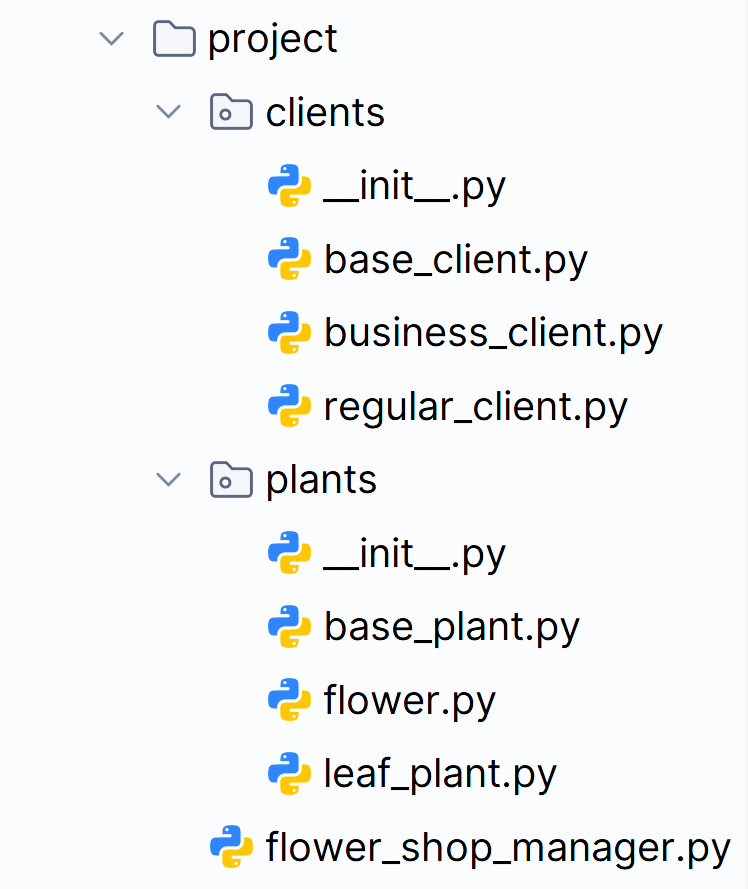
Python OOP Retake Exam - 18 December 2024

Flower Shop



*At Petal & Leaf Floral Boutique* *the plants found their way into loving homes, embodying the beauty of nature and the joy of nurturing life*. *The shop flourished, reflecting the harmonious connection between people and the environment, as each purchase marked a new beginning for both plant and owner.*

***Note: You are not allowed to change the folder and file structure and their names!***



## Judge Upload

For the **first two problems**, create a **zip** file with the **project** **folder** and **upload it** to the judge system.

For the **last problem**, create a **zip** file with the **test folder** and **upload it** to the judge system.

You do not need to include **your venv, .idea, pycache, and \_\_MACOSX** (for Mac users) **in the zip file**, so you do not exceed **the maximum allowed size** of **16.00 KB**.

## Structure (Problem 1) and Functionality (Problem 2)

Your task is to implement all the **classes' structure and functionality** (properties, methods, inheritance, abstraction, etc.)

You are **free to add additional attributes** (instance attributes, class attributes, methods, dunder methods, etc.) to simplify your code and increase readability if it does not change the project's final result under its requirements so that the program works properly.

## Class BasePlant

In the **base\_plant.py** file, the class **BasePlant** should be implemented. It is a **base class** for any **type of plant**,and it **should not be able to be instantiated**.

### Structure

The class should have the following attributes:

* **name:** **str**
  + The value represents the **name of the plant**.
  + If the name is **an empty string or contains only white spaces**, **raise** a ValueError with the message: **"Plant name cannot be null or empty!"**
* **price:** **float**
  + Represents the **price of each plant**.
  + If the price of the plant is **equal to or less than 0.0**, **raise** a ValueError with the message: **"Price must be greater than zero!"**
* **water\_needed:** **int**
  + Represents the **amount of water** needed in ml.
  + Must be **between 1 and 2000**, **both inclusive**. If not, **raise** a ValueError with the message: **"Water needed must be between 1 and 2000 ml!"**

### Methods

#### **\_\_init\_\_(name: str, price: float, water\_needed: int**)

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### plant\_details()

* A **method that provides** information for **each plant**.
* Remember that **each plant type** can **implement** the **method differently**.

## Class Flower

In the **flower.py** file, the class **Flower** should be implemented. **Flowers** are a **type of plant**.

### Methods

\_\_init\_\_(name: str, price: **float, water\_needed: int, blooming\_season: str)**

* In the **\_\_init\_\_** method, all the needed attributes must be set.
* **blooming\_season: str**
  + Represents the **season** when the **flower blooms**.
  + **Valid seasons** are: **"Spring"**, **"Summer"**, **"Fall"**, and **"Winter"**.
  + If the blooming season **is not** one of the valid ones, raise a ValueError with the message: **"Blooming season must be a valid one!"**

#### plant\_details()

* Provides information about the **Flower plants** in the following format:
  + The **price should be formatted to the second decimal** **place**.

**"Flower: {name}, Price: {price},** **Watering: {water\_needed}ml, Blooming Season: {blooming\_season}"**

## Class LeafPlant

In the **leaf\_plant.py** file, the class **LeafPlant** should be implemented. **Leaf plants** are a **type of plant**.

### Methods

#### \_\_**init\_\_(name: str, price: float, water\_needed: int, size: str)**

* In the **\_\_init\_\_** method, all the needed attributes must be set.
* **size: str**
  + Represents the **size** of the **leaf plant**.
  + **Valid sizes** are: **"S"**, **"M"**, and **"L"**.
  + If the size **is not** one of the valid ones, raise a ValueError with the message: **"Size must be a valid one!"**

#### plant\_details()

* Provides information about the **Leaf plants** in the following format:
  + The **price should be formatted to the second decimal** **place**.

**"Leaf Plant: {name}, Price: {price},** **Watering: {water\_needed}ml, Size: {size}"**

## Class BaseClient

In the **base\_client.py** file, the class **BaseClient** should be implemented. It is a **base class** for any **type of client**, and it **should not be able to be instantiated**.

### Structure

The class should have the following attributes:

* **name:** **str**
  + The value represents the **name of the client**.
  + The name must be **at least 2 characters long**. The name **can contain whitespaces**, but **leading or trailing whitespaces** are **not counted as characters**, so **only** **whitespaces are not allowed** either. If the name is invalid, **raise a ValueError** with the following message: **"Name must be at least two characters long!"**
* **phone\_number:** **str**
  + Represents the **phone number** of the **client**.
  + The phone number **can contain only digits**, if **not**, **raise** a **ValueError** with the message: **"Phone number can contain only digits!"**
* **discount: float**
  + Represents the **discount in percentage** (%) each client gets.
  + **Initially** **set** to **0.0**.
* **total\_orders:** **int**
  + Represents the **total number of client's orders**.
  + **Initially set** to **0**.

### Methods

#### **\_\_init\_\_(name: str, phone\_number: str**)

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### update\_discount()

* **Sets** the **discount** to a **certain percentage**, **depending on** the **client's type** and **number of orders**.
* Remember that **each client type** can **implement** the **method differently**.

#### update\_total\_orders()

* **Increases** the **client's total orders by 1**.

#### client\_details()

* **Returns** a **string** with **information** about the **client** in the format given below:
  + The **discount** should be **rounded to the smaller integer**.

**"Client: {name}, Phone number: {phone\_number}, Orders count: {total\_orders}, Discount: {discount}%"**

## Class RegularClient

In the **regular\_client.py** file, the class **RegularClient** should be implemented. The regular client is a **type of client**.

### Methods

#### \_\_init\_\_(**name: str, phone\_number: str**)

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### update\_discount()

* The method **updates** the **discount** as follows:
  + If the client has **1** **or more** **orders**, the discount percentage shall be **set to 5.0%**, **otherwise it remains 0.0%**.

## Class BusinessClient

In the **business\_client.py** file, the class **BusinessClient** should be implemented. The business client is a **type of client**.

### Methods

#### \_\_init\_\_(**name: str, phone\_number: str**)

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### update\_discount()

* The method **updates** the **discount** as follows:
  + If the client has **2** **or more** **orders**, the discount percentage shall be **set to 10.0%**, **otherwise it remains 0.0%**.

## Class FlowerShopManager

In the **flower\_shop\_manager.py** file, the class **FlowerShopManager** should be implemented. It will contain the functionality of the project.

### Structure

The class should have the following attributes:

* **income: float**
  + **Initially set** to **0.0**.
* **plants: list**
  + **Initially** an **empty list** to store **all plants objects** the shop has.
* **clients: list**
  + **Initially** an **empty list** to store **all clients objects** the shop has.

### Methods

#### \_\_init\_\_()

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### add\_plant(plant\_type: str, plant\_name: str, plant\_price: float, plant\_water\_needed: int, plant\_extra\_data: str)

The method **creates** (upon certain conditions) a **plant** of the given type and **adds** it to the **plants** collection:

* **First**, check **if the plant type is valid**, and **if not**, **raise** a **ValueError** with the following message: **"Unknown plant type!"**
* If the type is valid, **create and** **add** the **plant** to the **plants** collection. **Return** the following message: **"{plant\_name} is added to the shop as {plant\_type}."**
  + **Valid types** of plants are: **"Flower"** and **"LeafPlant"**.
  + **Plant extra data** will always be **valid** and containadditional information like **blooming season** or **plant size**.

#### add\_client(client\_type: str, client\_name: str, client\_phone\_number: str)

The method **creates** (upon certain conditions) a **client** of the given type and **adds** it to the **clients** collection:

* **First**, check if the **client type** is valid, and **if** **not**, **raise** a **ValueError** with the following message: **"Unknown client type!"**
* **Then**, check if a **client** with the **same** **phone number** is already added to the list regardless of their type, **do not duplicate phone numbers**,and **raise** a **ValueError** with the following message: **"This phone number has been used!"**
* If none of the above cases is reached, **create and add** the **client** to the **clients** collection. **Return** the following message:

**"{client\_name} is successfully added as a {client\_type}."**

* **Valid types** of clients are: **"RegularClient"** and **"BusinessClient"**.

#### sell\_plants(client\_phone\_number: str, plant\_name: str, plant\_quantity: int)

The method performs a sale upon certain conditions. A **client** with thegiven phone number **purchases plants** with the given **name and quantity** (one or more pieces),if **all pre-defined conditions** are **met**:

* **First**,check if sucha **client exists**:
  + Validate a client's **existence** with such **phone number** in the collection of added **clients**.
  + If **no client is found**, the method **raises** a **ValueError** with the following message:

**"Client not found!"**

* **Then**,check if there are **plants** with this **name**:
  + Validate the **existence** of **plants** with the given **name** in the collection of **plants**.
  + If **no plants are found**, the method **raises** a **ValueError** with the following message:

**"Plants not found!"**

* **After**,checkifthe **number** of **plants found** **corresponds** to the **quantity** the clientis trying to purchase:
  + If the **shop** does **not have enough quantity** of **plants** with the **given name**,the **sale is impossible**.
  + **Return** the following message: **"Not enough plant quantity."**
* **Finally**, the **shop** **sells** the **plants**.A series of **actions are required**:
  + **Sold plant objects are removed** from the **shop's plants** collection.
    - Make sure you **remove** the **correct count** of **plant objects** (first occurrences).
  + The **shop's income** is **updated** with the **amount** of the **current sale**.
    - Calculatethe **amount based** on **plant price**, **quantity** and **client's discount**.
  + The **client's number** of **total orders is updated**.
    - **Hint**: You can use the designated method [**update\_total\_orders()**](#_update_total_orders())
  + The **client's discount percentage is updated** tomake sure it will **affect** the **next order**.
    - **Hint**: You can use the designated method [**update\_discount()**](#_update_discount())
    - The **update** shall be made **after** the **calculations** for the **current order**.
    - Remember that **each type** of **client updates** their **discount differently**.
  + **Return** the following message and **format the amount** to the **second decimal place**:

**"{plant\_quantity}pcs. of {plant\_name} plant sold for {order\_amount}"**

#### remove\_plant(plant\_name: str)

The method **removes** the **first** **plant object with the given name** if such **exists**.

* **First**, check if there is at least one **plant** with such **name** in the collection of **plants**. If **not**, **return** the message: **"No such plant name."**
* If there is at least one **plant**, **remove** the **first occurrence** from the **shop's plants collection** and **return** the following message: **"Removed {plant\_details}"**
  + **Note**: Use the [**plant\_details()**](#_plant_details()) method.

#### remove\_clients()

The method **removes** **all clients who have no orders** (**total\_orders=0**).

* The method should **track the number of clients** (zero or more) being **removed**.
* **Return** the following message: **"{count} client/s removed."**

#### shop\_report()

The method should **return** detailed **information** about the **flower shop**:

* Include information about **unsold plants and their count**.
  + **Sort** them by the **count** of **unsold plants**,per **plant name** in **descending order**.
  + If there is more than one plant name with the **same count**, **sort** thembythe **plant's name ascending**. See the [**Examples**](#_Examples).
* Include information about **clients**, **sorted** by **total orders count**, **descending**,then by **phone number ascending**.
  + **Note**: Use the client's [**client\_details()**](#_client_details()) method
* The **report** should follow the **format** of the example below (**each on a new line**):
  + **Format** the **income** to the **second decimal** place.

**"~Flower Shop Report~**

**Income: {income}**

**Count of orders: {count\_of\_all\_orders}**

**~~Unsold plants: {unsold\_plants\_count}~~**

**{plant\_name1}: {count}**

**{plant\_name2}: {count}**

**...**

**{plant\_namen}: {count}**

**~~Clients number: {num\_of\_clients}~~**

**{client\_details1}**

**{client\_details2}**

**...**

**{client\_detailsn}"**

### Examples

|  |
| --- |
| **Test Code** |
| ***# Create an instance of FlowerShopManager***  manager = FlowerShopManager()  ***# Add plants***  print(manager.add\_plant("Flower", "Rose", 15.50, 200, "Spring"))  print(manager.add\_plant("Flower", "Rose", 15.50, 200, "Spring"))  print(manager.add\_plant("Flower", "Rose", 15.50, 200, "Spring"))  print(manager.add\_plant("Flower", "Rose", 15.50, 200, "Spring"))  print(manager.add\_plant("Flower", "Tulip", 12.00, 150, "Spring"))  print(manager.add\_plant("Flower", "Tulip", 12.00, 150, "Spring"))  print(manager.add\_plant("Flower", "Lily", 20.00, 180, "Summer"))  print(manager.add\_plant("LeafPlant", "Cactus", 8.00, 50, "M"))  print(manager.add\_plant("LeafPlant", "Cactus", 8.00, 50, "M"))  print(manager.add\_plant("LeafPlant", "Fern", 6.50, 100, "S"))  print(manager.add\_plant("LeafPlant", "Fern", 6.50, 100, "S"))  print(manager.add\_plant("LeafPlant", "Fern", 6.50, 100, "S"))  print(manager.add\_plant("LeafPlant", "Snake Plant", 12.00, 200, "L"))  print(manager.add\_plant("LeafPlant", "Snake Plant", 12.00, 200, "L"))  print()  ***# Add clients***  print(manager.add\_client("RegularClient", "Alice Johnson", "1234567890"))  print(manager.add\_client("RegularClient", "Bob Smith", "0987654321"))  print(manager.add\_client("BusinessClient", "Greenhouse Inc.", "5647382910"))  print(manager.add\_client("BusinessClient", "CoolGarden Plc.", "9647382910"))  print(manager.add\_client("RegularClient", "Peter Johnson", "382910"))  print()  ***# Perform sales***  print(manager.sell\_plants("1234567890", "Rose", 3))  print(manager.sell\_plants("0987654321", "Tulip", 2))  print(manager.sell\_plants("5647382910", "Cactus", 1))  print()  ***# Get shop report***  print(manager.shop\_report())  print()  ***# Perform sales***  print(manager.sell\_plants("1234567890", "Lily", 2))  print(manager.sell\_plants("0987654321", "Fern", 1))  print(manager.sell\_plants("5647382910", "Snake Plant", 2))  print()  ***# Remove a plant***  print(manager.remove\_plant("Nonexistent"))  print(manager.remove\_plant("Cactus"))  print()  ***# Get shop report***  print(manager.shop\_report())  print()  ***# Remove clients who have no orders***  print(manager.remove\_clients())  print(manager.remove\_clients()) |
| **Output** |
| Rose is added to the shop as Flower.  Rose is added to the shop as Flower.  Rose is added to the shop as Flower.  Rose is added to the shop as Flower.  Tulip is added to the shop as Flower.  Tulip is added to the shop as Flower.  Lily is added to the shop as Flower.  Cactus is added to the shop as LeafPlant.  Cactus is added to the shop as LeafPlant.  Fern is added to the shop as LeafPlant.  Fern is added to the shop as LeafPlant.  Fern is added to the shop as LeafPlant.  Snake Plant is added to the shop as LeafPlant.  Snake Plant is added to the shop as LeafPlant.  Alice Johnson is successfully added as a RegularClient.  Bob Smith is successfully added as a RegularClient.  Greenhouse Inc. is successfully added as a BusinessClient.  CoolGarden Plc. is successfully added as a BusinessClient.  Peter Johnson is successfully added as a RegularClient.  3pcs. of Rose plant sold for 46.50  2pcs. of Tulip plant sold for 24.00  1pcs. of Cactus plant sold for 8.00  ~Flower Shop Report~  Income: 78.50  Count of orders: 3  ~~Unsold plants: 8~~  Fern: 3  Snake Plant: 2  Cactus: 1  Lily: 1  Rose: 1  ~~Clients number: 5~~  Client: Bob Smith, Phone number: 0987654321, Orders count: 1, Discount: 5%  Client: Alice Johnson, Phone number: 1234567890, Orders count: 1, Discount: 5%  Client: Greenhouse Inc., Phone number: 5647382910, Orders count: 1, Discount: 0%  Client: Peter Johnson, Phone number: 382910, Orders count: 0, Discount: 0%  Client: CoolGarden Plc., Phone number: 9647382910, Orders count: 0, Discount: 0%  Not enough plant quantity.  1pcs. of Fern plant sold for 6.17  2pcs. of Snake Plant plant sold for 24.00  No such plant name.  Removed Leaf Plant: Cactus, Price: 8.00, Watering: 50ml, Size: M  ~Flower Shop Report~  Income: 108.67  Count of orders: 5  ~~Unsold plants: 4~~  Fern: 2  Lily: 1  Rose: 1  ~~Clients number: 5~~  Client: Bob Smith, Phone number: 0987654321, Orders count: 2, Discount: 5%  Client: Greenhouse Inc., Phone number: 5647382910, Orders count: 2, Discount: 10%  Client: Alice Johnson, Phone number: 1234567890, Orders count: 1, Discount: 5%  Client: Peter Johnson, Phone number: 382910, Orders count: 0, Discount: 0%  Client: CoolGarden Plc., Phone number: 9647382910, Orders count: 0, Discount: 0%  2 client/s removed.  0 client/s removed. |

# **Task 3: Unit Tests (100 points)**

You will **be provided with another skeleton** for this problem. **Open** the **new skeleton** as a **new project** and write tests for the **Gallery** class. The class will have some methods, fields, and one constructor, all of them working properly. You are **NOT ALLOWED** to change anything in the class code. Cover the whole class with unit tests to make sure that the class is working as intended. Submit **only the test** folder.