Міністерство освіти на науки України

Львівський Національний Університет Природокористування

|  |  |  |
| --- | --- | --- |
| Факультет |  | Кафедра |
| механіки, енергетики та ІТ |  | Інформаційних технологій |

***КУРСОВА РОБОТА***

з дисципліни: ***Обʼєктно-орієнтоване програмування***

**на тему:** Розробка системи обліку роздрібної торгівлі мінеральними добривами

Студент гр. КН-31 Герасименко Олександр Юрійович \_\_\_\_\_\_« » 2023.

(ПІБ, підпис, дата)

Керівник:\_\_\_\_\_\_\_\_\_к.т.н.,доцент Татомир А.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наук. ступінь, ПІБ)

Зміст

[Реферат 1](#_Toc151825132)

[Вступ 2](#_Toc151825133)

[1. АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ТА ОБҐРУНТУВАННЯ ЗАВДАННЯ НА РОБОТУ 3](#_Toc151825134)

[1.1. Аналіз предмету проектування 3](#_Toc151825135)

[1.2. Класифікація об’єктно-орієнтованих мов програмування 4](#_Toc151825136)

[1.3. Огляд та аналіз сучасних технологій та засобів проектування програмного забезпечення 4](#_Toc151825137)

[1.4. Універсальна мова проектування UML 5](#_Toc151825138)

[1.5. Уточнена постановка задачі на розробку програмного забезпечення 6](#_Toc151825139)

[2. РОЗРОБКА ЗАГАЛЬНОЇ СТРУКТУРИ ПРОГРАМИ 8](#_Toc151825140)

[2.1. Аналіз функцій системи 8](#_Toc151825141)

[2.2. Розроблення структурної моделі системи 9](#_Toc151825142)

[3. РОЗРОБЛЕННЯ ІЄРАРХІЇ КЛАСІВ 10](#_Toc151825143)

[3.1. Декомпозиція системи 10](#_Toc151825144)

[3.2. Аналіз взаємозв’язків між об’єктами 11](#_Toc151825145)

[3.3. Розроблення інтерфейсів класів 12](#_Toc151825146)

[3.4. UML-діаграма класів 13](#_Toc151825147)

[4. РОЗРОБКА СИСТЕМИ ВВОДУ-ВИВОДУ ДАНИХ 14](#_Toc151825148)

[5. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 17](#_Toc151825149)

[6. РОЗРОБКА ДОКУМЕНТАЦІЇ ДЛЯ СУПРОВОДЖЕННЯ ПРОГРАМНОГО ПРОДУКТУ 18](#_Toc151825150)

[Висновки 19](#_Toc151825151)

[Бібліографічний список 20](#_Toc151825152)

[Додаток А 21](#_Toc151825153)

[Додаток B 22](#_Toc151825154)

[Додаток C 23](#_Toc151825155)

УДК 004.9 : 631.1

Розробка системи обліку роздрібної торгівлі мінеральними добривами

Герасименко О.Ю. Кафедра IT – Дубляни, Львівський НУП, 2023

Курсова робота: 23 сторінок, 5 рисунків, 10 джерел.

Виконано аналіз предмету та сучасних засобів проектування. Виконана уточнена постановка задачі на розробку програмного забезпечення.

Розроблена структура моделі системи. Виконана декомпозиція, розробка інтерфейсу класів та побудована UML діаграма. Наведено на рис. 3.1.

Розроблена система вводу-виводу. Наведено на рис 4.1, рис 4.2.

Проведено тестування.

Розроблена документація для супроводження продукту.

# Вступ

У світі, де сільське господарство відіграє ключову роль у забезпеченні продовольчої безпеки та економічного розвитку, ефективна система обліку та управління стає вирішальним фактором для досягнення успіху в аграрній галузі. Серед важливих компонентів цього процесу займає важливе місце роздрібна торгівля мінеральними добривами.

Під час розробки системи обліку роздрібної торгівлі мінеральними добривами враховуються ключові аспекти управління складськими запасами, контроль за реалізацією та забезпечення якості цих продуктів. Така система не лише спростить процеси управління, але й сприятиме підвищенню ефективності сільськогосподарського виробництва.

У цій роботі розглядається, як створити такий додаток, використовуючи потужні мову програмування Python. Як вести облік в цьому додатку та в загальному як взаємодіяти з ним.

Даний проект націлено на аналіз і розробку ефективних інструментів, які допоможуть удосконалити та автоматизувати процеси обліку, що стане вагомим внеском у сферу роздрібної торгівлі мінеральними добривами та сприятиме покращенню ефективності у цій галузі.

# 1. АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ТА ОБҐРУНТУВАННЯ ЗАВДАННЯ НА РОБОТУ

Аналіз сучасного стану питання та обґрунтування завдання на роботу є важливим кроком у процесі розробки будь-якого проекту.

Мета цього проекту полягає в створенні інструмента, який дозволить користувачам вести облік торгівлі мінеральними добривами.

Обране завдання включає створення програми для додавання інформації. про надходження та продажі товару. Вона надасть користувачам можливість переглядати історію замовлень та термін придатності товару.

## 1.1. Аналіз предмету проектування

* *Визначення мети проекту*: мета цього проекту дати можливість користувачу вести облік торгівлі в електронному вигляді.
* *Визначення користувачів та їх потреб*: Користувачами може бути обмежена кількість так як функціонал орієнтований на конкретні потреби користувача.
* *Вивчення конкурентів та існуючих рішень*: Проведемо аналіз існуючих додатків або сервісів, які вже пропонують схожі функції. Визначимо їх переваги та недоліки, щоб зрозуміти, як ми можемо відзначитися серед конкурентів.
* *Технічний аналіз*: Програма написана повністю на Python.
* *Оцінка ризиків*: Визначимо можливі ризики, які можуть виникнути під час реалізації проекту, і розробимо план їх управління.

## 1.2. Класифікація об’єктно-орієнтованих мов програмування

Об'єктно-орієнтовані мови програмування можна класифікувати за декількома критеріями.

* *За спадкоємністю*
* *За способом доступу до даних та методів*
* *За механізмами поліморфізму*:
* *За типом інкапсуляції*:

## 1.3. Огляд та аналіз сучасних технологій та засобів проектування програмного забезпечення

Такий аналіз дозволяє зрозуміти які технології забезпечують очікуваний результат та ефективність

* Мови програмування: Огляд і аналіз мов програмування, які можуть бути використані для реалізації проекту. Розглядаються основні характеристики, продуктивність, підтримка спільноти та додаткові бібліотеки.
* Фреймворки: Аналіз фреймворків які спрощують розробку програмного забезпечення та забезпечують готові рішення.
* Інструменти для розробки: Аналіз різних інструментів для розробки ПЗ. Наприклад: Visual studio code або PyCharm.
* Методології розробки: Аналіз підходів для організації процесу розробки. Наприклад: Scrum, Kanban, DevOps.
* Бази даних та системи керування базами даних: Аналіз різних типів БД і СУБД. Наприклад, MySQL, PostgreSQL, MongoDB, PHPmyAdmin).

## 1.4. Універсальна мова проектування UML

UML (Unified Modeling Language) є стандартною універсальною мовою моделювання, призначеною для візуалізації, специфікацій, конструювання та документування програмних систем. Основні елементи та властивості UML включають:

* *Стандартизація*: UML розроблена Інститутом інженерів програмного забезпечення (SEI) та є міжнародним стандартом, який підтримується Об'єднаною міжнародною асоціацією з інформатики та обчислювальної техніки (ACM) та Інститутом інженерів електротехніки та електроніки (IEEE).
* *Діаграми класів*: Показують структуру системи, класи, їх атрибути, методи та взаємодії.
* *Мультипарадигмальність*: UML підтримує різні аспекти розробки програмного забезпечення, включаючи ООП.
* *Діаграми взаємодії*: Включають діаграми послідовності та діаграми співпраці для візуалізації взаємодії між об’єктами.
* *Спрощення проектування*: UML спрощує процес проектування, дозволяючи моделювати систему на високому рівні та виділяти ключові компоненти.

## 1.5. Уточнена постановка задачі на розробку програмного забезпечення

**Назва проекту**: Розробка системи обліку роздрібної торгівлі мінеральними добривами.

**Опис проблеми або потреби**: Проект спрямований на створення програмного забезпечення, яке надає користувачам можливість зберігати та переглядати інформацію про мінеральні добрива в торгівлі.

**Мета проекту**:

* Розробити ПЗ, яке дозволить користувачам добавляти інформацію про надходження мінеральних добрив, добавляти та переглядати інформацію про замовлення.
* Забезпечити інтерактивний інтерфейс для вибору дій та переміщення меню.
* Забезпечити можливість відображення терміну дій та можливість видалення простроченого товару

**Вимоги до функціональності**:

* можливість додавання партій мінеральних добрив різних видів та їх продажу.
* Можливість виводити інформацію про залишки та обсяги продажу.
* Врахувати обмежений термін зберігання добрив

**Нефункціональні вимоги**:

* Коректна робота ПЗ на операційних системах, що підтримуються.
* Висока продуктивність та оптимальне споживання ресурсів.
* Врахування помилок вводу

**Список обмежень**:

* Обмежений бюджет для розробки проекту.
* Терміни виконання проекту відповідно до графіку.

**Очікувані результати**:

* Готове ПЗ, яке дозволяє переглядати історію продаж та редагування інформації про товар.

# 2. РОЗРОБКА ЗАГАЛЬНОЇ СТРУКТУРИ ПРОГРАМИ

Розробка загальної структури програми - це процес проектування основних складових частин програмного забезпечення, який визначає загальну архітектуру та організацію програми до початку реалізації деталей. Цей етап включає в себе аналіз функцій системи та розроблення структурної моделі системи.

## 2.1. Аналіз функцій системи

Аналіз функцій дозволяє визначити, які функції повинні бути реалізовані в ПЗ:

1. **Функція вводу інформації**: Користувачі повинні мати можливість вводити інформацію для переміщення в меню та внесення змін.
2. **Врахування помилок вводу**: Програмний продукт повинен надавати враховувати помилковий ввід інформації від користувача та повідомляти про це.
3. **Доступність головних функцій**: Користувач повинен мати можливість повернутись до головного меню на різних етапах роботи з ПЗ.

Такий аналіз допомагає створити план розробки та реалізацію конкретних функцій.

## 2.2. Розроблення структурної моделі системи

Розроблення структурної моделі системи - це процес створення абстрактної моделі, яка відображає ключові елементи системи та їх взаємозв'язки. Ця модель допомагає зрозуміти організацію та структуру системи перед фазою реалізації.

Основні етапи розроблення структурної моделі системи включають:

1. **Ідентифікація елементів системи:** Це включає визначення всіх компонентів, які складають систему. Це можуть бути функції, процеси, модулі, компоненти програмного забезпечення або будь-які інші основні частини системи.
2. **Створення діаграм:** Зазвичай використовуються структурні діаграми, такі як діаграми блоків або ієрархічні діаграми, щоб візуалізувати структуру системи та взаємозв'язки між її елементами.
3. **Визначення зв'язків та взаємозв'язків:** Це включає встановлення взаємозв'язків між різними елементами системи. Наприклад, які елементи взаємодіють між собою, які компоненти залежать один від одного, тощо.

Структурна модель допомагає розробникам та іншим учасникам проекту краще розуміти систему як цілісний об'єкт, а не просто набір окремих елементів. Вона дозволяє визначити загальну структуру та організацію системи, що полегшує подальший процес реалізації, тестування та підтримки.

# 3. РОЗРОБЛЕННЯ ІЄРАРХІЇ КЛАСІВ

Розробка ієрархії класів - це процес створення ієрархії об'єктно-орієнтованої моделі, де класи організовані за спадковістю та взаємозв'язками між ними. Цей дозволяє створювати логічні групи класів, які спадково пов'язані між собою, забезпечуючи спадкування властивостей та методів.

## 3.1. Декомпозиція системи

Декомпозиція системи - це процес поділу складної системи на менші, більш прості та керовані частини або модулі. Цей процес дозволяє розбити складні завдання на менші, більш керовані і керовані частини, що полегшує розуміння, розробку та управління системою.

Декомпозиція системи:

* + **Модуль головного меню main\_menu()**: Відповідає за виклик функцій в залежності від дій користувача.
  + **Модуль замовлення order\_edit()**: Відповідає за корегування залишку товару та створення історії замовлення.
  + **Модуль для добавлення партії товару batch\_edit()**: Відповідає за корегування кількості товару.
  + **Модуль для відображення інформації про добрива obj\_data()**: Відповідає за виведення інформації про добрива та партії товару.

Декомпозиція системи використовується для спрощення складних завдань, полегшення розробки та тестування, поліпшення модульності та можливості розвитку системи. Вона дозволяє команді розробників концентруватися на конкретних аспектах системи, сприяючи більш ефективній та систематизованій роботі над проектом.

## 3.2. Аналіз взаємозв’язків між об’єктами

Аналіз взаємозв'язків між об'єктами є процесом визначення та розуміння зв'язків між різними об'єктами в об'єктно-орієнтованому програмуванні. Об'єкти в програмуванні взаємодіють між собою, обмінюються інформацією та виконують дії через свої взаємозв'язки:

* + Клас *Nitrogen\_Ferlitizer, Phosphorus\_ Ferlitizer* та *Potassium\_ Ferlitizer* можуть використовувати клас *types\_of\_mineral\_ferlitizers*  для взаємодії з масивами даних та іншими змінними.
  + Об’єкти класів *Nitrogen\_Ferlitizer, Phosphorus\_ Ferlitizer* та *Potassium\_ Ferlitizer мають унікальний масив який створюється в конструкторі головного класа та спільний метод для отримання типу добрива*.

Аналіз взаємозв'язків між об'єктами є допомагає розуміти структуру та логіку взаємодії між різними частинами програми, що сприяє більш ефективній та зрозумілій розробці програмного забезпечення

## 3.3. Розроблення інтерфейсів класів

Розроблення інтерфейсів класів - це процес визначення публічної частини класу, яка визначає методи та їх сигнатури, доступні для використання іншими класами або об'єктами. Інтерфейс визначає, як клас взаємодіє з іншими елементами програми, надаючи зовнішнє уявлення про його функціональність.

1. **Інтерфейс класу *types\_of\_mineral\_ferlitizers*:**
   * Змінна *self.type\_name* зберігає тип добрива якому воно належить.
   * Зміннаself\_type\_index зберігає порядковий номер типу добрива.
   * Змінна *storage* зберігає кількість цього добрива розділеного на партії
   * Змінна *overall\_amount* зберігає загальну кількість добрива в об’єкті
   * Змінна *storage\_date* зберігає дату добавлення партії в storage
   * Змінна *status* є пустим при оголошенні і заповнюється значеннями True або False. Якщо термін придатності закінчився в масив записується True в іншому випадку False
   * Метод *Show\_type()* повертає значення *self.type\_name*
   * Змінна *shelf\_life* зберігає термін придатності цього добрива.
   * Змінна *name* зберігає назву добрива
2. **Інтерфейс класу Nitrogen/Phosphorus/Potassium\_Ferlitizers**:
   * Поле super().\_\_init\_\_ *()* викликає конструктор головного класу.
   * Інші поля присвоюють змінним значення.

Розроблення інтерфейсів класів дозволяє створювати чітко визначені контракти між класами, що полегшує спільне використання та розширення коду. Це також сприяє зменшенню залежності між класами, оскільки зовнішній код взаємодіє з класом через його інтерфейс, а не безпосередньо через його реалізацію.

## 3.4. UML-діаграма класів

UML (Unified Modeling Language) - це мова для візуального моделювання програмного забезпечення, включаючи створення діаграм класів для подання структури та взаємозв'язків між класами в програмі. Наведено на рис. 3.1

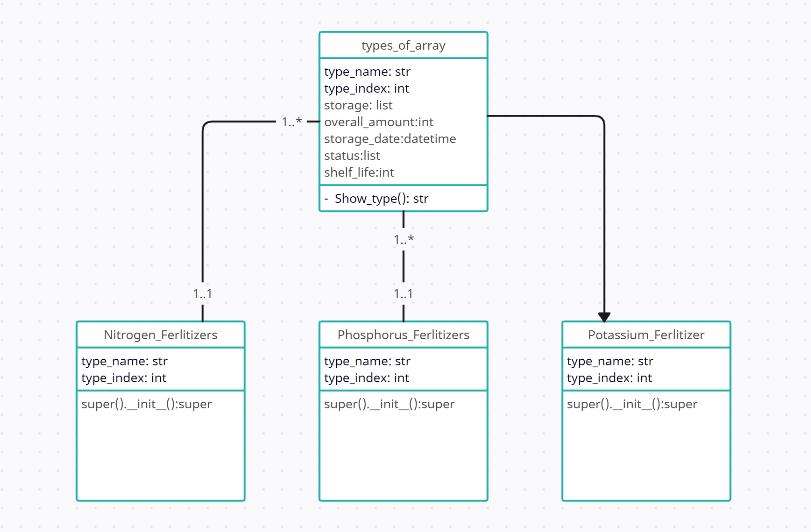


Рис. 3.1 UML таблиця

Така UML таблиця дозволяє швидко узагальнити ключові елементи системи, такі як класи, їх атрибути та методи, а також відносини між ними.

# 4. РОЗРОБКА СИСТЕМИ ВВОДУ-ВИВОДУ ДАНИХ

Розробка системи вводу-виводу даних - це процес створення та налагодження механізмів, які дозволяють програмі взаємодіяти з користувачем або зовнішніми пристроями для зчитування та запису даних.

В даному проекті було використана система вводу-виводу терміналу VS code.

Код програми знаходиться в «[*Додаток А*](#_Додаток_А)».

Основні функції програми знаходяться в «[Додаток B](#_Додаток_B)» та «[Додаток C](#_Додаток_C)»

Результати наведено на рис. 4.1, рис 4.2

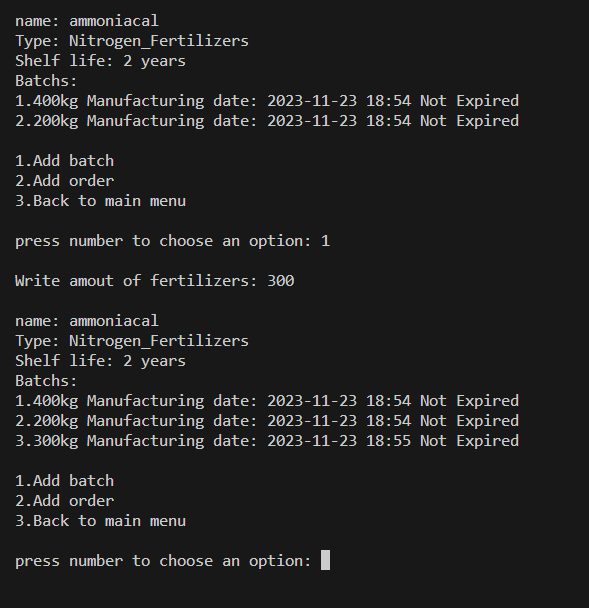


Рис. 4.1 Добавлення партії

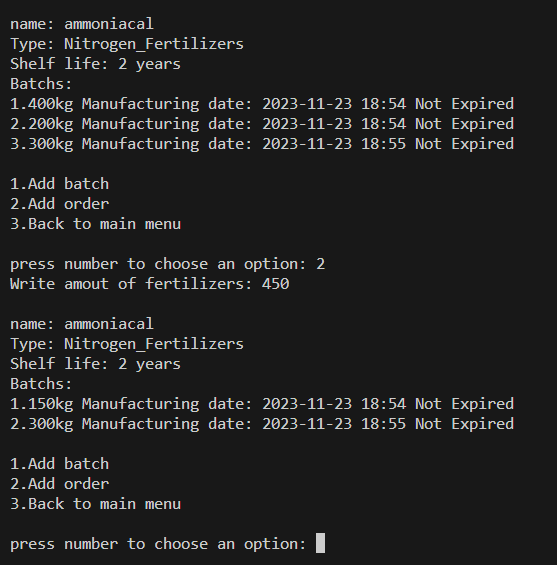


Рис. 4.2 Добавлення інформації про замовлення

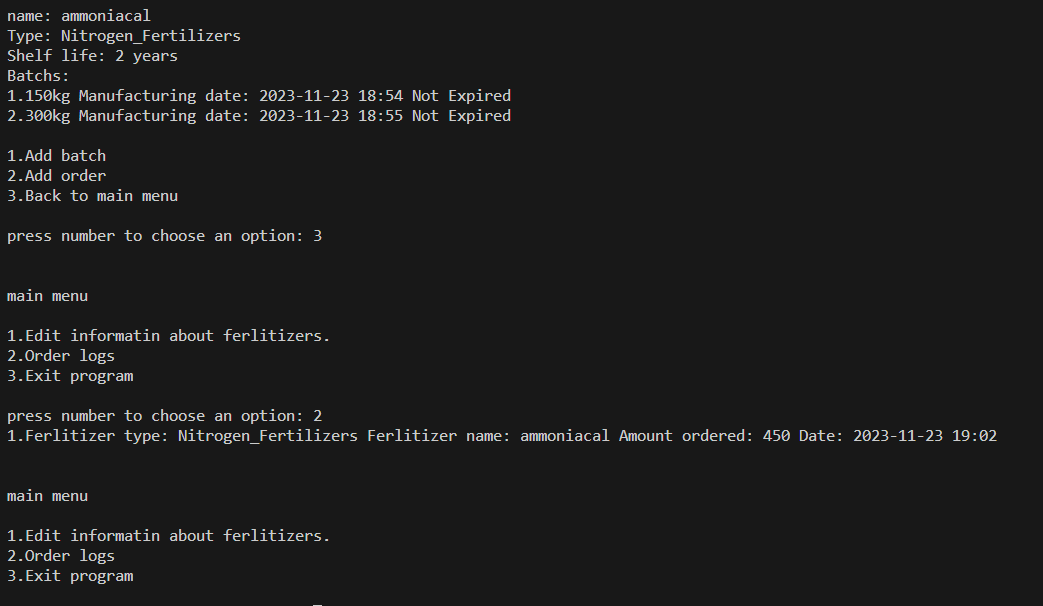


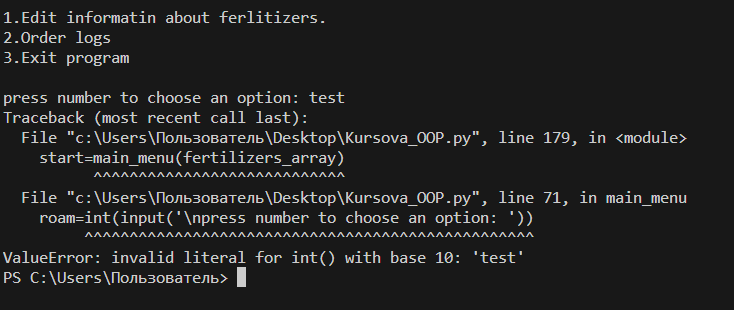
Рис. 4.2 Виведення інформації про замовлення

Це зручний інструмент для розробників, оскільки дозволяє виконувати різноманітні завдання без виходу з інтерфейсу редактора. Також, ви можете налаштовувати термінал у VS Code, використовуючи різні оболонки (shell) та розширення, щоб забезпечити зручність та ефективність вашої роботи.

# 5. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення - це процес перевірки програми або системи на відповідність специфікаціям, очікуванням користувача та вимогам, а також виявлення помилок або недоліків у функціональності.

Даний проект є достатньо малого об’єму що дозволило проводити тестування шляхом введення різних даних та порівняння результату з очікуваним.



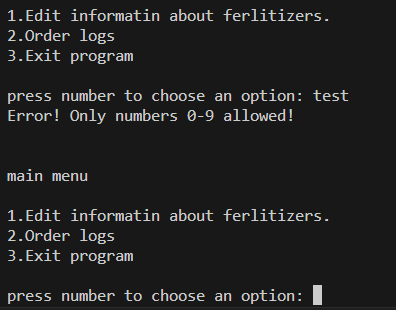
Рис. 5.1 Помилка під час тестування

Рис. 5.2 Виправлення помилки

Тестування дозволило виявити та виправити помилки, що посприяло коректній роботі програми та наявності повноцінного функціоналу.

# 6. РОЗРОБКА ДОКУМЕНТАЦІЇ ДЛЯ СУПРОВОДЖЕННЯ ПРОГРАМНОГО ПРОДУКТУ

Для запуску програму потрібно виконати декілька кроків:

1. **Інсталяція Python:** Якщо Python не встановлено на вашому комп'ютері, ви можете завантажити його з [офіційного сайту Python](https://www.python.org/) та встановити.
2. **Редактор коду:** Використовуйте текстовий редактор або інтегроване середовище розробки (IDE), таке як PyCharm, VS Code, або Jupyter Notebook, для редагування та запуску коду.
3. **Запуск програми:** Запуститіть файл. Або відкрийте ваш редактор коду, скопіюйте вихідний код програми до файлу з розширенням .py, збережіть файл. Потім виконайте цей файл за допомогою інтерпретатора Python.

Така документація дозволяє новим користувачам легко запустити програму.

# Висновки

Під час виконання роботи був проаналізований предмет проектування, огляд сучасних засобів проектування. Була вивчена та застосована мова проектування. Було розроблена загальна структура програми та ієрархія класів

Також була проведена декомпозиція системи та аналіз взаємозв'язків між об'єктами. Завдяки тестування було виявлені та виправленні помилки.

# Бібліографічний список

1. "Object-Oriented Programming in Python" авторства Michael H. Goldwasser та David Letscher
2. Шведов Д. В. "Універсальна мова проектування UML. Посібник для практичних занять" - Київ: Видавництво Київського університету, 2020. - 288 с.
3. Умаров Р. Ш., Васильев А. В. "Розробка сучасних додатків на Python" - Київ: Пітер, 2018. - 416 с.
4. "Learning Python Design Patterns" авторства Gennadiy Zlobin
5. "Python 3 Object-Oriented Programming" авторства Dusty Phillips
6. Fowler M., Beck K., Brant J., Opdyke W., Roberts D. "Рефакторинг. Покращення структури і архітектури коду" - Київ: Діалектика, 2019. - 480 с.
7. Хант Э., Томас Дж. "Персональне програмування на Python" - Київ: Видавництво Манн, Іванов і Фербер, 2021. - 432 с.
8. Realpython.com
9. Python.org
10. Tutorialspoint.com

# Додаток А

Повний код знаходиться за посиланням <https://github.com/Inai0/Kursova.git>

# Додаток B

main\_menu(fertilizers\_array):

    print ("main menu\n\n1.Edit informatin about ferlitizers.\n2.Order logs\n3.Exit program")

    roam=input('\npress number to choose an option: ')

    if not re.match("^[0-9]\*$", roam):

        print ("Error! Only numbers 0-9 allowed!")

        return True

    roam=int(roam)

    if roam==1:

        return (ferlitizers\_edit(fertilizers\_array))

    if roam==2:

        if len(logs)==0:

            print ("No logs")

            return True

        for i in range(len(logs)):

            print ("Orders:")

            print (str(i+1)+'.'+logs[i])

    if roam==3:return clear\_terminal()

    else:return True

def obj\_data(fertilizers\_array,choosen\_obj\_id):

    #Prints information about ferlitize

    print('\nname: ' + fertilizers\_array[choosen\_obj\_id].name + '\nType: '+fertilizers\_array[choosen\_obj\_id].Show\_type()+'\nShelf life: ' +str(fertilizers\_array[choosen\_obj\_id].shelf\_life) +' years\nBatchs: ')

    j=1

    for i in range(len(fertilizers\_array[choosen\_obj\_id].storage)):

        print(str(j)+"."+str(fertilizers\_array[choosen\_obj\_id].storage[i])+'kg Manufacturing date: '+str(fertilizers\_array[choosen\_obj\_id].storage\_date[i].strftime("%Y-%m-%d %H:%M"))+check\_status(fertilizers\_array,i,choosen\_obj\_id))

        j+=1

# Додаток C

def order\_edit(choosen\_obj\_id,fertilizers\_array):

    #deleting ordered amout from ferliteze's batches

    if len(fertilizers\_array[choosen\_obj\_id].storage)>0:

        while True:

            amount=input("Write amout of fertilizers: ")

            if not re.match("^[0-9]\*$", amount):

                print ("Error! Only numbers 0-9 allowed!")

            else: break

        amount=int(amount)

        dif=amount

    else:

        print("No fertilizers!")

        return fertilizers\_array

    fertilizers\_array[choosen\_obj\_id].overall\_amount=0

    for i in fertilizers\_array[choosen\_obj\_id].storage:

        fertilizers\_array[choosen\_obj\_id].overall\_amount+=i

    if amount>fertilizers\_array[choosen\_obj\_id].overall\_amount:

        print("\nNo enought fertilizers!")

        return fertilizers\_array

    logs.append('Ferlitizer type: '+fertilizers\_array[choosen\_obj\_id].Show\_type()+' Ferlitizer name: '+fertilizers\_array[choosen\_obj\_id].name+' Amount ordered: '+str(amount)+' Date: '+datetime.now().strftime("%Y-%m-%d %H:%M"))

    for i in range(len(fertilizers\_array[choosen\_obj\_id].storage)):

        if dif<=fertilizers\_array[choosen\_obj\_id].storage[i]:

            if dif==fertilizers\_array[choosen\_obj\_id].storage[i]:

                fertilizers\_array[choosen\_obj\_id].storage.pop(i)

                break

            fertilizers\_array[choosen\_obj\_id].storage[i]-=dif

            break

        else: dif-=fertilizers\_array[choosen\_obj\_id].storage[i]

    for j in range(i):

        fertilizers\_array[choosen\_obj\_id].storage.pop(0)

        fertilizers\_array[choosen\_obj\_id].storage\_date.pop(0)

    return fertilizers\_array