відповідно до багатошарової архітектури системи, де шарами архітектури є: шар

(рівень) доступу до даних (DAL), шар бізнес-логіки (BLL), шар представлення (інтерфейс

програмної системи) (PL). Тип програмного модулю – DLL.

Шар доступу до даних організувати таким чином, щоб він забезпечував збереження

даних у файлах за допомогою серіалізації або у БД із застосуванням Entity Framework.

Тип програмного модулю – DLL.

В шарі бізнес-логіки побудувати класи, що представляють об’єкти та дії над ними

відповідно до предметної області, наприклад, читачі, книги та абонементи, читач може

взяти книгу на свій абонемент при її наявності у сховищі та повернути її у заданий

термін. Для створення наборів об’єктів предметної області (студенти, викладачі, книги,

замовлення, страви, тощо) використовувати класи-узагальнені колекції. Всі операції

бізнес-логіки використовують об’єктну модель даних. Якщо дані потрібно отримати зі

сховища даних або зберегти у сховищі, шар бізнес-логіки звертається до шару доступу

до даних, передаючи\отримуючи збережені дані.

Шар представлення – це додаток типу відповідного до обраного інтерфейсу

(консольний чи GUI). Людина-користувач через шар представлення взаємодіє з

програмною системою через інтерфейс, реалізований шаром представлення. Для

виконання дій на вимогу користувача шар представлення звертається до шару бізнес-

логіки, передаючи в нього ведені користувачем дані та команди виконати певні дії.

Результати виконаних дій, отримані від шару бізнес-логіки, шар представлення

відображує користувачу у консолі\формах.

Принцип багатошаровості при будуванні архітектури є обов’язковим до

застосування!

4. Реалізувати обов’язковий контроль коректності введених даних в шарі представлення.

5. Реалізувати перевірку виняткових ситуацій в шарі бізнес-логіки та доступу до даних; в

тому числі – при роботі з даними. При необхідності створити власні класи виключень

(наприклад, виключення для ситуації перевищення ліміту кількості книжок на

абонементі в бібліотеці). Генерація/ виникнення вийняткових ситуацій, як правило

відбувається на іншому рівні (рівнях), ніж їх обробка.

6. Написати модульні тести, використовуючи певний фреймворк (MSTest, Nunit, Xunit,

тощо), до бізнес логіки. Мінімальне покриття тестами – 100% функціоналу однієї з

сутностей, а також – мінімум 50% функціоналу усіх інших сутностей. Покриття можна

продемонструвати відповідними засобами, наприклад AxoCover, CodeCoverage та ін

(допускається також детальне описання функціоналу, покритого модульними тестами).

Модульні тести повинні бути окремим проектом в рішенні. Для оформлення коду

модульних тестів обов’язково застосовувати принцип Triple A.

7. Сутності за зв’язки між ними повинні бути спроектовані, відповідно до базових правил

ООП, composition over inheritance, loose coupling – high cohesion, inversion of control

(IoC). А також - з використанням узагальнень.

8. При написанні коду застосувати правила «хорошого» стилю програмування. А саме:

створювати мнемонічні ідентифікатори, чітко форматувати код, застосувати при

необхідності коментарі. Не використовувати public полів у класах. За необхідності

доступу до них використати властивості, індексатори чи окремі методи-аксесори. Опис

класів наводити в окремих файлах. За неохайне оформлення коду можливе зниження

оцінки за курсову роботу!

9. Пояснювальна записка має містити наступні складові:

a. Титульний лист

b. Зміст

c. Опис завдання

d. Опис шарів проекту та загальна архітектура проекту

e. Опис інших компонентів проекту (при наявності)

f. Діаграма класів з усіми зв’язками та поясненнями

g. Описання деяких особливостей реалізації (допускається приведення прикладів

з кількома рядками коду)

h. Описання функціоналу та використання (допускається використання кількох

скрішнотів)

i. Використані джерела

j. Лістінг коду проекту (не друкувати, а викласти в репозиторій/ додати архівом)

k. Скріншоти (результати введення\виведення) проекту (не друкувати, а а

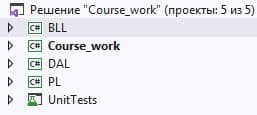
викласти в репозиторій/ додати архівом)

10. До розробленого застосування та оформленої пояснювальної записки розробити

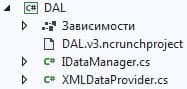
презентацію та доповідь (приблизно 5 хв.).

Опис шарів проекту та загальна архітектура проекту

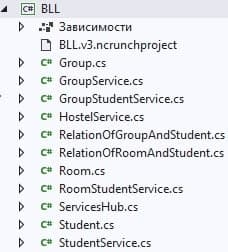
Рішення складається з п’яти проектів:



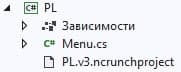
DAL – шар доступу до даних , забезпечує збереження даних у файли за допомогою серіалізації(XML). Складається із одного класу і одного інтерфейсу.



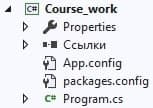
BLL – шар бізнес-логіки , описує класи об’єктів предметної області та сервіси , які проводять дії над наборами об’єктів(набори записуються в класи-узагальнені колекції) та звертаються до шару доступу до даних. Складається з одинадцяти класів.



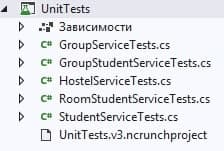
PL – шар представлення , взаємодіє з користувачем та звертається до сервісів шару бізнес-логіки. Складається із одного класу.



Course\_work – проект з якого починає роботу програма , створює об’єкт класу із шару представлення і визиває один його метод.

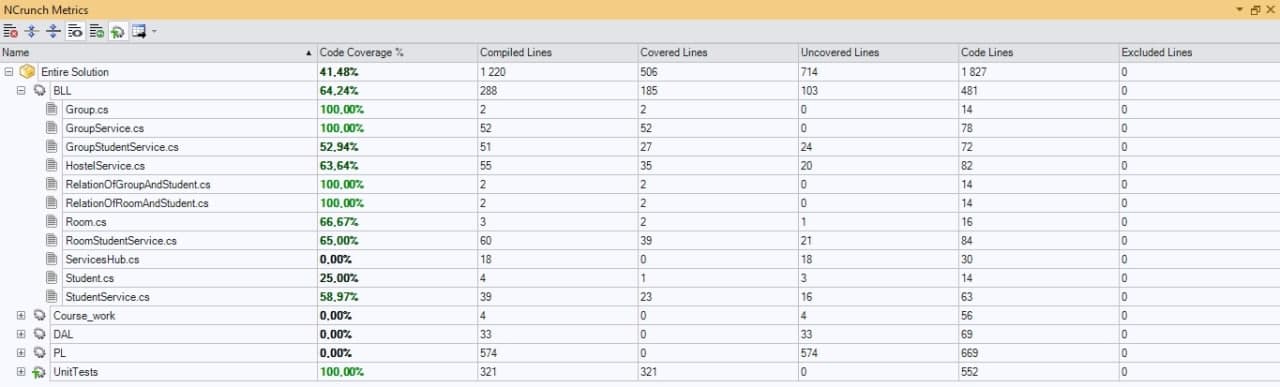


UnitTests – проект який тестує сервіси шару бізнес-логіки.

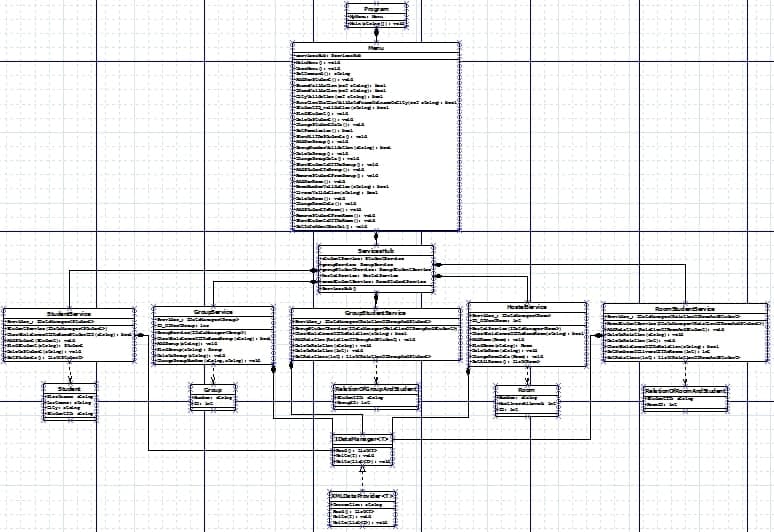


Опис інших компонентів проекту

NCrunch - це автоматичний інструмент тестування для Visual Studio. Був встановлений для визначення покриття коду.



Діаграма класів з усіми зв’язками та поясненнями



Клас XMLDataProvider<T> реалізує інтерфейс IDataManager<T>.

Клас StudentService включає в себе(композиція) інтерфейс IDataManager<Student> та залежить від класу Student.

Клас GroupService включає в себе(композиція) інтерфейс IDataManager<Group> та залежить від класу Group.

Клас GroupStudentService включає в себе(композиція) інтерфейс IDataManager<RelationOfGroupAndStudent> та залежить від класу RelationOfGroupAndStudent.

Клас HostelService включає в себе(композиція) інтерфейс IDataManager<Room> та залежить від класу Room.

Клас RoomStudentService включає в себе(композиція) інтерфейс

IDataManager< RelationOfRoomAndStudent > та залежить від класу RelationOfRoomAndStudent.

Клас ServicesHub включає в себе(композиція) класи StudentService, GroupService, GroupStudentService,

HostelService, RoomStudentService.

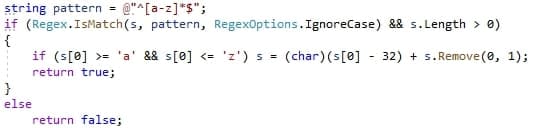
Клас Menu включає в себе(композиція) клас ServicesHub.

Клас Program включає в себе(композиція) клас Menu.

Описання деяких особливостей реалізації

Для перевірки введених користувачем даних були використані регулярні вирази.

Приклад:



Через те, що функція Main статична , то для того , щоб з неї запустити роботу класу Menu , він був теж зроблений статичним.



Описання функціоналу та використання

Клас XMLDataProvider<T> серіалізує у файл новий об’єкт , або набор об’єктів та десеріалізує набор об’єктів з файлу.

Клас StudentService додає,видаляє,перевіряє наявність або знаходить студента у наборі об’єктів. Також може отримати цей набор.

Клас GroupService додає,видаляє,перевіряє наявність або знаходить групу у наборі об’єктів. Також може змінити номер вказаної групи.

Клас GroupStudentService додає,видаляє або перевіряє наявність зв’язку групи зі студентом у наборі об’єктів. Також може отримати цей набор.

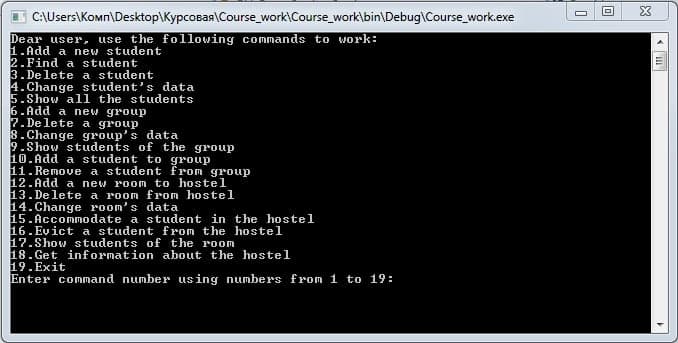
Клас HostelService додає,видаляє,перевіряє наявність або знаходить кімнату у наборі об’єктів. Також може змінити дані вказаної групи чи отримати набор об’єктів.

Клас RoomStudentService додає,видаляє або перевіряє наявність зв’язку кімнати зі студентом у наборі об’єктів. Також може отримати цей набор або кількість мешканців кімнати.

Клас ServicesHub у своєму конструкторі створює провайдерів даних задаючи їм файл , та передає цих провайдерів до відповідних сервісів.

Клас Menu спілкується з користувачем , та надає відповідні команди сервісам.

Програма спілкується з користувачем через консоль , і надає 19 команд(18 для роботи , 1 для виходу).



Користувачу треба ввести номер потрібної йому команди , та відповідно до цього виконувати інші дії , які запросить програма.

Використані джерела

1.Лекції з ООП 2 курс 1 семестр.

2.docs.microsoft.com

3.metanit.com