Звіт

З лабораторної роботи №2

Студента групи МІТ-31

Добровольського Арсенія Михайловича

Варіант №9

**Тема роботи:** Наслідування та поліморфізм в об’єктах Python.

**Мета роботи:** ознайомитися з поняттям наслідування в об’єктно-орієнтованому програмуванні, навчитися створювати класи та об’єкти з наслідуванням у Python.

Програми, наведені у лабораторній роботі

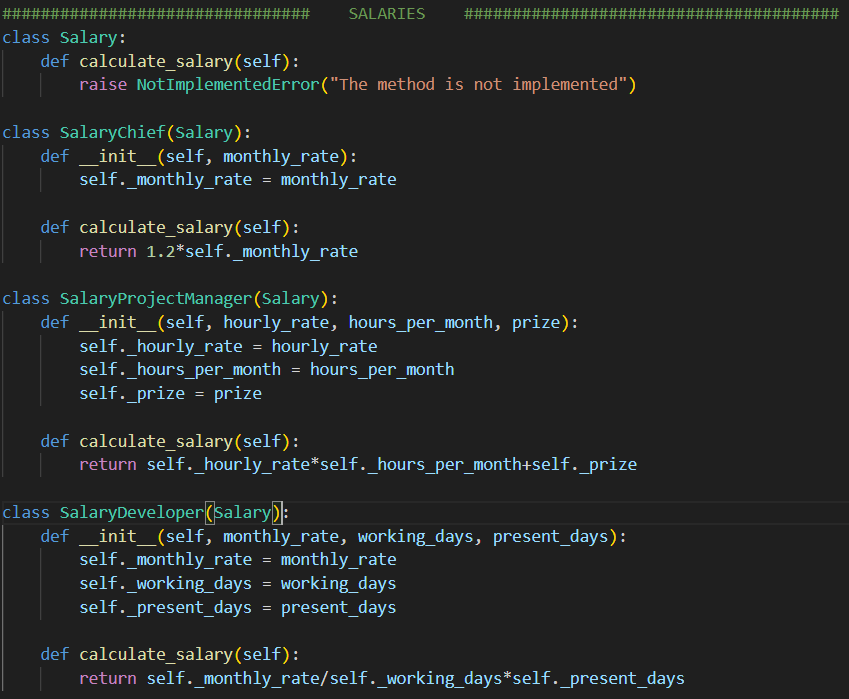
1. У класі **Rabbit**, який є підкласом класу **Animal**, було додано:

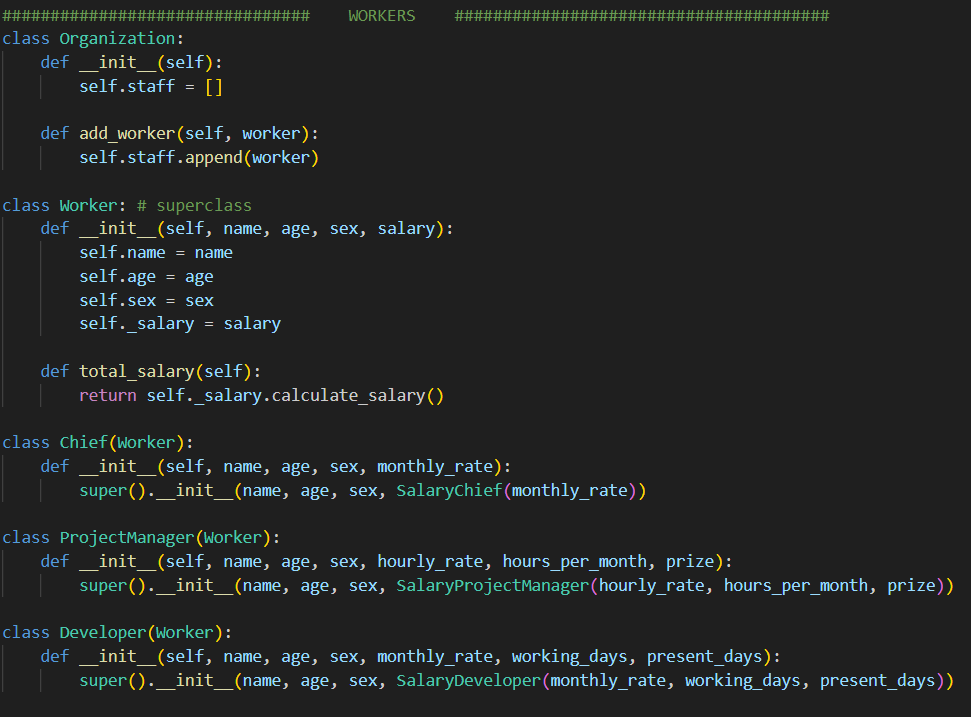
* Два поля (**\_jmpPerStep** – кількість стрибків кролика за один крок, та **\_jmpCounter** – загальна кількість стрибків кролика);
* Один метод (**step(self, stepAmount)**). Він приймає параметр **stepAmount**, який містить кількість кроків кролика. Цей метод виводить на екран кількість стрибків кролика за цю кількість кроків, а також збільшує лічильник **\_jmpCounter** на відповідну кількість стрибків.

1. Додавши метод **sayHello()** у підклас **Amfibia** і запустивши програму, можна переконатися, що виконається саме цей метод, а не метод суперкласу, адже при виконанні програми Python шукає необхідний метод спершу у класі, об’єкт якого викликає цей метод, виконує його, і лише якщо такий метод не знайдено, то відбувається його пошук в суперкласі.

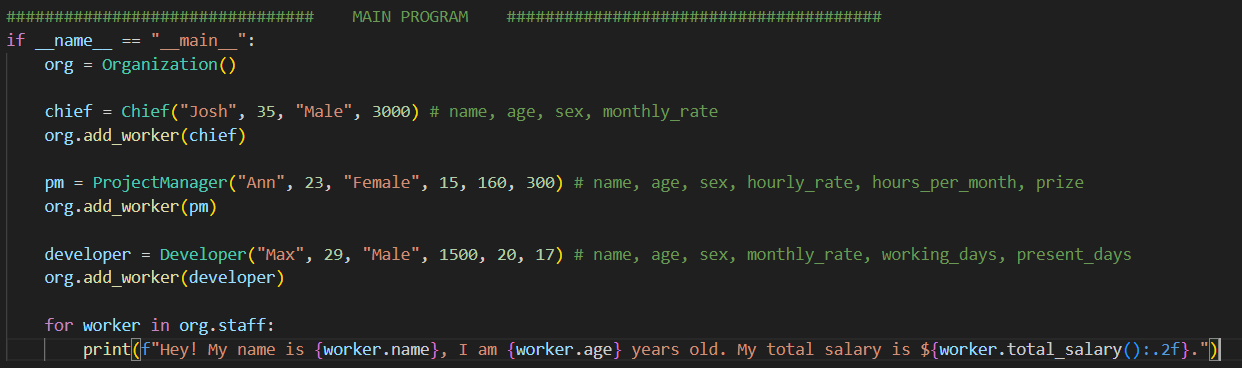
Індивідуальне завдання

Опис класів:





Основна програма:



Даний код створює ієрархію персоналу організації, де є три типи працівників: керівник, проєктний менеджер та розробник. Кожна посада має свою схему нарахування зарплати, яка встановлюється при ініціалізації об'єкта. Для реалізації цього аспекту в класах, які відповідають за створення працівників, було використано композицію.

Спершу визначено базовий клас **Salary**, в якому присутній метод **calculate\_salary()**, який відповідає за розрахунок зарплати. Від цього класу наслідуються три класи: **SalaryChief**, **SalaryProjectManager** і **SalaryDeveloper**, кожен з яких перевизначає метод **calculate\_salary()**. Так, наприклад, клас **SalaryChief** застосовує надбавку до місячної зарплати на 20%, а клас **SalaryProjectManager** розраховує зарплату на основі годинної ставки, кількості годин роботи на місяць та премії. У свою чергу клас **SalaryDeveloper** обчислює зарплату з урахуванням місячної ставки, кількості робочих днів у місяці та кількості днів, в які працівник був безпосередньо присутнім на роботі.

Наступними йдуть класи, пов’язані зі створенням працівників. Клас **Organization** представляє організацію та містить список усіх її співробітників. Метод **add\_worker()** додає нового працівника до списку.

Клас **Worker** є базовим класом для усіх типів працівників і містить загальні властивості (ім’я, вік, стать, розмір заробітної плати) та метод (**total\_salary()**), який обчислює зарплату для кожного працівника, викликаючи метод **calculate\_salary()** відповідного об'єкта зарплати.

Класи **Chief**, **ProjectManager** та **Developer** – це підкласи класу **Worker**, які представляють конкретні посади. Кожен з цих класів ініціалізує зарплату відповідно до своєї схеми нарахування зарплати.

Насамкінець, в основній програмі створюється об'єкт організації **org**, в якийдодаються три працівники з різними посадами. Для перевірки правильності обчислень та роботи програми загалом, додано виведення інформації про кожного співробітника у рядковому представленні.

**Висновок:** під час виконання лабораторної роботи ми детально ознайомилися з поняттями наслідування та поліморфізму в об’єктно-орієнтованому програмуванні, а також реалізували програму, яка описує ієрархію персоналу організації. Причому кожному співробітнику відбувається нарахування тієї чи іншої заробітної плати залежно від його посади.