Звіт з практичної роботи No15

Тема роботи: Основи об'єктно-орієнтованого програмування: проектування та

створення класів у Python.

Виконав: Кравченко Сергій, ТЦР-12

Дата: 02.06.2025

Мета роботи:

Навчитися створювати класи, визначати їх властивості та методи,

використовувати конструктори для ініціалізації об'єктів.

Хід роботи:

Постановка завдання:

Завдання 1. "Клас Людина"

● Створіть клас Person, який має властивості: ім'я, вік, місто

проживання.

● Додайте метод для виводу короткої інформації про людину.

Завдання 2. "Клас Автомобіль"

● Реалізуйте клас Car, який має властивості: марка, модель, рік

випуску, колір.

● Додайте метод для виводу повної інформації про автомобіль.

● Додайте метод для зміни кольору автомобіля.

Завдання 3. "Клас Банк"

● Створіть клас BankAccount, який містить ім'я власника, номер

рахунку, баланс.

● Додайте методи для поповнення рахунку, зняття коштів та перевірки

балансу.

● Додайте обробку винятків для перевірки достатності коштів на

рахунку.

Код програми:

class Person:

def init(self, name, age, city):

self.name = name

self.age = age

self.city = city

def display\_info(self):

print(f"Ім'я: {self.name}, Вік: {self.age}, Місто: {self.city}")

class Car:

def init(self, brand, model, year, color):

self.brand = brand

self.model = model

self.year = year

self.color = color

def display\_info(self):

print(f"Автомобіль: {self.brand} {self.model}, Рік: {self.year}, Колір: {self.color}")

def change\_color(self, new\_color):

self.color = new\_color

print(f"Колір змінено на: {self.color}")

class BankAccount:

def init(self, owner\_name, account\_number, balance=0):

self.owner\_name = owner\_name

self.account\_number = account\_number

self.balance = balance

def deposit(self, amount):

if amount > 0:

self.balance += amount

print(f"Рахунок поповнено на {amount}. Поточний баланс: {self.balance}")

else:

print("Сума поповнення має бути більше 0")

def withdraw(self, amount):

try:

if amount > self.balance:

raise ValueError("Недостатньо коштів на рахунку")

if amount <= 0:

raise ValueError("Сума зняття має бути більше 0")

self.balance -= amount

print(f"Знято {amount}. Поточний баланс: {self.balance}")

except ValueError as e:

print(f"Помилка: {e}")

def check\_balance(self):

print(f"Поточний баланс: {self.balance}")

if name == "main":

print("=== Приклад використання класу Person ===")

person1 = Person("Іван Петренко", 30, "Київ")

person1.display\_info()

print("\n=== Приклад використання класу Car ===")

car1 = Car("Toyota", "Camry", 2020, "чорний")

car1.display\_info()

car1.change\_color("білий")

car1.display\_info()

print("\n=== Приклад використання класу BankAccount ===")

account1 = BankAccount("Іван Петренко", "UA123456789", 1000)

account1.check\_balance()

account1.deposit(500)

account1.withdraw(200)

account1.withdraw(2000)

account1.withdraw(-100)

account1.check\_balance()

Приклад роботи програми:

=== Приклад використання класу Person ===

Ім'я: Іван Петренко, Вік: 30, Місто: Київ

=== Приклад використання класу Car ===

Автомобіль: Toyota Camry, Рік: 2020, Колір: чорний

Колір змінено на: білий

Автомобіль: Toyota Camry, Рік: 2020, Колір: білий

=== Приклад використання класу BankAccount ===

Поточний баланс: 1000

Рахунок поповнено на 500. Поточний баланс: 1500

Знято 200. Поточний баланс: 1300

Помилка: Недостатньо коштів на рахунку

Помилка: Сума зняття має бути більше 0

Поточний баланс: 1300

Висновки

У цій практичній роботі було реалізовано три класи (Person, Car, BankAccount), які демонструють основи об'єктно-орієнтованого програмування в Python. Кожен клас містить властивості та методи для роботи з об'єктами, включаючи ініціалізацію, вивід інформації та модифікацію даних. Особливу увагу приділено обробці винятків у класі BankAccount для коректної роботи з фінансовими операціями. Код є зрозумілим, модульним і готовим до подальшого розширення функціоналу.