Звіт

З лабораторної роботи №1

Студента групи МІТ-31

Добровольського Арсенія Михайловича

Варіант №9

**Тема роботи:** Знайомство з класами та об’єктами у Python.

**Мета роботи:** ознайомитися з парадигмою об’єктно-орієнтованого програмування, розглянути поняття класу та об’єкта, навчитися створювати класи та об’єкти в Python.

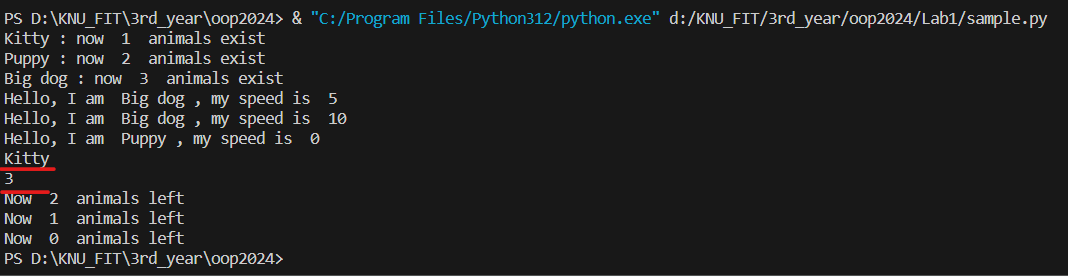
Програма-зразок, наведена у лабораторній роботі



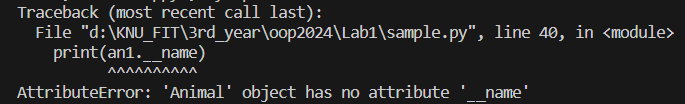
У цьому коді є два магічних методи:

1. **\_\_init\_\_**: Цей метод викликається автоматично при створенні нового об'єкта класу **Animal**. Він ініціалізує новий об'єкт з вказаними параметрами, такими як ім'я та швидкість, та збільшує лічильник **counter**, який відслідковує кількість створених екземплярів класу.
2. **\_\_del\_\_**: Цей метод викликається автоматично при видаленні об'єкта класу **Animal**. Він зменшує лічильник **counter**, який відслідковує кількість залишених екземплярів класу.

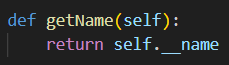
Намагаючись вивести ім’я першої тваринки та лічильник, отримали необхідні дані (підкреслені червоним):

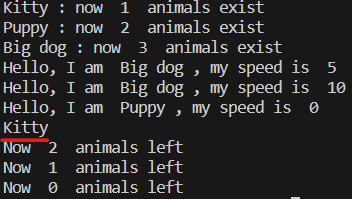


Приховавши змінні **name** і **counter** за допомогою двох підкреслень «\_\_», при виконанні коду виникає помилка:



Це пояснюється тим, що закриті поля і методи доступні лише в межах класу, відповідно немає можливості звертатися до них просто за назвою, як це було раніше. Цю помилку можна виправити, додавши геттер **getName()** – спеціальний метод, який дозволить отримати ім’я прихованого атрибуту name. При цьому змінна **counter** залишається доступною лише в межах класу.





Відображення параметрів екземпляра класу **an1**:



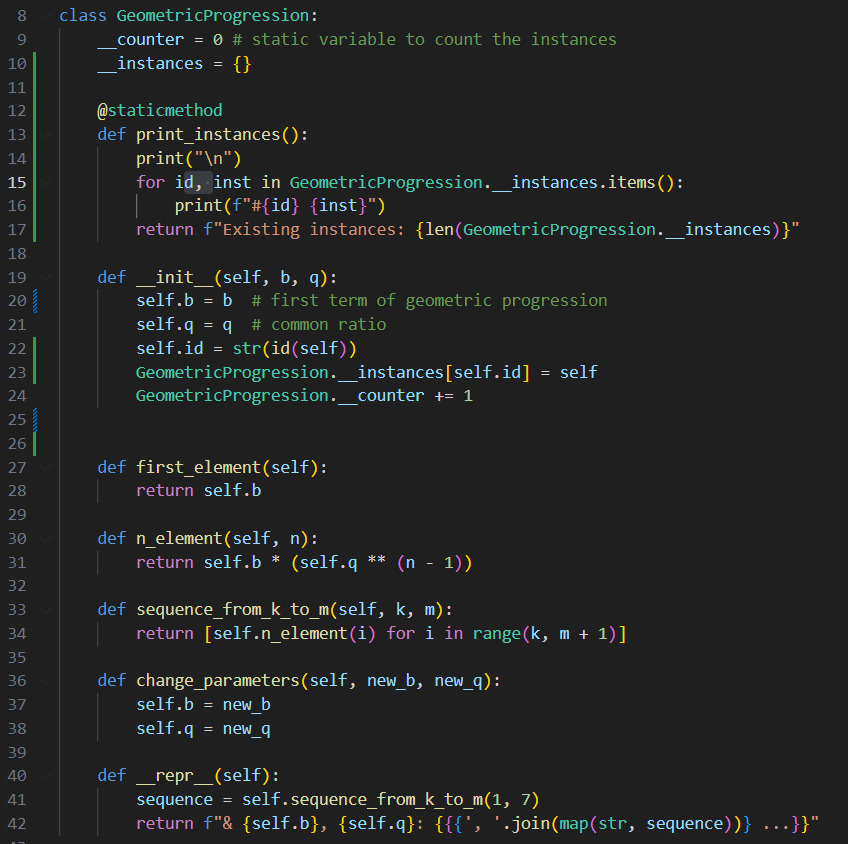
Оскільки поле **name** нині приватне, Python змінив його на **\_Animal\_\_name**, де **Animal** – ім’я класу. Спробуємо звернутися до нього за цим іменем:



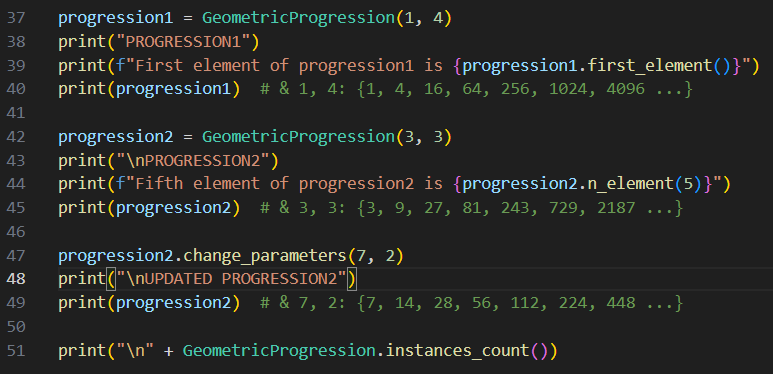


Завдання №1

Опис класу **GeometricProgression**:

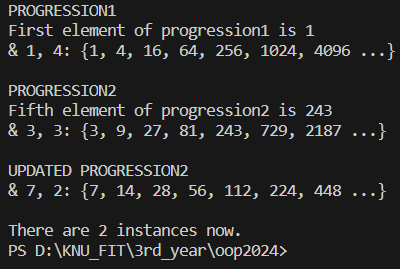


Створення екземплярів класу та робота з ними:



Цей код відображає опис класу **GeometricProgression**, який дозволяє генерувати геометричні прогресії залежно від вхідних даних. Клас має статичну приховану змінну **\_\_counter**, яка використовується для підрахунку кількості створених екземплярів класу. Конструктор класу **\_\_init\_\_** приймає значення першого члена прогресії b та знаменник q. Методи **first\_element**, **n\_element**, та **sequence\_from\_k\_to\_m** відповідають за повернення значення першого члена прогресії, обчислення n-го елемента та генерування послідовності від k-го до m-го елемента геометричної прогресії відповідно. Метод **change\_parameters** дозволяє змінити параметри геометричної прогресії, тобто задати нові значення **b** і **q**. Врешті-решт, метод **\_\_repr\_\_** повертає текстове представлення об’єкту, а статичний метод **instances\_count** відображає кількість існуючих екземплярів класу в даний момент часу.

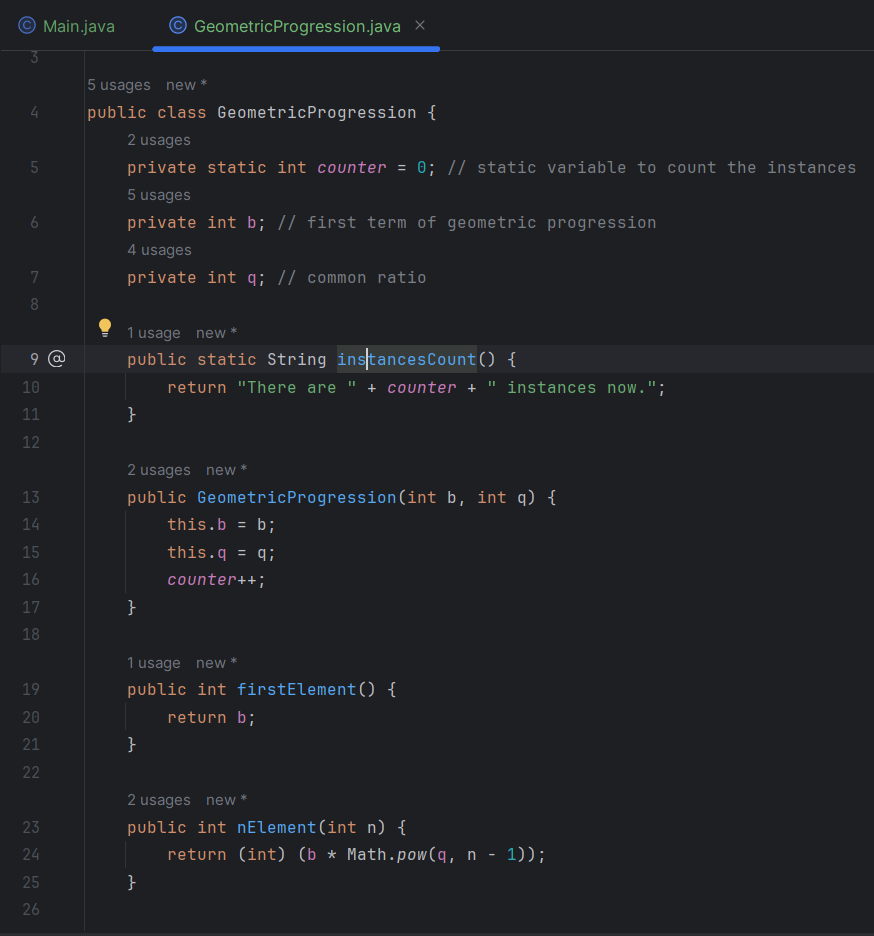
Далі створюються дві геометричні прогресії. В першій за допомогою методу **first\_element** відображається її перший член та виводиться власне прогресія. В другій відображається вже її п'ятий член (метод **n\_element**) і знову сама прогресія. Потім в другій прогресії змінюються значення параметрів **b** і **q**, завдяки чому отримуємо абсолютно нову послідовність, яка перезаписала попередню у **progression2**. Насамкінець, відбувається звертання до методу **instances\_count**, який виводить кількість наявних геометричних прогресій (оскільки даний метод статичний, звертаємося до нього через ім’я класу).



Завдання №2

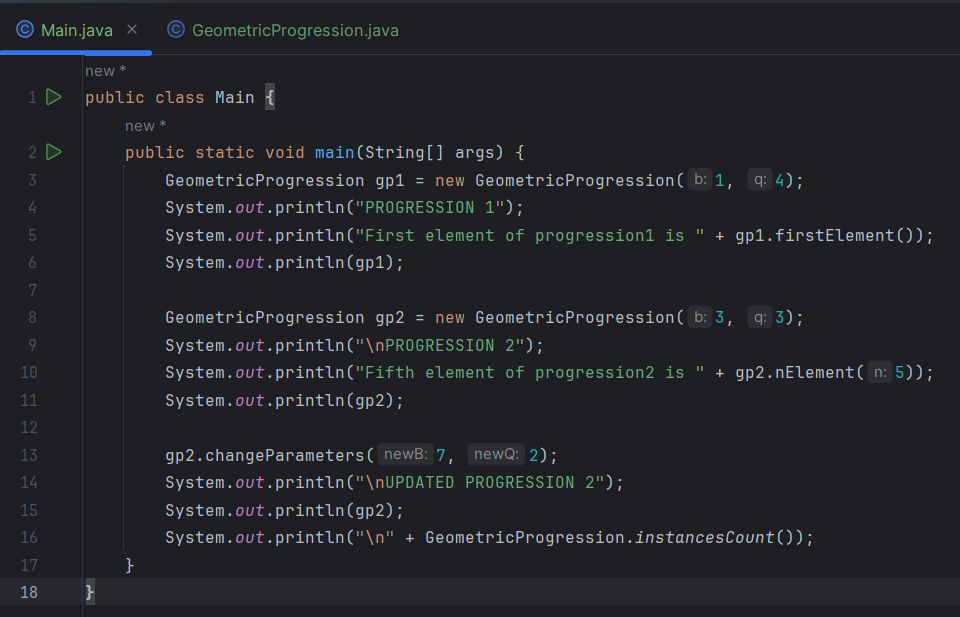
Реалізуємо ту саму задачу, використовуючи мову програмування Java:

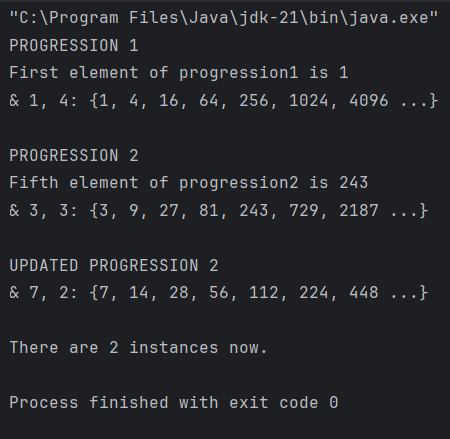
* Клас GeometricProgression:





* Основна програма:





Одна з основних відмінностей між даним кодом і попереднім – це використання ключового слова **static** в Java для статичної змінної **counter**, яка використовується для підрахунку кількості екземплярів класу. В Python приватна змінна оголошується за допомогою подвійного підкреслення «\_\_» на початку назви атрибуту класу, водночас у Java достатньо задати модифікатор доступу **private** для приватної змінної.

До інших відмінностей слід віднести такі:

* В Java типи даних змінних вказуються явно (int, double і т.д.), в Python типи динамічні.
* В Java використовуються генерики для колекцій (**List<Integer>**), в Python – звичайні списки [].
* В Java перевизначається метод **toString()** для повернення рядкового представлення об'єкта. В Python для цього використовується магічний метод **\_\_repr\_\_**. Також у Python конструктор представлений у вигляді магічного методу **\_\_init\_\_**, проте в Java назва конструктора збігається з назвою класу.

**Висновок:** в ході виконання лабораторної роботи було розглянуто парадигму об’єктно-орієнтованого програмування, зокрема поняття класів, об’єктів, статичних класів та методів. Також було реалізовано клас геометричної прогресії з необхідними функціями та його екземпляри на Python і Java. Це дозволило порівняти підходи й особливості двох мов програмування у створенні класів та об’єктів.