Assignment Programming

Fabrizio Rocco

June 2022

1 The project

The scope of this assignment is the conceptual and technical definition of a warehouse and annexed stocks.

2 Technical Specifications

The project has the following conceptual specifications:
The program must have the following features (constraints).
- Add stock.
- Remove stock.
- Get number of stock of a certain type.
- Three types of stock should be defined.

The program should provide at least one of the following functionality:
- Have two wearhouses were the stock can be shared between them.
- Provide a way of searching if stock is available.
- Create a report of all the stock available in the wearhouse.

3 Considerations and design

Please note the following considerations:

- Probably the assignment was built in a way that required the warehouse to use only the methods of the stocks class. I decided to use only part of them. The reasoning behind such choice is that if we'd like to add a new stock in the warehouse, and we use the stocks method to do that, what we're going to obtain is a general increase in all the stocks of that items. Instead, what I wanted to achieve is that each warehouse is independent in terms of products stored.

  For this reason my class Warehouse takes the stocks method in the constructor but then uses an ad-hoc addStocks and an ad-hoc removeStocks method.

- The code is entirely written in camelCase. I tried to keep as much consistent as possible all the variables and the capital letters.
  I used the tabs with the tab size of 4.
  Between each function there is a space of two lines.
- I used try and except for test purposes (especially for ValueErrors). Then I decided to limit them to increase the readability of the code.
- I'm aware of the fact that in general it's always better that a function returns something rather than print it. Despite this, I decided to use print methods since our application will work only on a CLI. In case we'll deploy our application or we'll use a specific GUI, it's good to use the return.
- For the sake of the project I created 4 predefined stocks (Phone, Tablet, Computer, Television). In a future version, could be interesting to let the user define which stocks to store.
- The code is tested on a virtual environment with Python 3.10 installed with the default packages.

4 The code

4.1 Import libraries and dependencies

The following snippet shows how there are no main libraries required. All these are added by Python 3.10 by default and click is for the design of the CLI.

```python
from typing_extensions import Self
from click import echo
from utils import let_user_pick
```

4.2 Stocks

The idea is that there is a single class called Stock that works as a parent class. All the other classes are just inherited from the parent one.
I used the super() function to make sure the constructor method of the children would be equal to the parent one.

```python
class Stock:

    def __init__(self) -> None:
        self.amount = 0


    def add(self, amount:int) -> None:
        """ This function increases the amount of stock by the
    input variable"""
        self.amount += amount


    def remove(self, amount:int) -> None:
        """ This function decreases the amount of stock by the
    input variable"""
        self.amount -= amount


    def get_amount(self) -> None:
        """ This function returns the amount of the given stock"""
        return self.amount
```

```python
# This part initializes 4 classes relative to the specific stocks
    we're going to use.
# Each of this class inherits methods from the main "Stock" class.
class Phone (Stock):

    def __init__(self) -> None:
        super().__init__()

class Computer (Stock):

    def __init__(self) -> None:
        super().__init__()

class Tablet (Stock):

    def __init__(self) -> None:
        super().__init__()

class Television (Stock):

    def __init__(self) -> None:
        super().__init__()
```

4.3 Warehouse class

The warehouse class uses the previously defined classes in the constructor method through the get_amount() function

```python
# The following part initializes a warehouse to manage all the
    previously defined stocks.
# The constructor uses the method from the "Stock" class.

class Warehouse():

    def __init__(self, phone, computer, tablet, television) -> None
    :
        """ The following constructor takes as parameter the
    instances of the stocks previously created and compute the
    current amount"""
        self.phone = phone.get_amount()
        self.computer = computer.get_amount()
        self.tablet = tablet.get_amount()
        self.television = television.get_amount()


    def addStock(self, stock:str, amount:int =1) -> None:
        """ The following function takes two parameters (stock, amount) which are respectively the type of the stock ("phone", "tablet", "computer", "television")
            and the amount of stocks we want to add as integer (Default value = 1).
            It increases the specific stock by the amount.
        """
        match stock:
            case "phone":
                self.phone += amount
            case "computer":
                self.computer += amount
            case "tablet":
                self.tablet += amount
            case "television":
                self.television += amount


    def removeStock(self, stock:str, amount:int=1) -> None:
        """ The following function takes two parameters (stock, amount) which are respectively the type of the stock ("phone", "tablet", "computer", "television")
            and the amount of stocks we want to remove as integer (Default value = 1).
            It decreases the specific stock by the amount.
            To avoid negative values, this function computes the maximum value between 0 and the difference of the current value and the amount given by the user.
            If the amount is 0, the function will print a warning message.
        """
        match stock:
            case "phone":
                self.phone = max(self.phone - amount, 0)
                if self.phones == 0:
                    print("\n Warning: no more phones! \n")
            case "computer":
                self.computer =  max(self.computer - amount, 0)
                if self.computer == 0:
                    print("\n Warning: no more computers! \n")
            case "tablet":
                self.tablet =  max(self.tablet - amount, 0)
                if self.tablet == 0:
                    print("\n Warning: no more tablets! \n")
            case "television":
                self.television =  max(self.television - amount, 0)
                if self.television == 0:
                    print("\n Warning: no more televisions! \n")


    def search(self, stock:str):
        """ The following function takes a single parameter (stock) which is the type of the stock ("phone", "tablet", "computer", "television").
            It searches the specific stock in the warehouse.
            It prints a message if the stock is available or not.
        """
        match stock:
            case "phone":
                if self.phone > 0:
                    print( "{} found!".format(stock))
                else:
                    print( "{} not found!".format(stock))
            case "computer":
                if self.computer > 0:
                    print( "{} found!".format(stock))
                else:
                    print( "{} not found!".format(stock))
            case "tablet":
                if self.tablet > 0:
                    print( "{} found!".format(stock))
                else:
                    print( "{} not found!".format(stock))
            case "television":
                if self.television > 0:
                    print( "{} found!".format(stock))
                else:
                    print( "{} not found!".format(stock))


    def generateReport (self):
        """ The following function takes no parameters and print a dictionary with the amount of stocks available"""
        self.report = {"Amount of Phones": self.phone, "Amount of Computers": self.computer, "Amount of Tablets": self.tablet, "Amount of Televisions": self.television,}
        return self.report
```

4.4 Main handler function

The handler function is a function created to run the script in the CLI. The first choice deals with the choice of the region (Barcelona vs Madrid) and then a main manù will be shown.
This list of possible actions is inserted in a while loop and each single option takes care of the geographical choice of the region (previously defined).
So the structure will be always function of menù (e.g. ”Add a stock”), choice of the region (e.g. ”If the user chose Madrid”)and code (e.g.”Code of the specific block”)

```
def main():
    """ The following variables instantiate the 4 stock's classes
    """
    phone = Phone()
    computer = Computer()
    tablet = Tablet()
    television = Television()

    """ The following lines allow the user to choose the city on which establish the warehouse between Barcelona and Madrid
        To do that, a specific function called "let_user_pick" has been provided in a separate file (utils.py) to facilitate user's choice
```

```python
    """
    cities = ["Barcelona", "Madrid"]
    region = let_user_pick(cities)
    match region:
        case 1:
            barcelonaWarehouse = Warehouse(phone, computer, tablet, television)
        case 2:
            madridWarehouse = Warehouse(phone, computer, tablet, television)


    """ This is the main men  of the script. It allows the user to choose the action to do between
        - Add a stock
        - Remove a stock
        - Find a stock
        - Get all the stocks for the current warehouse
        - Quit the script
        To do that, a specific function called "let_user_pick" has been provided in a separate file (utils.py) to facilitate user's choice
    """
    options = ["Add Stock", "Remove Stock", "Find Stock", "Get all Stocks", "Quit"]
    while True:
        handler = let_user_pick(options)
        match handler:

            case 1:
                if region == 1:
                    stockTypeOptions = ["Add a Phone stock", "Add a Computer stock", "Add a Tablet stock", "Add a Television stock"]
                    stockType = let_user_pick(stockTypeOptions)
                    match stockType:
                        case 1:
                            stockType = "phone"
                        case 2:
                            stockType = "computer"
                        case 3:
                            stockType = "tablet"
                        case 4:
                            stockType = "television"
                    try:
                        stockAmount = int(input("How many stocks of {} do you want to add?".format(stockType)))
                    except:
                        print("Please insert only numbers!")
                    barcelonaWarehouse.addStock(stockType, stockAmount)
                    print("\n Added {} stocks of {} ! \n".format(stockAmount,stockType))
                else:
                    stockTypeOptions = ["Add a Phone stock", "Add a Computer stock", "Add a Tablet stock", "Add a Television stock"]
                    stockType = let_user_pick(stockTypeOptions)
                    match stockType:
                        case 1:
                            stockType = "phone"
                        case 2:
                            stockType = "computer"
                        case 3:
                            stockType = "tablet"
                        case 4:
                            stockType = "television"
                    try:
                        stockAmount = int(input("How many stocks of {} do you want to add?".format(stockType)))
                    except:
                        print("Please insert only numbers!")
                    madridWarehouse.addStock(stockType,stockAmount)
                    print("\n Added {} stocks of {} ! \n".format(stockAmount,stockType))

            case 2:
                if region == 1:
                    stockTypeOptions = ["Remove a Phone stock", "Remove a Computer stock", "Remove a Tablet stock", "Remove a Television stock"]
                    stockType = let_user_pick(stockTypeOptions)
                    match stockType:
                        case 1:
                            stockType = "phone"
                        case 2:
                            stockType = "computer"
                        case 3:
                            stockType = "tablet"
                        case 4:
                            stockType = "television"
                    try:
                        stockAmount = int(input("How many stocks of {} do you want to remove?".format(stockType)))
                    except:
                        print("Please insert only numbers!")
                    barcelonaWarehouse.removeStock(stockType,stockAmount)
                    print("\n Removed {} stocks of {} ! \n".format(stockAmount,stockType))
                else:
                    stockTypeOptions = ["Remove a Phone stock", "Remove a Computer stock", "Remove a Tablet stock", "Remove a Television stock"]
                    stockType = let_user_pick(stockTypeOptions)
                    match stockType:
                        case 1:
                            stockType = "phone"
                        case 2:
                            stockType = "computer"
                        case 3:
                            stockType = "tablet"
                        case 4:
                            stockType = "television"
```

```python
                    try:
                        stockAmount = int(input("How many stocks of {} do you want to remove?".format(stockType)))
                    except:
                        print("Please insert only numbers!")
                    madridWarehouse.removeStock(stockType, stockAmount)
                    print("\n Removed {} stocks of {} ! \n".format(stockAmount,stockType))

        case 3:
            if region == 1:
                stockTypeOptions = ["Find a Phone stock", "Find a Computer stock", "Find a Tablet stock", "Find a Television stock"]
                stockType = let_user_pick(stockTypeOptions)
                match stockType:
                    case 1:
                        stockType = "phone"
                    case 2:
                        stockType = "computer"
                    case 3:
                        stockType = "tablet"
                    case 4:
                        stockType = "television"
                print(barcelonaWarehouse.search(stockType))

            else:
                stockTypeOptions = ["Find a Phone stock", "Find a Computer stock", "Find a Tablet stock", "Find a Television stock"]
                stockType = let_user_pick(stockTypeOptions)
                match stockType:
                    case 1:
                        stockType = "phone"
                    case 2:
                        stockType = "computer"
                    case 3:
                        stockType = "tablet"
                    case 4:
                        stockType = "television"
                print(madridWarehouse.search(stockType))

        case 4:
            if region == 1:
                print(barcelonaWarehouse.generateReport())
            else:
                print(madridWarehouse.generateReport())

        case 5:
            break
```