Problem Set

You have to create a simulation of a Technology or Digital Electronics store. This has to be done in OOP. Each of the classes will be at their own file, aka module in python. Then by importing the appropriate class, you would operate and implement the code.

For importing one class in another follow this rule:

```python
from my_file import MyFirstClass
```

Here MyFirstClass is a class defined in my_file. For using this class in another file, it is imported using this line.

Here are the description of the what you need to do:

Tech Class

- Create a **Tech** class in a file named **tech_product.py**. It should have the following properties
  - A **class variable** named **total_products** to count the number of products
  - A **class variable** named **discount** to apply discount to products. Initialize this as **0.5**
  - Following **instance variables** are to be set properly:
    - name : A String
    - price: An Integer
    - weight: An Integer
    - color: A String
- The number of products should be counted when we create an object
- Create a method named **apply_discount** for applying discounts upon the product's price.
- Create a method named **calculate_shipping_cost** that takes a **rate** parameter and returns the product of **weight and rate** as the shipping cost.
- Create a **Class Method** to return the total number of products, that is the number of objects instantiated.

After the tech class you need to create the following classes:

Laptop Class

- Create a class named **Laptop** in a separate file **laptop_tech.py** and **inherit** the **Tech** class.
- Set the appropriate **instance variables** of the **Tech** class. The additional class variables are:
  - ram: An Integer
  - cpu: A String
  - storage: An Integer

  Set the appropriate additional variables.
- Create an appropriate **string** representation of the object of the class.
- Create a method called **set_double_price** . This method doubles the price of the instance and sets it appropriately.

- Create a method called **change_specs** . This method takes in new **ram, storage** as parameters and sets the **instance variables** appropriately. Note that if the **ram and storage** is higher than the current specs, then the current price **must be increased by 10000**.

Mobile Class

- Create a class named **Mobile** in a separate file **mobile_tech.py** and **inherit** the **Tech** class.
- Set the appropriate **instance variables** of the **Tech** class. The additional class variables are:
  - screen: A String
  - camera: An Integer

  Set the appropriate additional variables.
- Create an appropriate **string** representation of the object of the class.
- Override the **apply_discount** method. This time the discount should be halved during calculation. Example:

  -
    ```
    super().discount/2
    ```

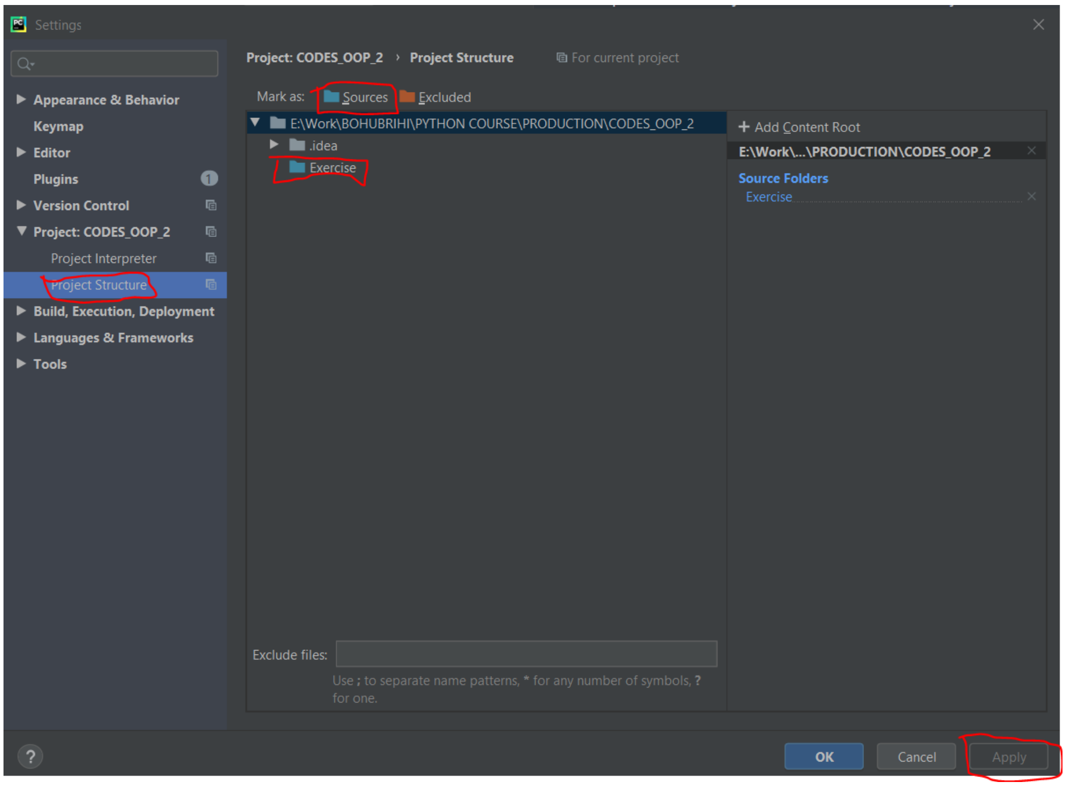
SalesPerson Class

- Create a **SalesPerson** class in a module named **sales_person.py**
- A **class variable** named **employee_id** to give a cumulative serial to the employee_id. That is every time a new instance of the employee is created, this value goes up by one. Set the initial value to be **0**.
- Following **instance variables** are to be set properly:
  - first_name: A String
  - last_name: A String
  - salary: An Integer
  - date_joined: Date Object(Look this up in the internet or documentation if unfamiliar)
  - products_sold: A list of products from the other classes you prepared, **Laptop and Mobile** classes. Initialize this as an empty list.
  - total_sales: An Integer. Initialize this as **0**.
- Write the following 6 methods:
  - Create a method **sell_products** that adds a product to the **product_list**. This method takes a **product** parameter. **Print the appropriate message**.
  - Create a method **display_sales** that displays all the products sold by a salesman instance or object. **Print the appropriate message**.
  - Create a method **calculate_sales** that calculates the total price of the products sold by a salesman instance or object. After calculation, set the **total_sales** to this value and return the total value.
  - Create a method **calculate_commission** that calculates the commission received by a salesperson for all the sales that he/she has made. This method takes **percentage** as a parameter and returns **the percentage upon total sales** of the salesperson instance. **Print the appropriate message**. (Hint: *Use the calculate_sales method*)
  - Create a method **total_products_sold** that returns **total number** of the products sold by a salesman instance or object. **Print the appropriate message**.

    - Create a method **sort_by_price** that **sorts** the products sold by a salesman instance or object according to **price. You must use a lambda function to do this.**

NOTE ABOUT PROJECT STRUCTURE

In the solution, the whole project is done on a separate folder named **Exercise**. In order for pycharm to recognize this as the source folder and access to autocompletion of codes and avoid unnecessary errors, do the following:

- Go to *File* > *Settings*
- There will be a tab on the left side of the window named **Project: CODES_OOP_2**
- The folder name **CODES_OOP_2** might be different if you start a new project on your own.
- Click on **Project Structure**
- Select The **Exercise** folder. Then above the exercise folder, You will see an option as **Mark as: Sources Excluded**
- After selecting the **Exercise** folder, Select the **Sources** option. Then on the right corner of the window, click on **Apply**. Then you can close the window. Here is a snapshot:
    - 

Settings

Project: CODES_OOP_2 › Project Structure   For current project

Mark as: Sources Excluded

E:\Work\BOHUBRIHI\PYTHON COURSE\PRODUCTION\CODES_OOP_2
.idea
Exercise

+ Add Content Root
E:\Work\...\PRODUCTION\CODES_OOP_2
Source Folders
Exercise

Exclude files:
Use ; to separate name patterns, * for any number of symbols, ? for one.