МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Комп’ютерне моделювання процесів та систем»

Ефективність та якість архітектурних рішень

**Звіт  
Лабораторна робота №2**

Довгий список параметрів. Надлишкові елементарні типи

Виконав ст.гр. ІКМ-М223д

Ковальов Д.Д.

Перевірив викладач

Хорошун А.С.

Харків 2024

Завдання

1. Виконати рефакторинг функції, зменшивши кількість параметрів та полегшивши розуміння та обслуговування коду.
2. Виконати рефакторинг цієї функції, зменшивши кількість параметрів та полегшивши розуміння та обслуговування коду

1.

def calculate\_score(name, age, gender, height, weight, score1, score2, score3, score4, score5):

    # Бізнес-логіка для розрахунку загального рейтингу

    total\_score = score1 + score2 + score3 + score4 + score5

    # Перевірка чи користувач є повнолітнім

    is\_adult = True if age >= 18 else False

    # Виведення результатів

    print("Name:", name)

    print("Age:", age)

    print("Gender:", gender)

    print("Height:", height)

    print("Weight:", weight)

    print("Total Score:", total\_score)

    print("Adult:", is\_adult)

# Приклад виклику функції

calculate\_score("John", 25, "Male", 175, 70, 85, 90, 75, 88, 92)

Результат рефакторингу:

def calculate\_score(data):

    name=data.get('name', '')

    age=data.get('age', 0)

    gender=data.get('gender', '')

    height=data.get('height', 0)

    weight=data.get('weight', 0)

    scores=data.get('scores', [])

    total\_score=sum(scores)

    is\_adult=True if age>=18 else False

    print("Name:", name)

    print("Age:", age)

    print("Gender:", gender)

    print("Height:", height)

    print("Weight:", weight)

    print("Total Score:", total\_score)

    print("Adult:", is\_adult)

data = {

    'name':"John",

    'age':25,

    'gender':"Male",

    'height':175,

    'weight':70,

    'scores':[85, 90, 75, 88, 92]

}

calculate\_score(data)

Замість того щоби передавати групу даних, отриманих з

іншого об’єкта в якості параметрів, можна передати в метод

сам об’єкт, використовуючи передачу всього об’єкта.Так і було вчинено, усі дані згруповано та передано у функцію як одну функцію.

2.

class User:

    type\_engineer = 1

    type\_manager = 2

    def \_\_init\_\_(self, name, age, type, phone, phone\_code):

        self.name = name

        self.age = age

        self.type = type

        self.phone = phone

        self.phone\_code = phone\_code

    def print\_details(self):

        # Виведення інформації про користувача

        print("Name:", self.name)

        print("Age:", self.age)

        print("Type:", self.type)

        print("Phone:", self.phone\_code + self.phone)

# Приклад використання класу

user = User("John", 25, User.type\_engineer, "9379992", “050”)

user.print\_details()

Результат:

class User:

    def \_\_init\_\_(self, name, age, type, phone, phone\_code):

        self.name = name

        self.age = age

        self.type = get\_type(type)

        self.phone = PhoneNumber(phone, phone\_code)

    def print\_details(self):

        print("Name:", self.name)

        print("Age:", self.age)

        print("Type:", self.type)

        print("Phone:", self.phone.full\_number())

class UserType:

    type\_engineer = 1

    type\_manager  = 2

    def get\_type(user\_type):

        if user\_type == UserType.type\_engineer:

            return "Engineer"

        elif user\_type == UserType.type\_manager:

            return "Manager"

        else:

            return "Unknown"

class PhoneNumber:

    def \_\_init\_\_(self, number, code):

        self.number = number

        self.code = code

    def full\_number(self):

        return self.code + self.number

# Приклад використання класу

user = User("John", 25, UserType.type\_engineer, "9379992", "050")

user.print\_details()

Тут було створено два нових окремих класи, перший для номеру телефону, другий для типу працівника. Перший відокремлює та групує примітивні дані, що поєднуються за сенсом, що зменшує кількість параметрів конструктора, а другий – зменшує кількість полів основного класу.