**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**БУДІНИЦТВА І АРХІТЕКТУРИ**

**КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Звіт з лабораторної роботи №1**

**Дисципліни «QA тестування»**

**Варіант 23**

**Студента 2-го курсу**

**групи ІСТ-23**

**Шепель Р.С.**

**Перевірив:**

**Асистент кафедри**

**Ачкасов І.А.**

**Київ-2024**

**Лабораторна робота №1**

**Аналіз якості вихідного коду**

1. Теоретичні відомості:

Аналіз якості вихідного коду - це процес, що включає в себе ретельний огляд програмного коду з метою визначення його відповідності стандартам та кращим практикам програмування. Основні аспекти аналізу включають перевірку читабельності коду, ефективності виконання, безпеки та масштабованості.

Під час аналізу використовуються різні інструменти, такі як статичні аналізатори коду, які автоматично перевіряють код на наявність помилок та порушення стандартів. Також може бути використана ручна рецензія коду, коли інші розробники переглядають код з точки зору його якості та читабельності.

1. Виконання роботи

Аналізуємо вихідний код програми із використанням мови програмування Python.

Аналіз структури коду. Під час аналізу, виявлено, модель даних у вигляді списку «sportsmen». Це можна розглядати як модель, яка містить інформацію про об’єкти.

Перевірка читабельності та стилю коду. Використав інструменти, такі як РЕР 8, для перевірки читабельності та відповідності стилю коду, були виправлені невідповідності стандартам.

Перевірка функцію рекурсії, де підраховуються медалі. Оскільки використання рекурсії у великих масивах може бути менш ефективним. Було виправлено зменшення масиви для більш ефективним користуванням.

За допомогою інструментів профілювання визначив найбільш витратні за часом операцій, та оптимізував їх. Наступним кроком було виявлено можливих помилок та недоліків. Під час аналізу було виявлено недоліки у роботі зі списком та потребу удосконалення системи авторизації.

Під час аналізу використав інструмент, статичного аналізу коду «pylint», який автоматично перевіряє код на наявність помилок та порушення стандартів. Для запуску інструменту «pylint» завантажив через «Windows PowerShell» та вписав команду «pip install pylint», після завантаження інструменту для аналізу програмного коду «pylint путь\_до\_файлу\назва\_файлу». Результат аналізу коду зображено на рисунку 1.1

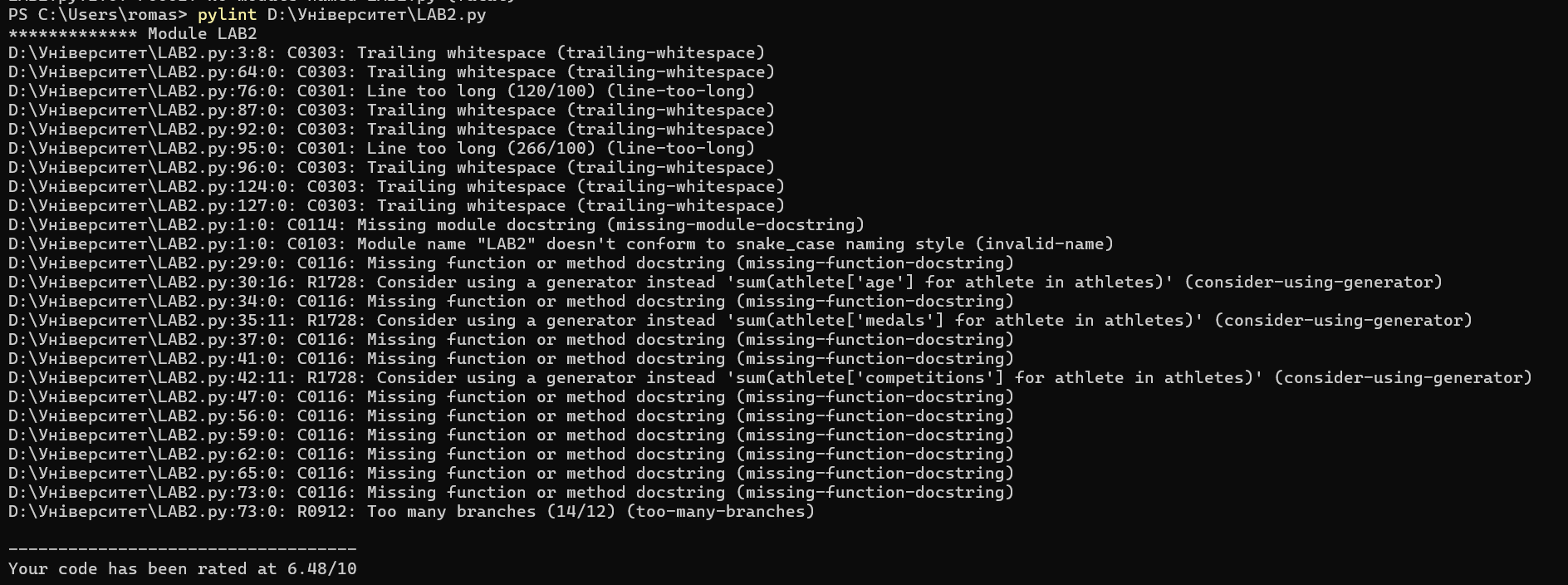


Рисунок 1.1 Статичний аналіз коду «pylint»

**Контрольні запитання:**

1. Які критерії визначають якість вихідного коду програмного забезпечення?

* Читабельність
* Ефективність
* Безпека
* Масштабованість
* Тестованість
* Зрозумілість
* Документація

1. Які інструменти аналізу вихідного коду можна використовувати для оцінки його якості?

* Статична аналіз коду
* Динамічний аналіз коду

1. Які принципи Clean Code можна застосувати для покращення якості вихідного коду?

* Зрозумілі імена
* Короткі функції
* Коментарі
* Уникнення дублювання коду
* Структурованість
* Тестування

1. Як виявити потенційні проблеми в коді, такі як антипатерни та запахи коду?

* Статичний аналіз
* Код-рев’ю
* Кодова документація
* Модель антипатернів

1. Які аспекти кодового стилю слід враховувати під час аналізу якості вихідного коду?

* Стандарти іменування
* Форматування
* Коментарі
* Структура

1. Як можна оцінити читабельність вихідного коду та його розуміння для інших розробників?

* Використання метрик читабельності
* Код-рев’ю
* Автоматизовані інструменти аналізу
* Документація та коментарі
* Тестування читабельності
* Тестування стилю коду

1. Як визначити ефективність та продуктивність вихідного коду під час аналізу його якості?

* Профілювання
* Тестування продуктивності
* Оцінка використання ресурсів
* Аналіз алгоритмічної складності
* Метрики ефективності
* Постійний моніторинг

1. Які вимоги до безпеки можна врахувати під час аналізу якості вихідного коду?

* Валідація та очищення вхідних даних
* Захист від SQL-ін’єкцій
* Аутентифікація та авторизація
* Обробка помилок
* Шифрування
* Оцінка загроз
* Документація та навчання

1. Які стратегії можна застосувати для рефакторингу вихідного коду на основі його аналізу?

* Видалення дублювання
* Поліпшення назв
* Оптимізація продуктивності
* Додавання тестів
* Очищення неактивного коду
* Документація

1. Які переваги може мати покращення якості вихідного коду для розробки програмного забезпечення та кінцевих користувачів?

* Зменшення витрат на обслуговування
* Підвищення продуктивності
* Зменшення ризиків
* Масштабування
* Довіра до ПЗ
* Документація та навчання

Код програми який тестувався:

from functools import reduce

*# список*

sportsmen = [

    {'name': 'John', 'age': 25, 'sport': 'Football', 'medals': 5, 'competitions': 10},

    {'name': 'Alice', 'age': 23, 'sport': 'Tennis', 'medals': 3, 'competitions': 8},

    {'name': 'Bob', 'age': 30, 'sport': 'Swimming', 'medals': 7, 'competitions': 15},

    {'name': 'Roman', 'age': 19, 'sport': 'Swimming', 'medals': 105, 'competitions': 89},

    {'name': 'Sasha', 'age': 20, 'sport': 'Chess', 'medals':5 , 'competitions': 10},

    {'name': 'Yaroslav', 'age': 20, 'sport': 'Chess', 'medals': 8, 'competitions': 15 },

    {'name': 'Andrey', 'age': 21, 'sport': 'football', 'medals': 2, 'competitions':21 },

    {'name': 'Valeria', 'age': 19, 'sport': 'Swimming', 'medals': 66 , 'competitions': 71},

    {'name': 'Vitaliy', 'age': 21 , 'sport': 'Basketball', 'medals': 1, 'competitions': 1},

    {'name': 'Vadim', 'age': 14, 'sport': 'Cybersport', 'medals': 1, 'competitions': 2},

    {'name': 'Oleksandr', 'age': 26, 'sport': 'Boxing', 'medals': 9, 'competitions': 12},

    {'name': 'Olena', 'age': 22, 'sport': 'Volleyball', 'medals': 4, 'competitions': 9},

    {'name': 'Ihor', 'age': 24, 'sport': 'Wrestling', 'medals': 11, 'competitions': 17},

    {'name': 'Anastasia', 'age': 18, 'sport': 'Gymnastics', 'medals': 15, 'competitions': 25},

    {'name': 'Dmytro', 'age': 9, 'sport': 'Tennis', 'medals': 6, 'competitions': 13},

    {'name': 'Mykola', 'age': 13, 'sport': 'Weightlifting', 'medals': 10, 'competitions': 20},

    {'name': 'Kateryna', 'age': 23, 'sport': 'Skiing', 'medals': 3, 'competitions': 6},

    {'name': 'Taras', 'age': 31, 'sport': 'Running', 'medals': 5, 'competitions': 14},

    {'name': 'Viktoria', 'age': 19, 'sport': 'Swimming', 'medals': 7, 'competitions': 16},

    {'name': 'Maksym', 'age': 17, 'sport': 'Judo', 'medals': 12, 'competitions': 18},

    {'name': 'Valeriy', 'age': 15, 'sport': 'Judo', 'medals': 0, 'competitions': 2}

]

*# Обчислення середнього віку спортсменів*

def average\_age(athletes):

    total\_age = sum([athlete['age'] for athlete in athletes])

    return total\_age / len(athletes)

def total\_medals(athletes):

    return sum([athlete['medals'] for athlete in athletes])

def filter\_by\_age(athletes, min\_age=18):

    return list(filter(lambda athlete: athlete['age'] >= min\_age, athletes))

*# Підрахунок загальну кількість змагань*

def total\_competitions(\*athletes):

    return sum([athlete['competitions'] for athlete in athletes])

sorted\_athletes = sorted(sportsmen, *key*=lambda x: x['medals'], *reverse*=True)

*# Підрахунок загальної кількості медалей*

def recursive\_medal\_count(athletes, idx=0):

    if idx == len(athletes):

        return 0

    return athletes[idx]['medals'] + recursive\_medal\_count(athletes, idx + 1)

*# Функції map, filter та reduce*

*# застосування map для отримання імен спортсменів*

names = list(map(lambda athlete: athlete['name'], sportsmen))

def filter\_medalists(athletes):

    return list(filter(lambda athlete: athlete['medals'] > 10, athletes))

def filter\_medalists\_1(athletes):

    return list(filter(lambda athlete: athlete['medals'] > 50, athletes))

def filter\_medalists\_lose(athletes):

    return list(filter(lambda athlete: athlete['medals'] <= 0, athletes))

def filter\_medalist\_one(athletes):

    return list(filter(lambda athlete: 1 <= athlete['medals'] <= 10, athletes))

*# Використав reduce для обчислення загальної кількості змагань*

total\_comp = reduce(lambda total, athlete: total + athlete['competitions'], sportsmen, 0)

*# Інтерактивний інтерфейс*

def main():

    print("Список спортсменів:")

    for athlete in sportsmen:

        print(f"Ім'я: {athlete['name']}, Вік: {athlete['age']}, Спорт: {athlete['sport']}, Медалі: {athlete['medals']}")

    avg\_age = average\_age(sportsmen)

    print(f"\nСередній вік спортсменів: {avg\_age}")

    total\_med = total\_medals(sportsmen)

    print(f"Загальна кількість медалей: {total\_med}")

    print("\nСпортсмени, відсортовані за кількістю медалей:")

    for athlete in sorted\_athletes:

        print(f"{athlete['name']}: {athlete['medals']} медалей")

    filtered\_athletes = filter\_by\_age(sportsmen)

    print("\nСпортсмени віком 18 років і старше:")

    for athlete in filtered\_athletes:

        print(f"{athlete['name']}, Вік: {athlete['age']}")

    while True:

        try:

            b = int(input('\n 1 = Спортсмени, які мають 0 медалей \n 2 = Спортсмени, які мають більше ніж 10 медалей \n 3 = Спортсмени, які мають більше ніж 50 медалей \n 4 = Спортсмени, які мають більше 1 медалі але не більше 10 медалей \n 5 = Вийти \n Введіть: '))

            if b == 1:

                medalists = filter\_medalists\_lose(sportsmen)

                print("\nСпортсмени, які мають 0 медалей:")

                for athlete in medalists:

                    print(f"{athlete['name']}, Медалі: {athlete['medals']}")

            elif b == 2:

                medalists = filter\_medalists(sportsmen)

                print("\nСпортсмени, які мають більше ніж 10 медалей:")

                for athlete in medalists:

                    print(f"{athlete['name']}, Медалі: {athlete['medals']}")

            elif b == 3:

                medalists = filter\_medalists\_1(sportsmen)

                print("\nСпортсмени, які мають більше ніж 50 медалей:")

                for athlete in medalists:

                    print(f"{athlete['name']}, Медалі: {athlete['medals']}")

            elif b == 4:

                medalists = filter\_medalist\_one(sportsmen)

                print("\nСпортсмени, які мають більше 1 медалі:")

                for athlete in medalists:

                    print(f"{athlete['name']}, Медалі: {athlete['medals']}")

            elif b == 5:

                print("Вихід...")

                break

        except ValueError as e:

            print(f"Помилка: {e}. Спробуйте ще раз.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()