Object Oriented Programming Lab

During this project, we'll fix the knowledge related to what we have learned so far about classes, instances, inheritance, methods and so on.

Please install and use a Python 3.x version. the newest, the better. As for the IDE, I recommend PyCharm from JetBrains, there is a PyCharm 2022 version on their website and you can use your e-uvt.ro email addresses for a full functional license valid for one year.

The proposed theme is to create a skeleton for an online store, which could be used in text mode. The suggested store will be one which sells electronics, but basically it could sell anything else. Like in a real store, this one will contain some categories of products and for each category, we need to have distinct products. We will be able to manage the categories and the products (`add, remove, list`). For those of you who feel adventurous implement `update/modify` too. Like in a real store, we need to place orders, to buy specific products.

From the programming point of view, we need to implement a series of classes which will model the objects in the store. Below are some suggestions (but feel free to amend or improve the list) that should help you during the design of the project.

`Categories` - will contain a collection of all categories we enter. Examples of categories: `Amplifiers`, `Receivers`, `Speakers`, `Turntables` and so on

`Category` - describes a single category

`Product` - base class for all the products we'll have

`Amplifier` - inherits the Product class and will contain info specific for that kind of product (power, number of channels, size)

`Receiver` - inherits the Products class and contains info specific for that kind of product (number of channels, color, size)

`Turntable` - inherits the Product class and contains specific info for that kind of product (speed, connection type (wired, bluetooth), size)

`Products` - contains a collections of all the products in the store

`Orders` - models an order to be placed in the store. Basically a collection of products and quantities also with a destination address

All these components of our store will be persisted on the disk. We won't use any database, but we'll use JSON serialization. So, when designing the requested classes, take into account that their instances have to be serialized.

The main program will display a menu from which we can perform various operation in the store. We could have, for example, hierarchical menus, or flat menus.

An example of hierarchical menu:

```
1. Categories
2. Products
3. Orders
4. Exit
```

Then, if you choose "`1.Categories`" another submenu will be displayed:

```
1. Add a category
2. Remove a category
3. Display all the categories
```

The same for `Products` and `Orders`

When thinking of flat menus we can have:

```
1. Add a category
2. Remove a category
3. Display all the categories
4. Add a product
5. Remove a product
6. Display all the products\
7. Place a new order
8. Display all orders
9. Exit
```

Suggestion for implementing the menus: one possible way is to use multiple if-s, but imagine that the code will become a mess with so many possible branches, so I would simulate a switch instruction using dictionaries.

```
def add_category(category):
     // add category stuff

def remove_category(category):
     // remove category stuff

def display_categories():
    // display categories stuff

.
.
.

def error_handler():
    print("This option does not exist")


menus = {1: add_category, 2: remove:category, 3: display_category, …}
option = int(input("Enter your option:"))

func = menus.get(option, error_handler)
func()
```