Лабораторна робота №3

Звіт

З дисципліни “Об'єктно-орієнтоване програмування ”

на тему: “Абстрактні класи ”.

Студента 3 курсу: Групи МІТ-31 Демиденко Андрій

**Київ - 2024р.**

**Комплексний звіт**

**Мета:** ознайомитись з поняттям абстрактного класу, навчитись створювати абстрактні класи у Пайтон.

**Завдання для самостійного опрацювання варіант**

**Варіант №6**

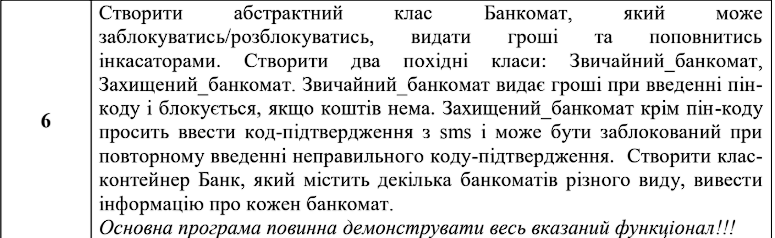


Рис. 3.1 – завдання для варіанту

**Завдання №1**

Для початку визначимо основні компоненти системи та класи, на яких будується архітектура банкоматної інфраструктури (рис.3.2). Базовий клас **ATM** є абстрактним — у ньому описано загальні риси банкоматів. Його метод **withdraw()** оголошено абстрактним, тобто обов’язковим для реалізації у спадкоємцях.

Класи-нащадки:

* **RegularATM** — реалізує звичайну поведінку зняття коштів з банкомата;
* **SecureATM** — розширює **RegularATM**, додаючи SMS-підтвердження;
* **Bank** — окремий клас, який агрегує банкомати, ілюструючи приклад агрегації.

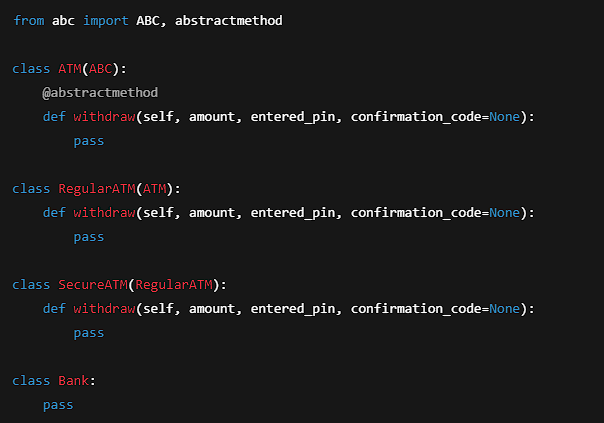


Рис. 3.2 – Загальна структура класів та абстрактного успадкування

## Опис основних методів класу ATM (рис. 3.3):

* **\_\_init\_\_** — ініціалізує атрибути банкомата: ім’я, PIN-код, баланс, статус;
* **deposit(amount)** — поповнення рахунку;
* **check\_balance()** — повертає поточний баланс користувача;
* **get\_status()** — виводить активний стан або блокування банкомата;
* **\_\_str\_\_()** — повертає структуровану текстову інформацію про банкомат;
* **withdraw()** — абстрактний метод, обов’язковий до реалізації у спадкоємцях.

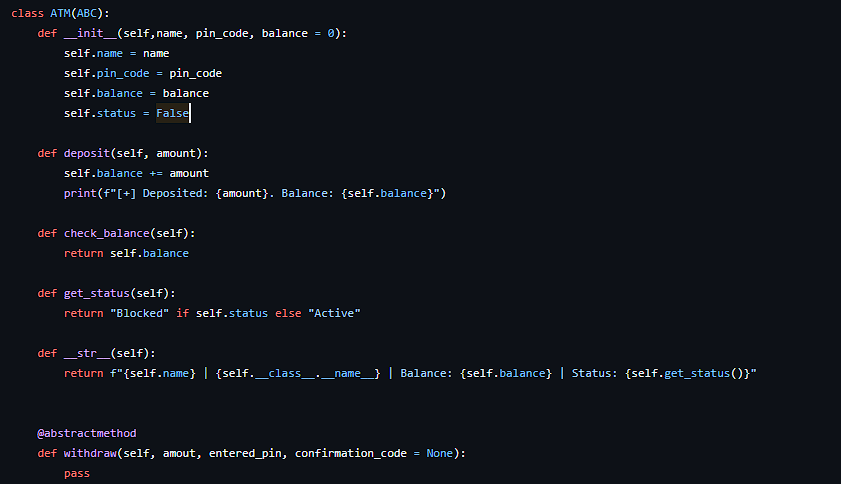


Рис. 3.3 – Основні методи класу ATM.

## Наслідування та особливості класів

### Клас RegularATM (рис. 3.4):

RegularATM реалізує стандартну логіку зняття коштів. Метод withdraw(...) перевіряє:

* чи **не заблоковано банкомат** (self.status);
* чи **введено правильний PIN-код**;
* чи **достатньо коштів на рахунку.**

У разі помилки (наприклад, недостатньо балансу) банкомат блокується. Це симулює базовий рівень безпеки.

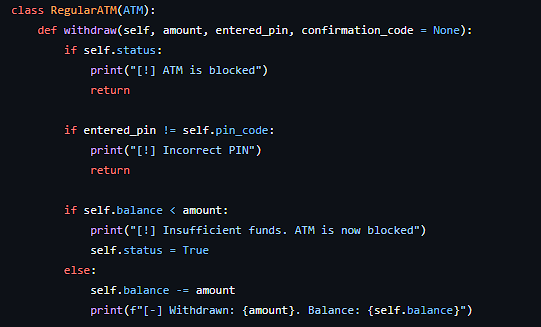


Рис. 3.4 – Клас **RegularATM.**

### Клас SecureATM (рис. 3.5)

SecureATM — розширена версія RegularATM, яка додає:

* **двофакторну аутентифікацію** через SMS-код;
* **ліміт спроб** на введення неправильного SMS-коду (2 спроби);
* **автоматичне блокування** банкомата після перевищення кількості спроб.

Метод withdraw(...) у цьому класі додатково перевіряє:

1. PIN-код (як у RegularATM);
2. SMS-код (confirmation\_code);
3. Ведеться підрахунок кількості невдалих спроб (failed\_attempts);
4. Якщо баланс недостатній — також блокується.

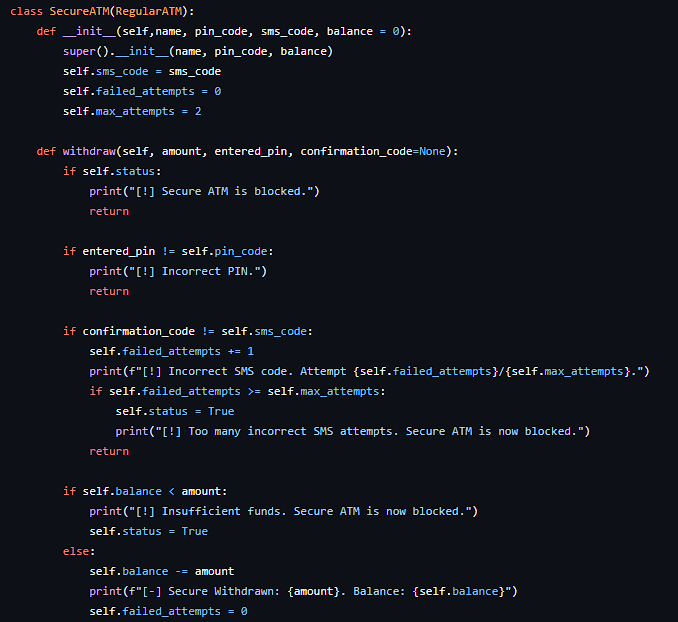
****

Рис. 3.5 – Клас **SecureATM.**

### Клас Bank (рис. 3.6)

Клас Bank виконує функцію **менеджера банкоматів**. Він:

* зберігає банкомати у словнику self.atms;
* дозволяє:
  + **додавати нові банкомати** (add\_atm);
  + **видаляти банкомати** (remove\_atm);
* реалізує ітерацію по банкоматах (\_\_iter\_\_), що дозволяє легко проходити по всіх пристроях у банку.

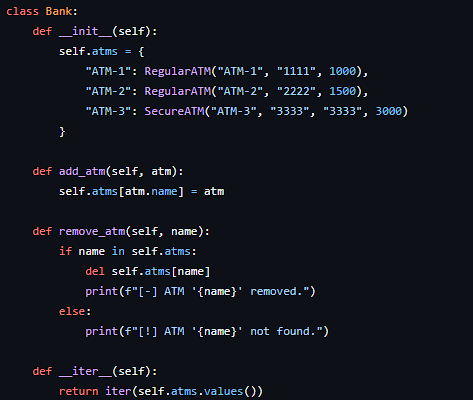


Рис. 3.6 – Клас **Bank**

Основна частина програми, тобто "**main**", виконує роль демонстрації роботи всієї системи банкоматів. Вона слугує точкою входу, де ініціалізується об’єкт **Bank**, а далі поетапно виконуються приклади взаємодії з банкоматами: перегляд, поповнення, зняття коштів, обробка помилкових введень, додавання й видалення пристроїв.

Цей блок ілюструє, як класи співпрацюють між собою, і перевіряє логіку, закладену в абстрактному класі та його підкласах. Він написаний просто та послідовно — це зручно для тестування поведінки програми й розуміння, як функціонують основні компоненти. Такий підхід дозволяє не тільки перевірити правильність реалізації, а й надалі легко розширити або адаптувати систему до нових вимог **(рис. 3.7).**

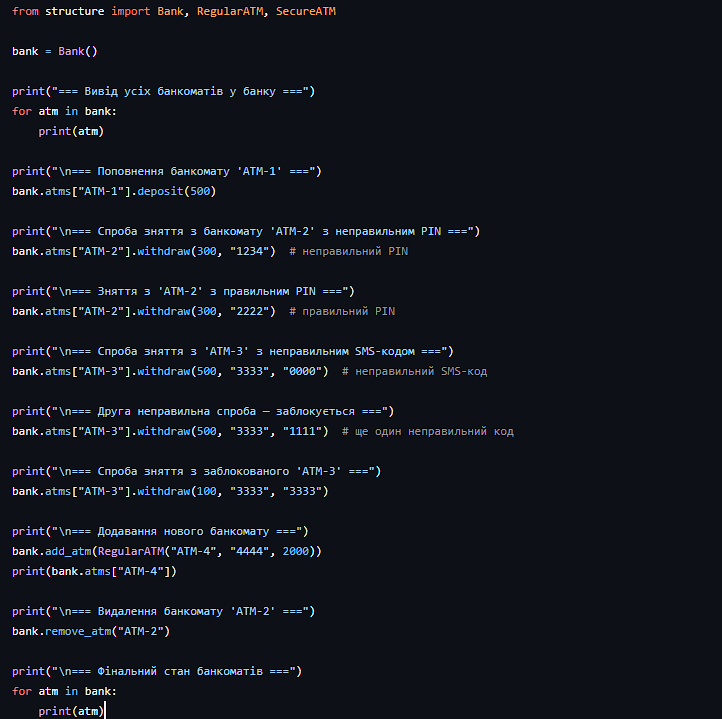


Рис. 3.7 – Основна частина програми **main.**

У цьому прикладі реалізовано об'єктно-орієнтовану модель банкоматної системи із застосуванням абстрактного класу **ATM**. Під час виконання програми створюється об'єкт класу Bank, який містить кілька банкоматів різного типу — звичайні (**RegularATM**) та захищені (**SecureATM**). На початку програми відбувається вивід усіх банкоматів, що зберігаються в банку. Завдяки реалізації методу ітерації у класі **Bank**, можна зручно обійти колекцію банкоматів і вивести кожен з них у вигляді рядка, що надає короткий опис банкомата, його тип, баланс і статус (активний або заблокований).

Далі демонструється поповнення рахунку одного з банкоматів — це здійснюється методом **deposit**, який є спільним для всіх типів банкоматів і реалізований у базовому класі **ATM**. Після цього відбувається спроба зняття коштів із неправильним PIN-кодом. У цьому випадку банкомат не дозволяє операцію й повідомляє про помилку без зміни балансу. При правильному PIN-коді та наявності достатнього балансу кошти успішно знімаються, а баланс оновлюється.

Особливий акцент зроблено на функціональності захищеного банкомата **SecureATM**, який крім перевірки PIN-коду також вимагає підтвердження через SMS-код. У разі неправильного введення коду система фіксує кількість помилкових спроб, і при досягненні ліміту банкомат блокується. Таким чином, реалізовано додатковий рівень безпеки.

Також показано динамічну зміну складу банку — додається новий банкомат, який одразу доступний у системі, та видаляється один із наявних. Це свідчить про те, що система легко розширюється та підтримує модифікацію під час роботи.

Загалом використання абстрактного класу **ATM** забезпечує чітке визначення інтерфейсу всіх банкоматів у системі. Абстрактний метод **withdraw** змушує всі підкласи реалізувати власну логіку зняття коштів, яка може бути адаптована до вимог безпеки, як-от у випадку **SecureATM**. Це дозволяє зберігати узгодженість, розширюваність та інкапсуляцію в системі. В результаті ми отримуємо гнучку архітектуру, яка підтримує як прості, так і складні банкомати, і водночас гарантує, що кожен з них реалізує базову необхідну функціональність, визначену в абстрактному класі **(рис. 3.8).**

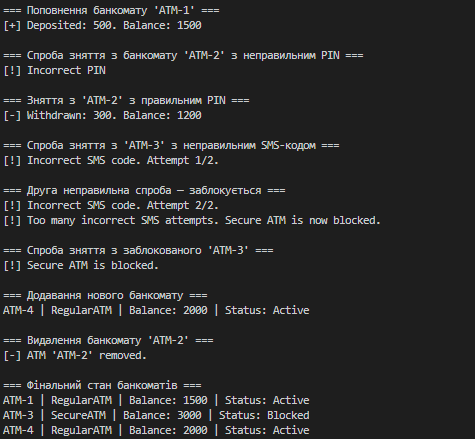


Рис. 3.8 – вивід у консоль

**Висновок:** Під час виконання лабораторної роботи було засвоєно ключове поняття абстрактного класу як шаблону для створення ієрархії класів. Реалізація банкоматів на основі абстрактного класу дозволила забезпечити повторне використання коду, гнучкість та безпеку розширення. Навички, здобуті під час роботи, є основою для побудови масштабованих об'єктно-орієнтованих систем у Python.

[Git Reposytory](https://github.com/AndrewDem0/OOP-Learning)