НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва дисципліни)

на тему:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студента (ки) \_\_\_\_\_ курсу \_\_\_\_\_\_ групи

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

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

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ- 2024рік

**ЗМІСТ**

ВСТУП.................................................................................................................................. 4

1 ПОРЯДОК ВИКОНАННЯ КУРСОВОЇ РОБОТИ......................................................... 5

1.1 Об'єктно-орієнтований аналіз предметної області ................................................. 5

1.2 Об'єктно-орієнтоване проектування ........................................................................ 6

1.3 Об'єктно-орієнтоване програмування ...................................................................... 6

1.4 Тестування програми ................................................................................................. 7

1.5 Оформлення пояснювальної записки....................................................................... 8

1.6 Захист курсової роботи.............................................................................................. 8

2 ЗМІСТ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ ....................................................................... 9

3 ПРАВИЛА ОФОРМЛЕННЯ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ ................................ 14

3.1 Загальні вимоги ........................................................................................................ 14

3.2 Нумерація.................................................................................................................. 15

3.3 Оформлення цитат і переліку посилань................................................................. 16

3.4 Оформлення додатків .............................................................................................. 17

4 ПОРЯДОК ВИКОНАННЯ ТА ЗАХИСТУ КУРСОВОЇ РОБОТИ............................. 18

4.1 Хід виконання та захисту курсової роботи ........................................................... 18

4.2 Критерії оцінювання курсової роботи ................................................................... 20

5 ТЕМИ ІНДИВІДУАЛЬНИХ ЗАВДАНЬ НА КУРСОВУ РОБОТУ........................... 25

5.1 Варіанти завдань ............................................Ошибка! Закладка не определена.

5.1.1 Тематика базових індивідуальних завданьОшибка! Закладка не

определена.

5.1.2 Тематика додаткових індивідуальних завданьОшибка! Закладка не

определена.

СПИСОК ЛІТЕРАТУРИ................................................................................................... 26

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ.......................................................................... 27

ДОДАТОК Б ТИТУЛЬНА СТОРІНКА........................................................................... 29

ДОДАТОК В ЛИСТ ЗАВДАННЯ.................................................................................... 30

ДОДАТОК Г ТЕКСТИ ПРОГРАМНОГО КОДУ........................................................... 32

**Вступ**

Виконання курсової роботи з дисципліни "Основи програмування" є невід’ємною частиною навчального процесу студентів, які навчаються за спеціальністю 121 "Інженерія програмного забезпечення" на бакалаврському рівні. Робота має на меті не лише поглиблення теоретичних знань, отриманих під час вивчення курсу, але й їх застосування для розв’язання конкретних фахових задач, що сприяє формуванню висококваліфікованих фахівців у галузі програмування.

Тема моєї курсової роботи — "ОСББ. Управління будинком. Електронний довідник даних про мешканців та орендарів багатоквартирного будинку". Проект передбачає розробку програмного забезпечення, що дозволить автоматизувати процес збору, зберігання та обробки інформації про мешканців та орендарів: їхні особисті дані, номер квартири, параметри проживання, вартість оренди та комунальних послуг.

Основними завданнями, які стоять переді мною під час виконання курсової роботи, є:

* Розробка об'єктно-орієнтованої моделі даних;
* Програмування функціоналу для розрахунку квартплати з урахуванням різних факторів, таких як площа квартири, кількість мешканців, споживання води та електроенергії;
* Створення інтуїтивно зрозумілого графічного інтерфейсу користувача;
* Реалізація механізмів збору та аналізу даних для розрахунку доходів і витрат ОСББ.

Курсова робота вимагає застосування сучасних технологій розробки програмного забезпечення і використання навичок, набутих в процесі навчання, як у теоретичному, так і в практичному аспектах. Завершення проекту потребує від студента здатності до самостійної роботи, аналітичного мислення та вміння розв’язувати комплексні задачі. За результатами виконаної роботи будуть сформовані висновки, які покажуть ефективність запропонованого рішення.

Цей проект відображає практичне застосування набутих знань і спрямований на поліпшення управління житловими комплексами через використання інноваційних програмних рішень, що є актуальним і сучасним викликом в галузі інженерії програмного забезпечення.

**1 ПОРЯДОК ВИКОНАННЯ КУРСОВОЇ РОБОТИ**

Основні етапи виконання курсової роботи:

− об'єктно-орієнтований аналіз предметної області задачі;

− об'єктно-орієнтоване проектування архітектури програмної системи;

− об'єктно-орієнтоване програмування;

− тестування програми;

− оформлення пояснювальної записки;

− захист курсової роботи.

* 1. **Об'єктно-орієнтований аналіз предметної області**

А) Предметна область цього програмного забезпечення присвячена управлінню багатоквартирними будинками та обліку інформації про мешканців, орендарів, комунальні витрати та витрати на ремонт.

Терміни та поняття:

* Багатоквартирний будинок
* Квартира
* Кімната
* Орендар (Tenant)
* Власник квартири (LandLord)
* Витрати на комунальні послуги (UtilityExpenses)
* Витрати на ремонт (RepairExpense)
* Оренда
* Дохід від оренди

Суб'єкти:

* Власники квартир (LandLord)
* Орендарі (Tenant)
* Об'єднання співвласників багатоквартирного будинку (ОСББ)

Об'єкти:

* Багатоквартирний будинок (Building)
* Квартира (Apartment)
* Кімната (Room)

Способи взаємодії суб'єктів:

* Власники квартир здають їх в оренду орендарям
* Орендарі сплачують орендну плату власникам квартир
* ОСББ збирає інформацію про мешканців, орендарів, комунальні витрати та витрати на ремонт

Способи використання об'єктів:

* Багатоквартирний будинок є сукупністю квартир
* Квартира складається з кімнат та використовується для проживання
* Кімната є частиною квартири та використовується для різних цілей (спальня, вітальня тощо)

Закономірності:

* Вартість оренди квартири залежить від її площі, кількості кімнат та місцезнаходження
* Комунальні витрати залежать від кількості мешканців, споживання ресурсів (електрика, вода, опалення тощо)
* Витрати на ремонт виникають після закінчення терміну оренди та залежать від ступеня зносу квартири
* Дохід власника квартири формується від орендної плати за вирахуванням витрат на комунальні послуги та ремонт

Отже, ця предметна область охоплює управління багатоквартирними будинками, облік інформації про мешканців, орендарів, комунальні витрати та витрати на ремонт, а також взаємодію між власниками квартир, орендарями та об'єднанням співвласників будинку.

Б) Словник предметної області включатиме такі елементи:

Іменники:

1. Будинок (Building)
2. Квартира (Apartment)
3. Кімната (Room)
4. Орендар (Tenant)
5. Власник (LandLord)
6. Оренда (Rent)
7. Витрати (Expenses)
8. Комунальні послуги (Utilities)
9. Ремонт (Repair)
10. Дохід (Income)
11. Об'єднання співвласників (Owners' Association)
12. Платіж (Payment)
13. Площа (Area)
14. Поверх (Floor)
15. Контракт (Contract)
16. Термін (Term)

Дієслова, пов'язані з іменниками:

1. Будинок: будувати, володіти, управляти, обслуговувати.
2. Квартира: орендувати, винаймати, проживати, ремонтувати, обладнати.
3. Кімната: використовувати, меблювати, прибирати.
4. Орендар: орендувати, винаймати, сплачувати, проживати, звітувати.
5. Власник: володіти, здавати в оренду, отримувати дохід, ремонтувати.
6. Оренда: укладати, сплачувати, поновлювати, припиняти.
7. Витрати: нараховувати, сплачувати, відшкодовувати.
8. Комунальні послуги: споживати, оплачувати, розраховувати.
9. Ремонт: проводити, оцінювати, фінансувати.
10. Дохід: обчислювати, отримувати, розподіляти.
11. Об'єднання співвласників: управляти, координувати, представляти.
12. Платіж: здійснювати, обробляти, відстежувати.
13. Площа: обчислювати, вимірювати, оцінювати.
14. Поверх: розташовувати, будувати.
15. Контракт: укладати, переглядати, поновлювати.
16. Термін: встановлювати, продовжувати, закінчувати.

Цей словник охоплює основні терміни та поняття, пов'язані з управлінням багатоквартирними будинками, орендою квартир, комунальними витратами, ремонтами та взаємодією між учасниками процесу.

В) Функціональні вимоги до розроблюваної системи управління багатоквартирними будинками:

1. Облік інформації про будинки:

- Можливість додавання, видалення та редагування інформації про будинки (назва, кількість поверхів, адреса тощо).

- Перегляд списку всіх будинків у системі.

2. Облік інформації про квартири:

- Можливість додавання, видалення та редагування інформації про квартири (номер, кількість кімнат, площа, поверх, вартість за квадратний метр).

- Прив'язка квартири до певного будинку.

- Перегляд списку всіх квартир у певному будинку.

3. Облік інформації про власників квартир (домовласників):

- Можливість додавання, видалення та редагування інформації про власників (ім'я, контактна інформація, квартири у власності).

- Перегляд списку всіх власників квартир у системі.

- Перегляд інформації про квартири, якими володіє певний власник.

4. Облік інформації про орендарів:

- Можливість додавання, видалення та редагування інформації про орендарів (ім'я, контактна інформація, орендована квартира).

- Перегляд списку всіх орендарів у системі.

- Перегляд інформації про квартиру, яку орендує певний орендар.

5. Управління орендою квартир:

- Можливість встановлення терміну оренди для квартири.

- Розрахунок вартості оренди квартири на основі площі, кількості кімнат та вартості за квадратний метр.

- Облік доходу від оренди для власника квартири.

6. Облік комунальних витрат:

- Можливість додавання та редагування інформації про комунальні витрати для певної квартири (вартість електроенергії, води, опалення тощо).

- Перегляд історії комунальних витрат для квартири.

7. Облік витрат на ремонт:

- Можливість додавання та редагування інформації про витрати на ремонт для певної квартири (опис ремонту, вартість).

- Перегляд історії витрат на ремонт для квартири.

8. Звітність та аналітика:

- Генерація звітів про дохід від оренди для власників квартир.

- Генерація звітів про комунальні витрати та витрати на ремонт для квартир.

- Аналіз та візуалізація даних (наприклад, графіки доходів або витрат за певний період).

9. Обробка виключних ситуацій:

- Належна обробка виключних ситуацій, таких як некоректний введений користувачем вхідний дані, порушення обмежень даних тощо.

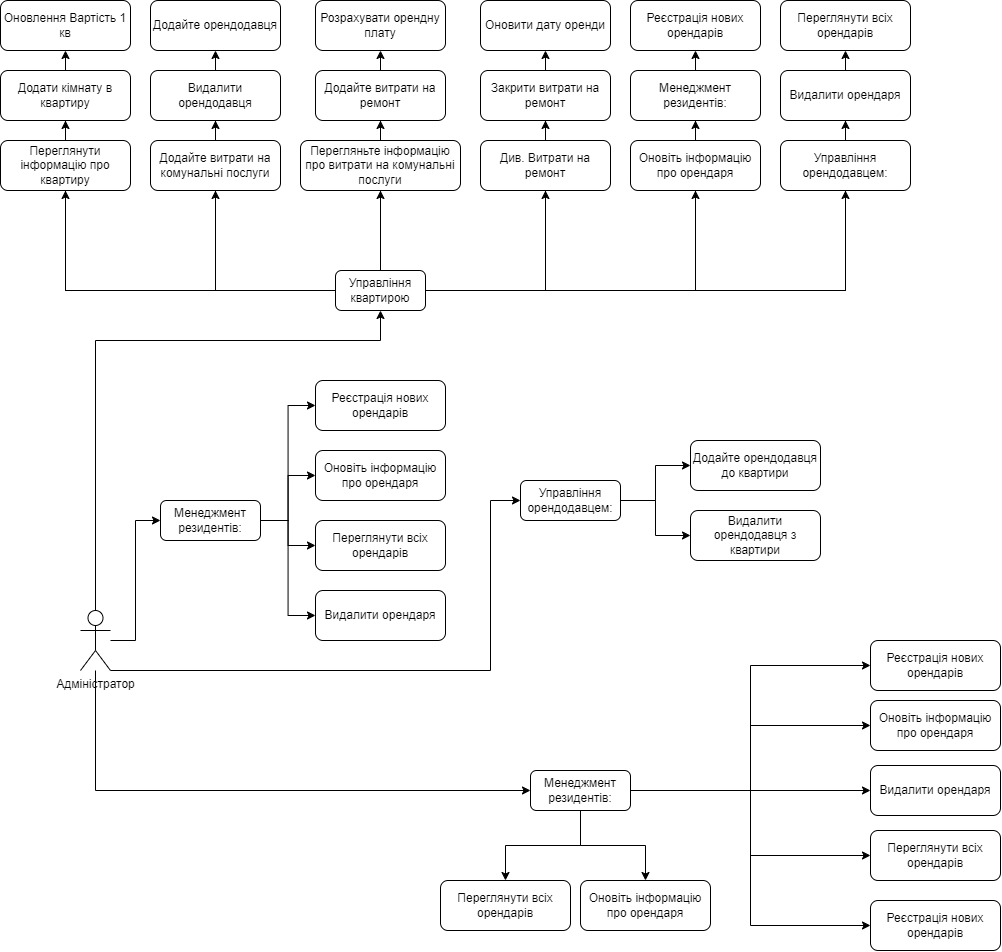
Ці функціональні вимоги охоплюють основні аспекти управління багатоквартирними будинками, обліку інформації про власників, орендарів, квартири, оренду, комунальні витрати та ремонти, а також генерацію звітності та аналітики.

Приблизний сценарій поведінки об'єктів в системі управління багатоквартирними будинками:

1. Створення об'єктів класу Building (будинки):
   * Адміністратор системи створює нові об'єкти класу Building, вказуючи назву будинку та кількість поверхів.
2. Додавання квартир до будинку:
   * Для кожного будинку адміністратор створює об'єкти класу Apartment, вказуючи номер квартири, кількість кімнат, поверх, вартість за квадратний метр та прив'язуючи її до відповідного об'єкту Building.
3. Створення об'єктів класу LandLord (власників квартир):
   * Адміністратор створює об'єкти класу LandLord для кожного власника квартири, вказуючи їх контактні дані.
   * Для кожного власника квартири встановлюється зв'язок з відповідними об'єктами класу Apartment, які він володіє.
4. Здача квартир в оренду:
   * Власник квартири (об'єкт LandLord) встановлює терміни оренди для своїх квартир (об'єктів Apartment) та визначає вартість оренди.
   * Система розраховує дохід власника квартири від оренди на основі вартості оренди та терміну.
5. Створення об'єктів класу Tenant (орендарів):
   * Адміністратор або власник квартири створює об'єкти класу Tenant для кожного орендаря, вказуючи їх контактні дані.
   * Для кожного орендаря встановлюється зв'язок з об'єктом класу Apartment, який він орендує.
6. Облік комунальних витрат:
   * Власник квартири або адміністратор створює об'єкти класу UtilityExpenses для кожної орендованої квартири, вказуючи витрати на електроенергію, воду, опалення тощо за певний період (місяць).
   * Об'єкти UtilityExpenses зберігаються у відповідних об'єктах класу Apartment.
7. Облік витрат на ремонт:
   * Після закінчення терміну оренди власник квартири створює об'єкти класу RepairExpense для своїх квартир, вказуючи опис ремонту та вартість.
   * Об'єкти RepairExpense зберігаються у відповідних об'єктах класу Apartment.
8. Генерація звітів та аналітики:
   * Адміністратор або власник квартири може запросити звіти про дохід від оренди для певного власника квартири або для всіх власників.
   * Адміністратор або власник квартири може запросити звіти про комунальні витрати та витрати на ремонт для певної квартири або для всіх квартир.
   * Система може генерувати графіки та візуалізації на основі даних про доходи, витрати тощо.
9. Обробка виключних ситуацій:
   * У випадку некоректного введення даних користувачем або порушення обмежень даних, система повинна належним чином обробляти ці виключні ситуації та повідомляти користувача про помилки.

Цей сценарій демонструє, як об'єкти різних класів (Building, Apartment, LandLord, Tenant, UtilityExpenses, RepairExpense) взаємодіють між собою для виконання різних функцій системи управління багатоквартирними будинками, таких як облік інформації про будинки, квартири, власників та орендарів, управління орендою, облік комунальних витрат та ремонтів, а також генерація звітності та аналітики.

Діаграмма прецедентів можна побачити на - **рис 1.1**



**Рис 1.1**

* 1. **Об'єктно-орієнтоване проектування**

1.2 Об'єктно-орієнтоване проектування

На цьому етапі розробки курсового проекту описується архітектура системи, яка базується на принципах об'єктно-орієнтованого програмування (ООП).

1.2.1 Обробка словника предметної області

Першим кроком в ООП-проектуванні є аналіз предметної області, що включає виявлення ключових понять та їх характеристик. Це робиться шляхом обробки словника предметної області, який містить перелік іменників та дієслів.

Іменники слугують основою для виділення класів системи. Класи представляють собою сутності предметної області, які мають власні властивості та методи.

Дієслова використовуються для визначення методів класів. Методи описують дії, які може виконувати об'єкт класу.

На основі аналізу словника предметної області формується діаграма класів, яка відображає зв'язки між класами та їхні характеристики.

1.2.2 Визначення характеристик класів

Наступним кроком є визначення характеристик (атрибутів та методів) виділених класів.

Атрибути описують властивості об'єкта класу. Атрибути можуть бути простими (наприклад, тип даних int, string) або складними (масиви, об'єкти інших класів).

Методи описують дії, які може виконувати об'єкт класу. Методи можуть мати параметри та повертати значення.

Важливо зазначити, що набори атрибутів і методів у різних класах можуть перетинатися. Це означає, що один і той самий атрибут або метод може використовуватися в декількох класах.

1.2.3 Виявлення відносин між класами

Одним з ключових принципів ООП є інкапсуляція, яка передбачає приховування внутрішньої реалізації класу та надання доступу до нього лише через його інтерфейс. Цей принцип реалізується за допомогою відносин між класами.

Існує два основних типи відносин між класами:

Успадкування - це відношення, яке дозволяє одному класу (потомку) успадковувати властивості та методи іншого класу (батька). Це означає, що об'єкти класу-потомка мають доступ до всіх атрибутів і методів класу-батька.

Агрегація - це відношення, яке описує зв'язок частин з цілим. Це означає, що один об'єкт класу може містити один або декілька об'єктів іншого класу.

1.2.4 Застосування принципів ООП

На етапі ООП-проектування важливо враховувати не тільки особливості предметної області, але й такі принципи ООП як абстракція, інкапсуляція, поліморфізм.

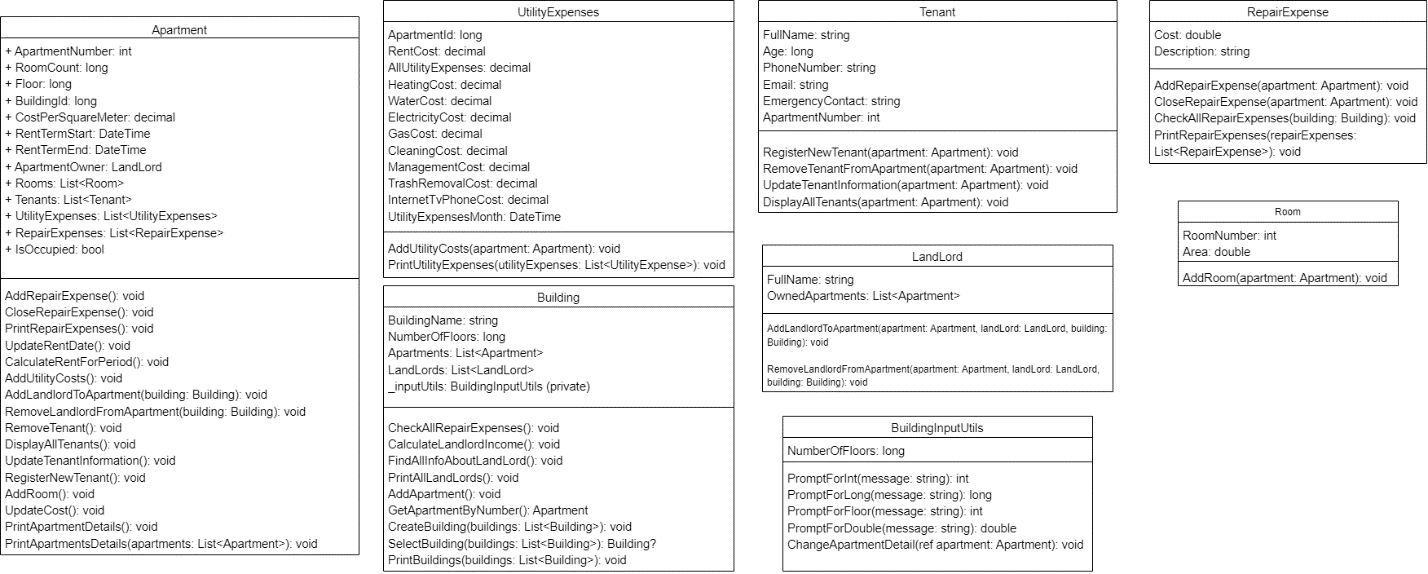
Абстракція - це принцип, який дозволяє приховувати деталі реалізації об'єкта та надавати користувачеві доступ лише до його суттєвих характеристик.

Інкапсуляція - це принцип, який описує приховування внутрішньої реалізації об'єкта та надання доступу до нього лише через його інтерфейс.

Поліморфізм - це принцип, який дозволяє об'єктам різних класів реалізовувати одну і ту ж операцію по-різному.

1.2.5 Результати проектування

Результатом ООП-проектування є діаграма класів, яка відображає зв'язки між класами та їхні характеристики. Діаграма класів є важливим інструментом для розуміння архітектури системи та її подальшої реалізації.

Діаграмма классів **– рис 1.2**

**Рис 1.2**

* 1. **Об'єктно-орієнтоване програмування**

Об'єктно-орієнтоване програмування (ООП) - це парадигма програмування, яка базується на концепції "об'єктів", які містять дані у формі полів (іноді називаються атрибутами або властивостями) та коду у формі процедур (відомих як методи). Одним із ключових принципів ООП є інкапсуляція - упаковка даних та методів, що маніпулюють цими даними, в одному місці.

На етапі реалізації класів попередньо спроектована діаграма класів використовується для написання програмного коду, що реалізує ці класи. Діаграма забезпечує зручну відправну точку, визначаючи ієрархію класів, властивості та прототипи методів. Головний акцент робиться на реалізації алгоритмів та програмуванні методів класів.

Кожен клас зазвичай міститься в окремому файлі з двома основними розділами:

1. **Інтерфейс класу**: Тут визначаються властивості та методи класу, які становлять його публічний інтерфейс. Це ті частини класу, до яких можуть звертатися інші класи.
2. **Реалізація класу**: Тут знаходиться реальний код, який реалізує поведінку методів класу. Це приватна частина класу, недоступна ззовні.

Для тестування та демонстрації функціональності розроблюється окремий додаток. Цей додаток створює екземпляри класів, викликає їх методи та перевіряє, чи вони працюють належним чином. Він може мати графічний інтерфейс користувача (GUI) з меню та інтерактивними елементами управління, що дозволяють користувачеві вводити дані та взаємодіяти із системою.

Повний код, включаючи класи та тестовий додаток, має бути ретельно задокументований за допомогою коментарів та належних пояснень. Це допоможе полегшити розуміння коду та його подальше супроводження.

Давайте тепер перейдемо до практичної частини та розглянемо реалізацію наданих вами класів у контексті цього об'єктно-орієнтованого підходу.

* 1. **Тестування програми**

На цьому етапі проводиться тестування розробленого програмного забезпечення.

Важливо розрізняти процеси тестування і налагодження програмного коду. Налагодження програмного коду виконується програмістом за допомогою вбудованих засобів середовища розробки і зводиться, в основному, до виявлення синтаксичних і семантичних помилок в тексті програми.

Тестування - це процес, що вимагає планування і виконання ряду попередніх процедур, основною з яких є складання набору тестових прикладів, що утворюють тест-план. Тестові приклади в більшості випадків грунтуються на функціональних вимогах до системи і можуть зачіпати різні рівні розробки, такі як модульне тестування, інтеграційне тестування, системне тестування.

Методика тестування розробленого програмного забезпечення передбачає виконання спрощеного варіанта модульного тестування. Це означає тестування всіх методів розроблених класів. Тому план тестування повинен містити список різних варіантів виклику методів класів з очікуваними результатами і порядок тестового виконання цих викликів. Тести можна згрупувати для однотипних методів, щоб на наступному етапі проектування тестового додатка це було враховано.

Після успішного тестування програми можна переходити до наступних етапів. У випадку невдачі під час тестування потрібно повернутись до попередніх етапів курсового проектування. Результати тестування та виявлені обмеження програми необхідно задокументувати.

* 1. **Оформлення пояснювальної записки**

**ДОДАТОК Б ТИТУЛЬНА СТОРІНКА**

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва дисципліни)

на тему:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студента (ки) \_\_\_\_\_ курсу \_\_\_\_\_\_ групи

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

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

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

**\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ- 2024 рік

ДОДАТОК В ЛИСТ ЗАВДАННЯ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Спеціальність «Інженерія програмного забезпечення»

Курс \_\_ Група ІП-\_\_\_ Семестр \_\_\_

**ЗАВДАННЯ**

**на курсову роботу студента**

Ткаченко Костянтин Олександрович

(прізвище, ім’я, по батькові)

1. Тема роботи

2. Строк здачі студентом закінченої роботи

3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень)

6. Дата видачі завдання

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін вико нання етапів  роботи | Підписи ке рівника, сту дента |
| 1. | Отримання теми курсової роботи |  |  |
| 2. | Підготовка ТЗ |  |  |
| 3. | Аналіз предметної області |  |  |
| 4. | Проектування архітектури програмної системи |  |  |
| 5. | Розробка сценарію роботи програми |  |  |
| 6. | Узгодження з керівником інтерфейсу користувача |  |  |
| 7. | Розробка програмного забезпечення |  |  |
| 8. | Узгодження з керівником плану тестування |  |  |
| 9. | Тестування програми |  |  |
| 10. | Підготовка пояснювальної записки |  |  |
| 11. | Здача курсової роботи на перевірку |  |  |
| 12. | Захист курсової роботи |  |  |

Студент

(підпис) (прізвище, ім’я, по батькові)

Керівник

(підпис) (прізвище, ім’я, по батькові)

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_\_ р.

**Анотація:**

Курсова робота реалізує консольний додаток для управління житловими будинками, апартаментами, мешканцями та власниками будинків. Додаток надає можливість створення та вибору будинків, додавання апартаментів до будинків, а також управління апартаментами, мешканцями та власниками. Користувач може здійснювати фінансові транзакції, такі як додавання витрат на комунальні послуги та розрахунок орендної плати. Також реалізовані функції для управління власниками будинків, включаючи перегляд інформації про власників та розрахунок загального доходу від оренди. Використовується об'єктно-орієнтований підхід для створення та управління об'єктами будинків, апартаментів та мешканців.

Вступ ................................................................................................................. 24

Опис основного функціоналу .............................................................................. 4

Реалізація консольного додатку .......................................................................... 5

2.1. Основна частина;................................................................................... 5

2.2.1 Постановка задачі.....................................................

2.2.2 Аналіз предметної області.......................................... 8

2.3.3 Опис архітектури програмної системи ........................................ 10

2.4.4 Опис програмного забезпечення ................................................. 12

2.5.5 Результати тестування .................................................................. 14

2.6.6 Інструкція користувача ................................................................ 15

Висновки ........................................................................................................... 16

Список використаних джерел ............................................................................. 17

Додатки .............................................................................................................. 18

**Вступ**

Курсова робота має на меті розробку консольного додатку для керування житловими будинками, апартаментами, мешканцями та власниками. Головною метою цього додатку є створення зручного та ефективного інструменту для автоматизації процесів управління нерухомістю. У вступі обґрунтовується актуальність теми дослідження, а також вказується на потребу у розробці програмного забезпечення, спрямованого на оптимізацію управління житловими приміщеннями та пов'язаними з ними процесами.

Для досягнення цієї мети у розділі "Опис основного функціоналу" надається огляд функцій, які має виконувати розроблений додаток. Детально розглядається функціонал управління будинками, апартаментами, мешканцями та власниками, а також фінансові транзакції, пов'язані з управлінням нерухомістю.

У наступному розділі "Реалізація консольного додатку" надається опис структури проекту та основного алгоритму роботи програми. Детально розглядаються основні класи та їх взаємодія.

В розділі "Управління будинками" розглядаються процеси створення та вибору будинків у додатку. Описується функціонал для додавання та вибору будівельних об'єктів.

Послідовно, у розділі "Управління апартаментами" надається огляд функціоналу для управління апартаментами, включаючи їх створення, перегляд інформації та роботу з мешканцями та власниками.

У "Фінансові транзакції" описуються процеси фінансового обліку в додатку, включаючи роботу з витратами та розрахунок арендної плати.

В розділі "Управління власниками будинків" аналізується функціонал, пов'язаний з управлінням власниками нерухомості, включаючи перегляд інформації про власників та розрахунок доходу від оренди.

* 1. **Опис основного функціоналу**

Цей розділ надає огляд основного функціоналу розробленого консольного додатку для керування житловими будинками, апартаментами, мешканцями та власниками. Описуються основні можливості програми, які включають в себе:

**Створення та управління будинками:**

* + Можливість створення нових будинків з вказанням основних характеристик.
  + Перегляд списку існуючих будинків.
  + Вибір конкретного будинку для подальшої роботи.

**Управління апартаментами:**

* + Додавання нових апартаментів до вже існуючих будинків.
  + Перегляд інформації про апартаменти, включаючи їх розмір, кількість кімнат тощо.
  + Обробка дій з мешканцями та власниками апартаментів, такі як реєстрація нових мешканців, оновлення інформації про них тощо.

**Фінансові транзакції:**

* + Додавання витрат на комунальні послуги та інші розходи.
  + Розрахунок орендної плати за проживання в апартаментах.

**Управління власниками будинків:**

* + Перегляд інформації про власників нерухомості.
  + Розрахунок доходу від оренди за апартаменти.

Цей розділ робить акцент на функціях програмного забезпечення, які дозволяють зручно та ефективно керувати житловою нерухомістю та пов'язаними з нею процесами.

**2. Реалізація консольного додатку**

**2.1**. **Структура проекту**

Структура проекту консольного додатку включає наступні основні класи:

* **Клас Program**: Це вхідна точка програми, де знаходиться метод Main(). В цьому класі відбувається ініціалізація додатку та його головний цикл роботи.
* **Клас Building**: Відповідає за моделювання будинків, зберігає інформацію про апартаменти та власників.
* **Клас Apartment**: Представляє модель апартаментів у будинках. Містить методи для управління мешканцями, власниками та фінансовими операціями.
* **Клас Landlord**: Відповідає за моделювання власників нерухомості та їхню інформацію.
* **Клас UtilityExpense**: Моделює витрати на комунальні послуги та інші витрати.
* **Клас RepairExpense**: Моделює витрати на ремонтні роботи в апартаментах.
* **Клас Room**: Відповідає за моделювання кімнат у апартаментах.
* **Клас Tenant**: Представляє модель мешканців апартаментів.

**2.2**. **Основний алгоритм роботи**

1. **Ініціалізація та завантаження даних:**
   * При запуску програми відбувається ініціалізація необхідних змінних та структур даних.
   * Завантажуються дані про наявні будівлі та апартаменти з файлу (якщо такий існує).
2. **Відображення головного меню:**
   * Користувачеві пропонується меню з трьома пунктами:
     + Створення нового будинку
     + Вибір існуючого будинку
     + Перегляд списку всіх будинків
3. **Обробка вибору користувача:**
   * Залежно від обраного пункту меню виконується відповідна дія:
     + **Створення нового будинку:**
       - Запит у користувача інформації про новий будинок (назва, адреса, кількість поверхів тощо).
       - Створення нового об'єкта "Будинок" з отриманою інформацією.
       - Збереження нового будинку у списку будівель.
     + **Вибір існуючого будинку:**
       - Відображення списку доступних будівель.
       - Запит у користувача номера вибраного будинку.
       - Завантаження інформації про обраний будинок з списку або з файлу.
       - Перехід до меню роботи з обраним будинком.
     + **Перегляд списку всіх будинків:**
       - Відображення списку всіх доступних будівель з базовою інформацією (назва, адреса).
       - Повернення до головного меню.
4. **Робота з обраним будинком:**
   * Після вибору будинку користувачеві пропонується меню з такими пунктами:
     + Додати апартамент
     + Переглянути всі апартаменти
     + Вибрати апартамент
     + Операції з власниками
     + Вихід
   * Обробка вибору користувача аналогічно до головного меню.
5. **Робота з апартаментом:**
   * Після вибору апартаменту користувачеві пропонується меню з такими пунктами:
     + Керування апартаментом
     + Керування мешканцями
     + Керування власниками
     + Фінансові операції
     + Вихід
   * Обробка вибору користувача аналогічно до меню роботи з будинком.
6. **Повторення циклу:**
   * Пункти 2-5 повторюються до тих пір, поки користувач не обере пункт "Вихід" на будь-якому рівні меню.
7. **Завершення роботи:**
   * При виборі пункту "Вихід" програма завершує роботу.

**Основна частина**

**2.2.1 Постановка задачі**

У цьому розділі буде представлена інформаційна сутність задачі, яка є ключовою для розробки програмної системи. Основною метою цього розділу є аналіз специфікації вимог технічного завдання на курсовий проект та формулювання додаткових вимог, що випливають із нього та мети курсової роботи.

Результатом аналізу буде конкретизація постановки завдання, структури вхідних та вихідних даних, основних операцій та методів, які використовуються для їх обробки. Також будуть визначені основні параметри та функціональні можливості програмної системи, що становлять загальну концепцію розроблення.

Інформаційна сутність задачі включає:

* Опис предметної області: ОСББ (Об’єднання співвласників багатоквартирного будинку) та управління багатоквартирним будинком.
* Опис функціоналу електронного довідника даних про мешканців та орендарів, включаючи відомості про ПІБ, номер квартири, термін та вартість оренди, розмір квартири, кількість мешканців, вартість квартирної плати та інші параметри.
* Вимоги до розробки програмної системи: використання об’єктно-орієнтованого підходу, дотримання принципів ООП, впровадження обробки виключень, застосування єдиної конвенції найменувань та принципів написання "чистого" коду.
* Перелік основних операцій та методів, які необхідно реалізувати у програмній системі.
* Опис вхідних та вихідних даних, включаючи формати та способи представлення інформації.

Цей розділ визначає загальний контекст та обґрунтовує необхідність розробки програмної системи, що забезпечить ефективне управління багатоквартирним будинком та покращить взаємодію між його мешканцями та орендарями.

**2.2.2 Аналіз предметної області**

У цьому розділі буде проведено аналіз предметної області проекту, що полягатиме у реферативному описі основних аспектів ОСББ (Об’єднання співвласників багатоквартирного будинку) та управління багатоквартирним будинком.

Реферативний опис предметної області:

* ОСББ: це об’єднання власників квартир у багатоквартирному будинку, які спільно приймають рішення щодо управління та експлуатації спільних частин будинку.
* Управління багатоквартирним будинком: включає в себе організацію обслуговування будинку, забезпечення ремонтних робіт, контроль за платежами та інші аспекти повсякденного життя мешканців та орендарів.
* Електронний довідник даних: програмна система, яка забезпечує зберігання та обробку інформації про мешканців та орендарів багатоквартирного будинку, включаючи їх особисті дані, інформацію про квартири, умови оренди, платежі та інше.

Діаграма прецедентів системи:

1. Реєстрація нових мешканців та орендарів.
2. Оновлення інформації про мешканців та орендарів.
3. Перегляд інформації про всіх мешканців та орендарів.
4. Видалення мешканців та орендарів з бази даних.
5. Перегляд інформації про конкретного мешканця або орендаря.

Сценарій роботи програми:

1. Користувач запускає програму.
2. Вибирає необхідну опцію з меню.
3. Здійснює відповідні дії, які включають реєстрацію, оновлення, перегляд та видалення інформації про мешканців та орендарів.
4. Завершує роботу програми.

Аналіз предметної області дозволяє краще зрозуміти потреби користувачів та визначити функціональні можливості програмної системи.

**2.2.3 Опис архітектури програмної системи**

Архітектура програмної системи визначає структуру та організацію програмних компонентів, їх взаємозв'язки та способи взаємодії між ними. У випадку нашої програмної системи, основною архітектурною концепцією є об'єктно-орієнтований підхід, що дозволяє моделювати різні аспекти системи як окремі об'єкти з властивостями та методами.

**Компоненти системи:**

1. **Класи**: В системі існує ряд класів, таких як **Apartment**, **Building**, **LandLord**, **Tenant**, **Room**, **UtilityExpense**, та **RepairExpense**, які відображають основні сутності системи та їх взаємозв'язки.
2. **Методи**: Кожен клас має власні методи для виконання різних операцій, таких як додавання витрат на ремонт, обслуговування комунальних послуг, реєстрація нового орендаря тощо.
3. **Взаємозв'язки**: Класи взаємодіють між собою через методи та властивості. Наприклад, клас **Building** може містити список квартир (**Apartment**), а кожна квартира може мати свого власника (**LandLord**).

**Архітектурні засоби:**

1. **Об'єктно-орієнтований підхід**: Використання класів, об'єктів, властивостей та методів для моделювання системи.
2. **Розділення на класи та файли**: Кожен клас має відповідний файл з інтерфейсом та реалізацією, що сприяє організації коду та його зручності під час розвитку та зберігання.
3. **Детальна документація**: Кожен клас та метод повинні бути детально задокументовані, щоб забезпечити зрозумілість та легкість у використанні для інших розробників.

**2.2.4 Опис програмного забезпечення**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  п/п | Назва классу | Назва методу | Призначення  методу | Опис вхідних  Парамет ів | Опис вихідних парамет рів | Заголовний  файл |
| 1 | Apartment | AddRepairExpense() | Додавання нового запису про витрати на ремонт апартаментів. | - | - | Apartment.cs |
| 2 | Apartment | CloseRepairExpense() | Закрити запис про витрати на ремонт апартаментів. | - | - | Apartment.cs |
| 3 | Apartment | PrintRepairExpenses() | Вивести на друк інформацію про витрати на ремонт апартаментів. | - | - | Apartment.cs |
| 3 | Apartment | UpdateRentDate() | Оновити дату початку та кінця орендного терміну для апартаментів. | - | - | Apartment.cs |
| 4 | Apartment | CalculateRentForPeriod() | Розрахувати та вивести загальну суму орендної плати за певний період. | - | - | Apartment.cs |
| 5 | Apartment | AddUtilityCosts() | Додати новий запис про витрати на комунальні послуги для апартаментів. | - | - | Apartment.cs |
| 6 | Apartment | AddLandlordToApartment(Building building) | Додати орендодавця до апартаментів. | building (Building): Об'єкт класу Building, що представляє будівлю, до якої належать апартаменти | - | Apartment.cs |
| 7 | Apartment | RemoveLandlordFromApartment(Building building) | RemoveLandlordFromApartment(Building building) | building (Building): Об'єкт класу Building, що представляє будівлю, до якої належать апартаменти. | - | Apartment.cs |
| 8 | Apartment | RemoveTenant() | Видалити орендаря з апартаментів. | - | - | Apartment.cs |
| 9 | Apartment | DisplayAllTenants() | Вивести інформацію про всіх орендарів апартаментів. | - | - | Apartment.cs |
| 10 | Apartment | UpdateTenantInformation() | Оновити інформацію про орендаря апартаментів. | - | - | Apartment.cs |
| 11 | Apartment | RegisterNewTenant() | Зареєструвати нового орендаря для апартаментів. | - | - | Apartment.cs |
| 12 | Apartment | AddRoom() | Додати кімнату до апартаментів. | - | - | Apartment.cs |
| 13 | Apartment | UpdateCost() | Цей метод дозволяє оновити вартість оренди за квадратний метр для апартаментів. | - | - | Apartment.cs |
| 14 | Apartment | PrintApartmentDetails() | Друк детальної інформації про апартаменти. | - | - | Apartment.cs |
| 15 | Apartment | PrintApartmentsDetails(List<Apartment> apartments) (статичний): | Друк інформації про декілька апартаментів. | apartments (List<Apartment>): Список апартаментів, для яких потрібно вивести інформацію. | - | Apartment.cs |
| 16 | Apartment | CalculateRentForPeriod(DateTime startPeriod, DateTime zendPeriod) | Ця приватна функція розраховує суму оренди за заданий період. | startPeriod: Дата початку періоду оренди.  endPeriod: Дата закінчення періоду оренди. | Повертає double значення, що являє собою загальну суму оренди за заданий період. | Apartment.cs |
| 17 | Apartment | CalculateTotalArea() | Ця приватна функція розраховує загальну площу квартири. | - | Повертає double значення, що являє собою загальну площу квартири. | Apartment.cs |
| 18 | Building | CheckAllRepairExpenses | Ця функція викликає метод CheckAllRepairExpenses класу RepairExpense, ймовірно, передаючи поточний об'єкт Building як аргумент. | this (об'єкт Building) | - | Building.cs |
| 19 | Building | CalculateLandlordIncome | Ця функція розраховує дохід власника нерухомості на основі оренди його квартир протягом певного періоду. | - | - | Building.cs |
| 20 | Building | FindAllInfoAboutLandLord() | Знайти всю інформацію про конкретного власника нерухомості. | - | - | Building.cs |
| 21 | Building | PrintAllLandLords() | **:** Вивести список усіх власників нерухомості. | - | - | Building.cs |
| 22 | Building | AddApartment() | Додати нову квартиру до списку квартир будів | - | - | Building.cs |
| 23 | Building | GetApartmentByNumber() | Отримати об'єкт квартири за її номером. | - | Об'єкт класу Apartment (якщо знайдено квартиру за номером) або null (якщо квартира не знайдена) | Building.cs |
| 24 | Building | CreateBuilding() | Створити новий об'єкт класу Building та додати його до списку будівель. | List<Building> buildings: Список будівель, до якого буде додано новий об'єкт. | - | Building.cs |
| 25 | Building | SelectBuilding() | Обрати будівлю з наданого списку. | List<Building> buildings: Список будівель, з якого користувач буде обирати. | Building?: Об'єкт класу Building (якщо вибрано будівлю) або null (якщо вибір недійствний або немає будівель) | Building.cs |
| 26 | Building | PrintBuilding() | Вивести список усіх будівель. | List<Building> buildings: Список будівель для виводу. | - | Building.cs |
| 27 | RepairExpense | AddRepairExpense | Додати нові витрати на ремонт до квартири. | Apartment apartment: Об'єкт класу Apartment, до якого додаються витрати на ремонт. | - | RepairExpense.cs |
| 28 | RepairExpense | CloseRepairExpense | Закрити витрати на ремонт (видалити їх зі списку витрат квартири). | Apartment apartment: Об'єкт класу Apartment, з якого видаляються витрати на ремонт. | - | RepairExpense.cs |
| 29 | RepairExpense | CheckAllRepairExpenses | Перевірити всі витрати на ремонт для певного власника нерухомості. | Building building: Об'єкт класу Building, який містить інформацію про власників та їх квартири. | - | RepairExpense.cs |
| 30 | RepairExpense | PrintRepairExpenses | Вивести список витрат на ремонт. | List<RepairExpense> repairExpenses: Список об'єктів класу RepairExpense, що містить дані про витрати на ремонт. | - | RepairExpense.cs |
| 31 | Room | AddRoom | Додати нову кімнату до квартири. | Apartment apartment: Об'єкт класу Apartment, до якого додається кімната. | - | Room.cs |
| 32 | UtilityExpense | AddUtilityCosts | Додати витрати на комунальні послуги для певного місяця до квартири. | Apartment apartment: Об'єкт класу Apartment, до якого додаються витрати на комунальні послуги. | - | UtilityExpense.cs |
| 33 | UtilityExpense | PrintUtilityExpenses | Вивести список витрат на комунальні послуги. | List<UtilityExpense> utilityExpenses: Список об'єктів класу UtilityExpense, що містить дані про витрати на комунальні послуги. | - | UtilityExpense.cs |
| 34 | LandLord | AddLandlordToApartment | Додати власника нерухомості (LandLord) до квартири. | Apartment apartment: Об'єкт класу Apartment, до якого додається власник.  Building building: Об'єкт класу Building, до якого належить квартира. | - | LandLord.cs |
| 35 | LandLord | RemoveLandlordFromApartment | Видалити власника нерухомості з квартири. | Apartment apartment: Об'єкт класу Apartment, до якого додається власник.  Building building: Об'єкт класу Building, до якого належить квартира. | - | LandLord.cs |
| 36 | Tenant | RegisterNewTenant | Зареєструвати нового орендаря в квартирі. | Apartment apartment: Об'єкт класу Apartment, в якому реєструється орендар. | - | Tenant.cs |
| 37 | Tenant | RemoveTenantFromApartment | Видалити орендаря з квартири. | Apartment apartment: Об'єкт класу Apartment, з якого видаляється орендар. | - | Tenant.cs |
| 38 | Tenant | UpdateTenantInformation | Оновити інформацію про орендаря. | Apartment apartment: Об'єкт класу Apartment, в якому проживає орендар. | - | Tenant.cs |
| 39 | Tenant | DisplayAllTenants | Вивести список усіх орендарів квартири. | Apartment apartment: Об'єкт класу Apartment, для якого виводяться орендарі. | - | Tenant.cs |
| 40 | Building | PromptForInt | Запросити в користувача ціле число. | message (string): Строкове повідомлення, що виводиться перед запитом вводу. | int: Ціле число, введене користувачем. | Building |
| 41 | Building | PromptForLong | Запросити в користувача довге ціле число. | message (string): Строкове повідомлення, що виводиться перед запитом вводу. | long: Довге ціле число, введене користувачем. | Building |
| 42 | Building | PromptForFloor | Запросити в користувача номер поверху, враховуючи загальну кількість поверхів у будівлі. | message (string): Строкове повідомлення, що виводиться перед запитом вводу. | int: Номер поверху, введений користувачем. | Building |
| 43 | Building | PromptForDouble | Запросити в користувача число з рухомою комою. | message (string): Строкове повідомлення, що виводиться перед запитом вводу. | double: Число з рухомою комою, введене користувачем. | Building |
| 44 | Building | ChangeApartmentDetail | Дозволяє змінити деталі квартири. | apartment (ref Apartment): Посилання на об'єкт класу Apartment, деталі якого потрібно змінити. | - | Building |

**2.2.7 Результати його тестування**

У цьому розділі представлені методики та результати тестування програмної системи, яка була розроблена для управління нерухомістю. Тестування було спрямоване на перевірку функціональності системи, обробку виключень, а також точність обчислень.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Модуль | Тестовий сценарій | Очікуваний результат | Результат | Статус |
| Створення будівлі | Створити будівлю з усіма заповненими полями (адреса, кількість поверхів) | Повинно бути створено новий об'єкт класу Building з введеними даними. | Створено новий об'єкт класу Building. | Пройдено |
| Створення будівлі | Створити будівлю з пустими полями (адреса, кількість поверхів) | Повинно бути повідомлення про помилку і запит на введення даних. | Виведено повідомлення про помилку. | Пройдено |
| Створення будівлі | Створити будівлю з нечисловим значенням для кількості поверхів | Повинно бути повідомлення про помилку і запит на введення числового значення. | Виведено повідомлення про помилку. | Пройдено |
| Створення квартири | Створити квартиру в існуючому будинку, заповнивши всі поля (номер квартири, кількість кімнат, площа) | Повинна бути створена нова квартира в обраному будинку. | Створено нову квартиру. | Пройдено |
| Створення квартири | Створити квартиру в неіснуючому будинку | Повинно бути повідомлення про помилку. | Виведено повідомлення про помилку. | Пройдено |
| Створення квартири | Створити квартиру з пустими полями (номер квартири, кількість кімнат, площа) | Повинно бути повідомлення про помилку і запит на введення даних. | Виведено повідомлення про помилку. | Пройдено |
| Створення квартири | Створити квартиру з нечисловим значенням для номера квартири або кількості кімнат | Повинно бути повідомлення про помилку і запит на введення числового значення. | Виведено повідомлення про помилку. | Пройдено |
| Створення квартири | Створити квартиру з від'ємним значенням площі | Повинно бути повідомлення про помилку і запит на введення додатного значення. | Виведено повідомлення про помилку. | Пройдено |
| Перегляд інформації про будівлі | Переглянути список всіх створених будівель | Повинен відобразитись список всіх будівель з їх адресами та кількістю поверхів. | Відображено список всіх будівель. | Пройдено |
| Перегляд інформації про будівлі | Спробувати переглянути список будівель за відсутності створених будівель | Повинно бути повідомлення про відсутність будівель. | Виведено повідомлення про відсутність будівель. | Пройдено |
| Перегляд інформації про квартиру | Обрати існуючу квартиру та переглянути її деталі (номер квартири, кількість кімнат, площа, поверх, список мешканців, орендарів, власників) | Повинна відобразитись детальна інформація про обрану квартиру. | Відображено детальну інформацію про обрану квартиру. | Пройдено |
| Перегляд інформації про квартиру | Спробувати переглянути деталі квартири в неіснуючому будинку | Повинно бути повідомлення про помилку. | Виведено повідомлення про помилку. | Пройдено |
| Редагування інформації про квартиру | Обрати квартиру та змінити її вартість за квадратний метр | Вартість квартири повинна бути оновлена. | Вартість квартири оновлена. | Пройдено |

**2.2.8 Інструкцію користувача**

**Призначення програми**

Дана програма є електронним довідником мешканців та орендаторів багатоквартирного будинку. Вона дозволяє власникам та управлінцям багатоквартирних будинків зручно керувати інформацією про квартири, мешканців, орендарів та витрати на утримання будинку.

**Вимоги до системи**

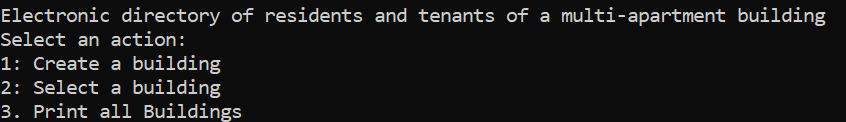
Програма написана на мові програмування C# і працює в середовищі .NET. Для роботи програми необхідно мати встановлену останню версію .NET Runtime.

**3.1 Склад програмного забезпечення**

Програма "ОСББ. Управління будинком" складається з наступних файлів:

* Program.cs - основний виконуваний файл програми
* Buildings.cs - файл зберігання інформації про будівлі
* Apartments.cs - файл зберігання інформації про квартири
* Tenants.cs - файл зберігання інформації про мешканців
* Landlords.cs - файл зберігання інформації про власників квартир
* UtilityExpenses.cs - файл зберігання інформації про комунальні платежі
* RepairExpenses.cs - файл зберігання інформації про витрати на ремонт
* Room.cs – файл зберігання інформації про кімнати

Інтерфейс користувача При запуску програми користувачеві відображається головне меню з наступними опціями:



1. Створити будинок При виборі цієї опції користувач може створити новий будинок, ввівши необхідні дані, такі як адреса та кількість квартир. – рис 1.3

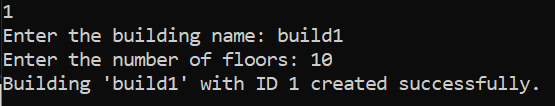


Рис 1.3

1. Вибрати будинок Якщо в системі вже є створені будинки, користувач може вибрати один з них для подальшої роботи з квартирами та мешканцями.
2. Вивести всі будинки Ця опція дозволяє користувачеві переглянути список всіх створених будинків та їх основну інформацію.

Після вибору будинку користувачеві відкривається вікно з меню дій, пов'язаних з обраним будинком: - рис 1.4

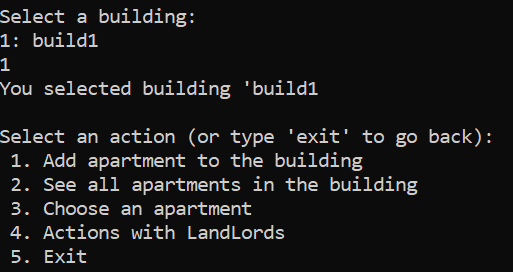


Рис 1.4

1. Додати квартиру Ця опція дозволяє додати нову квартиру до обраного будинку, ввівши необххідні дані, такі як номер квартири та площа. – рис 1.4

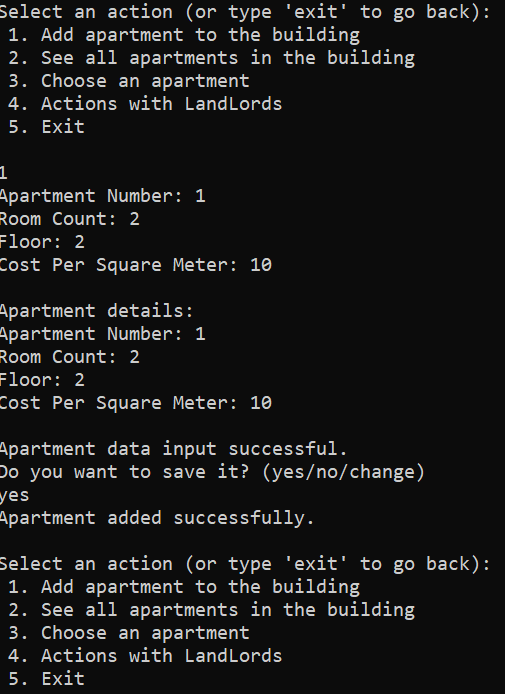


Рис 1.4

1. Переглянути всі квартири За допомогою цієї опції користувач може переглянути список всіх квартир в обраному будинку з їх основною інформацією.
2. Вибрати квартиру При виборі цієї опції користувач може перейти до управління обраною квартирою, де є можливість виконувати різні дії, пов'язані з квартирою, мешканцями та орендарями.
3. Дії з орендодавцями Ця опція надає доступ до функцій, пов'язаних з управлінням орендодавцями в обраному будинку.
4. Вихід Повернення до головного меню.

Після вибору квартири користувачеві відкривається меню з наступними опціями: - рис 1.5

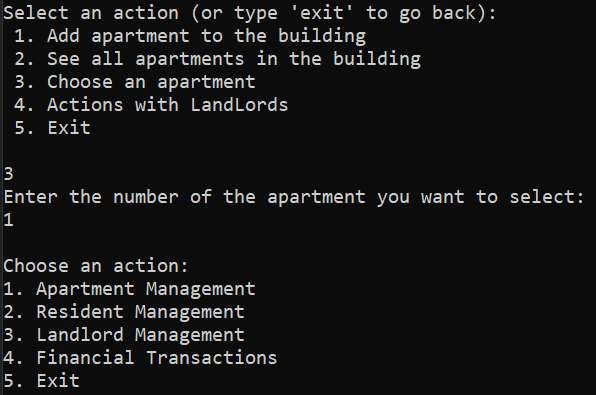


Рис 1.5

1. Управління квартирою

* Оновити вартість квадратного метра
* Додати кімнату
* Переглянути інформацію про квартиру

1. Управління мешканцями

* Зареєструвати нового мешканця
* Оновити інформацію про мешканця
* Переглянути список мешканців
* Видалити мешканця

1. Управління орендодавцями

* Додати орендодавця
* Видалити орендодавця

1. Фінансові операції

* Додати комунальні витрати
* Переглянути інформацію про комунальні витрати
* Розрахувати орендну плату
* Оновити дату оренди
* Додати витрати на ремонт
* Закрити витрати на ремонт
* Переглянути витрати на ремонт

1. Вихід

Опис основних дій в програмі

Створення будинку При виборі опції "Створити будинок" в головному меню користувачеві буде запропоновано ввести назву будинку та кількість поверхів. Після введення даних буде створено новий об'єкт класу Building і доданий до списку будинків в програмі.

Додавання квартири Після вибору будинку і опції "Додати квартиру" користувачеві буде запропоновано ввести номер квартири, кількість кімнат, поверх, на якому розташована квартира, та вартість одного квадратного метра. Після введення даних буде створено новий об'єкт класу Apartment і доданий до списку квартир в обраному будинку.

Управління мешканцями В розділі "Управління мешканцями" користувач може виконувати наступні дії:

* Зареєструвати нового мешканця, ввівши його ім'я, прізвище та інші необхідні дані.
* Оновити інформацію про існуючого мешканця, вибравши його зі списку і змінивши потрібні дані.
* Переглянути список всіх мешканців обраної квартири з їх основною інформацією.
* Видалити мешканця зі списку.

Управління орендодавцями В цьому розділі користувач може додавати нових орендодавців до обраної квартири або видаляти існуючих орендодавців.

Фінансові операції В цьому розділі користувач може виконувати дії, пов'язані з фінансовим управлінням квартири:

* Додавати комунальні витрати, такі як витрати на опалення, воду, електрику тощо за конкретний місяць.
* Переглядати інформацію про комунальні витрати квартири за різні періоди.
* Розраховувати орендну плату за обраний період з урахуванням площі квартири, вартості за квадратний метр, комунальних витрат та термінів оренди.
* Оновлювати дату початку та закінчення строку оренди.
* Додавати витрати на ремонт квартири з описом та сумою.
* Закривати (видаляти) витрати на ремонт після його завершення.
* Переглядати інформацію про усі витрати на ремонт квартири.

Дії з орендодавцями В цьому розділі користувач може виконувати наступні дії, пов'язані з орендодавцями в обраному будинку:

* Переглядати список всіх орендодавців в будинку з інформацією про кількість квартир, які їм належать.
* Переглядати детальну інформацію про обраного орендодавця і його квартири, включаючи дані про мешканців, комунальні витрати та ремонти.
* Розраховувати загальний дохід від усіх квартир, що належать обраному орендодавцю за заданий період.
* Переглядати загальні витрати на ремонт усіх квартир, що належать обраному орендодавцю.

Це основні функції програми та інструкції з їх використання. Програма надає зручний інтерфейс для управління інформацією про багатоквартирні будинки, квартири, мешканців, орендодавців, комунальні та ремонтні витрати, а також розрахунок орендної плати.

**Висновок**

У ході виконання курсової роботи було розроблено програмне забезпечення для управління електронним довідником мешканців та орендаторів багатоквартирних будинків. На етапі аналізу предметної області було визначено основні вимоги та функціональні можливості системи, а також спроєктовано архітектуру додатку.

На етапі проєктування було розроблено структуру класів та їх взаємозв'язки для ефективного моделювання сутностей системи, таких як будинки, квартири, мешканці, орендодавці, комунальні та ремонтні витрати. Використання об'єктно-орієнтованого підходу забезпечило гнучкість та можливість подальшого розширення функціональності додатку.

Під час етапу реалізації було створено консольне додаток на мові C#, що реалізує всі необхідні функції для управління даними багатоквартирних будинків. Використання сучасних технологій та принципів об'єктно-орієнтованого програмування дозволило досягти високої якості та надійності коду.

Результатом роботи є повноцінний програмний продукт, який дозволяє власникам та управлінцям багатоквартирних будинків ефективно керувати інформацією про квартири, мешканців, орендарів, комунальні та ремонтні витрати, а також розраховувати орендну плату. Програма забезпечує зручний інтерфейс для виконання усіх необхідних операцій та має можливість подальшого розширення функціональності.

Таким чином, мета курсової роботи, яка полягала у розробці програмного забезпечення для автоматизації процесів управління багатоквартирними будинками, була успішно досягнута. Розроблене програмне рішення є надійним, ефективним та зручним у використанні, що підтверджується результатами тестування та відповідністю до поставлених вимог.

**ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ**

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Кафедра

Інформатики та програмної інженерії

Затвердив

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 р.

Виконавець:

Студент Ткаченко Костянтин Олександрович

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: <ОСББ.Управліннябудинком>

з дисципліни:

«Основи програмування»

Київ 2024

**ДОДАТОК Г ТЕКСТИ ПРОГРАМНОГО КОДУ**

Тексти програмного коду програмного забезпечення

< ОСББ.Управліннябудинком>

(Найменування програми (документа))

CD-RW

(Вид носія даних)

арк,Кб

(Обсяг програми (документа), арк., Кб)

студента 1 курсу групи ІП-з31

Ткаченко Костянтина Олександровича

Програма складається з декількох класів, кожен з яких відповідає за певну функціональність. Нижче наведено лістинги цих класів з коментарями та поясненнями.

**Building**

using FuncCore.Persons;

namespace FuncCore;

public class Building

{

public string BuildingName { get; set; } // Name of the building

public long NumberOfFloors { get; set; } // Total number of floors in the building

public List<Apartment> Apartments { get; set; } // List of apartments in the building

public List<LandLord> LandLords { get; set; } // List of landlords associated with the building

private Building() // Constructor for Building class

{

Apartments = new List<Apartment>();

LandLords = new List<LandLord>();

}

public void CheckAllRepairExpenses() // Checks all repair expenses

{

RepairExpense.CheckAllRepairExpenses(this);

}

public void CalculateLandlordIncome() // Calculates total income for a landlord

{

// Code omitted for brevity

}

public void FindAllInfoAboutLandLord() // Retrieves all information about a specified landlord

{

// Code omitted for brevity

}

public void PrintAllLandLords() // Prints details of all landlords

{

// Code omitted for brevity

}

public void AddApartment() // Adds a new apartment to the building

{

// Code omitted for brevity

}

public Apartment GetApartmentByNumber() // Retrieves an apartment by its number

{

// Code omitted for brevity

}

public static void CreateBuilding(List<Building> buildings) // Creates a new building and adds it to the list

{

// Code omitted for brevity

}

public static Building? SelectBuilding(List<Building> buildings) // Selects a building from a list

{

// Code omitted for brevity

}

public static void PrintBuildings(List<Building> buildings) // Prints all buildings in the list

{

// Code omitted for brevity

}

private int PromptForInt(string message) // Prompts user for an integer input

{

// Code omitted for brevity

}

private long PromptForLong(string message) // Prompts user for a long integer input

{

// Code omitted for brevity

}

private int PromptForFloor(string message) // Prompts user for a floor number input

{

// Code omitted for brevity

}

private double PromptForDouble(string message) // Prompts user for a double input

{

// Code omitted for brevity

}

private void ChangeApartmentDetail(ref Apartment apartment) // Allows changing details of an existing apartment

{

// Code omitted for brevity

}

}

**RepairExpense**

namespace FuncCore;

public class RepairExpense

{

// Represents the cost of the repair expense

public double Cost { get; set; }

// Represents the description of the repair expense

public string Description { get; set; }

// Method to add a repair expense to an apartment

public static void AddRepairExpense(Apartment apartment)

{

// Code omitted for brevity

}

// Method to close a repair expense for an apartment

public static void CloseRepairExpense(Apartment apartment)

{

// Code omitted for brevity

}

// Method to check all repair expenses for a specific landlord's apartments

public static void CheckAllRepairExpenses(Building building)

{

// Code omitted for brevity

}

// Method to print all repair expenses for an apartment

public static void PrintRepairExpenses(List<RepairExpense> repairExpenses)

{

// Code omitted for brevity

}

}

**Room**

namespace FuncCore;

public class Room

{

// Represents the room number

public int RoomNumber { get; set; }

// Represents the area of the room

public double Area { get; set; }

// Method to add a room to an apartment

public static void AddRoom(Apartment apartment)

{

// Code omitted for brevity

}

}

**UtilityExpense**

using System.Globalization;

namespace FuncCore;

public class UtilityExpense

{

// Represents the cost of rent

public decimal RentCost { get; set; }

// Represents the total utility expenses

public decimal AllUtilityExpenses { get; set; }

// Represents the cost of heating

public decimal HeatingCost { get; set; }

// Represents the cost of water

public decimal WaterCost { get; set; }

// Represents the cost of electricity

public decimal ElectricityCost { get; set; }

// Represents the cost of gas

public decimal GasCost { get; set; }

// Represents the cost of cleaning

public decimal CleaningCost { get; set; }

// Represents the cost of management

public decimal ManagementCost { get; set; }

// Represents the cost of trash removal

public decimal TrashRemovalCost { get; set; }

// Represents the cost of internet, TV, and phone

public decimal InternetTvPhoneCost { get; set; }

// Represents the month for utility expenses

public DateTime UtilityExpensesMonth { get; set; }

// Method to add utility costs to an apartment

public static void AddUtilityCosts(Apartment apartment)

{

// Code omitted for brevity

}

// Method to print utility expenses for an apartment

public static void PrintUtilityExpenses(List<UtilityExpense> utilityExpenses)

{

// Code omitted for brevity

}

}

**LandLord**

namespace FuncCore.Persons;

public class LandLord

{

// Represents the full name of the landlord

public string FullName { get; set; }

// Represents the list of apartments owned by the landlord

public List<Apartment> OwnedApartments { get; set; } = new List<Apartment>();

// Method to add a landlord to an apartment

public static void AddLandlordToApartment(Apartment apartment, Building building)

{

// Code omitted for brevity

}

// Method to remove a landlord from an apartment

public static void RemoveLandlordFromApartment(Apartment apartment, Building building)

{

// Code omitted for brevity

}

}

**Tenant**

namespace FuncCore.Persons;

public class Tenant

{

// Represents the full name of the tenant

public string FullName { get; set; }

// Represents the age of the tenant

public long Age { get; set; }

// Represents the phone number of the tenant

public string PhoneNumber { get; set; }

// Represents the email of the tenant

public string Email { get; set; }

// Represents the emergency contact of the tenant

public string EmergencyContact { get; set; }

// Represents the apartment number of the tenant

public int ApartmentNumber { get; set; }

// Method to register a new tenant for an apartment

public static void RegisterNewTenant(Apartment apartment)

{

// Code omitted for brevity

}

// Method to remove a tenant from an apartment

public static void RemoveTenantFromApartment(Apartment apartment)

{

// Code omitted for brevity

}

// Method to update tenant information

public static void UpdateTenantInformation(Apartment apartment)

{

// Code omitted for brevity

}

// Method to display all tenants of an apartment

public static void DisplayAllTenants(Apartment apartment)

{

// Code omitted for brevity

}

}

**2.1 Реалізація консольного додатку**

* 1. **Захист курсової роботи**