UML Exercise

To get your hands dirty with OOP, you're going to create an office setup based off of the UMLs below.

Write each class in a different file, and make sure you're importing them where necessary - good luck!

**Document Class**

**-------------------------**

employee\_name: string

type: string

**Employee Class**

**-----------------**

name: string

salary: int (default is 10000)

work(): A function that prints "<name> is working"

**Clerk Class**

**-----------------**

name: string

salary: int (default is 15000)

skills: string[]

add\_skill(skill: string): add a new skill

work(office: Office): A function that pushes a random number of new documents (between 1-10) to the office documents list, and prints "<name> added <# of documents> documents". The document type should be one of the clerk skills.

**Manager Class**

**----------------**

employees: list of employees

salary: int (default is 20000)

name: string

hire\_employee(name): A function that adds s a new clerk

work(office): make all the employees work

**Cleaner Class**

**---------------**

name: string

salary: int (default is 10000)

work(): A function that prints "<name> is cleaning"

**Office Class**

**------------**

documents: list of documents

managers: list of managers

cleaners: list of cleaners

hire\_cleaner(name): adds a new cleaner

hire\_manager(name): adds a new manager

add\_document(document): adds a document

start\_work\_day(): make all managers and all cleaners work

**main.py**

----------

In this file we will put all of our work into practice.

Implement the following items

* create an office
* hire 3 managers to the office
* each manager should hire 2 clerks
* hire 3 cleaners to the office
* start a work day in the office

**Extension**

**1. Raise salaries**

We want to reward the hard working employees, and increase their salaries.

Make the office work for 30 days, and then increase the salaries of all the hard working employees. In our case we will say that those are the ones that added more than 170 documents.

**2. Handle documents**

As part of the clerk work, add document handling. The clerk can handle only a document that he did not submit but the document type matches his skills. Each time the work function will be called the clerk will handle a single document if it exists.

It is up to you to chose how to implement this, so take a minute to think about it.

Good luck!

**DONE( )**

And that's a wrap. OOP is very cool and very popular. Make sure you're using it henceforth ~