Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра «Електронних обчислювальних машин»


Звіт

з лабораторної роботи № 9

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «“ОСНОВИ ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ У PYTHON”»

**Виконав:**
студент групи *КІ-306*
*Олесько Б. А.*
**Прийняв:**
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

Мета: оволодіти навиками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Модулі Модулем у Python називається файл з розширенням *.py. Ці файли можуть містити звичайні скрипти, змінні, функції, класи і їх комбінації. Python дозволяє структурувати код програм у різні модулі та доступатися до класів, функцій і змінних, які у них знаходяться з інших модулів. Для цього використовуються два оператори – import та from-import.

Варіант 11

11. Монітор 11. Сенсорний екран

```
from TouchScreen import TouchScreen
from monitor import  Monitor
def main():
    monitor = Monitor("aa","1")
    ts =TouchScreen("aa","1","mouse",True,200,5,True,12,24)
    TouchScreen.x=26
    ts.change_type("New Type2")
    monitor.check_additional()
    monitor.change_additional_devices("keyboard")
    monitor.buy(2)
    monitor.turn_off_available()
    monitor.change_type("New Type")
    monitor.change_version("1.0")
    monitor.status()
    monitor.add_price(50)
    monitor.add_count(5)
    ts.status()
    monitor.clear_log_file()
    monitor.write_to_log_file("dfdgfgdfgs")
if __name__ == "__main__":
    main()
```

```
class Monitor:
  def __init__(self, types="AAd27O", versions="0.1",additionals=True,additional__devices = "mouse",prices = 100,counts =
1,is_availables = True,x=0):
    self.additional = additionals
    self.additional_devices = additional__devices
    self.type = types
    self.version = versions
    self.price = prices
    self.count =counts
    self.is_available = is_availables
  def check_additional(self):
    print (f"This monitor has additional devices.\n") if  self.additional == True else print (f"This monitor does not have
additional devices.\n")
  def change_additional_devices(self, devices):
    if not self.additional:
      print ("This monitor does not have additional devices.\n")
    else:
      print(f"You changed additional devices from {self.additional_devices} to {devices}\n")
      self.additional_devices = devices
  def buy(self, number):
    if self.count < number:
      print(f"Oops! You want to buy more than we have available.\n")
```

```
        else:
            n = int(number)
            self.count -= n
            print (f"You bought {n} monitors. Current count: {self.count}\n")
    def turn_off_available(self):
        self.is_available = False
        print(f"Monitor is now unavailable.\n")
    def change_type(self, new_type):
        print(f"Type changed {self.type} to: {new_type}\n")
        self.type = new_type
    def change_version(self, new_version):
        print(f"Version {self.version} changed to: {new_version}\n")
        self.version = new_version
    def status(self):
        status_info = f"Type: { self.type}\nVersion: { self.version}\nPrice: { self.price}\nCount: { self.count}\nAvailability: {
self.is_available} "
        print(status_info)
    def add_price(self, prices):
        print(f"New price:{self.price} -->{prices}\n")
        self.price += prices
    def add_count(self, counts):
        print(f"New count: {self.count} --> {counts}\n")
        self.count += counts
    def clear_log_file(self):
        with open("Olesko.txt", "w") as log_file:
            log_file.truncate(0)
    def write_to_log_file(self, message):
        with open("Olesko.txt", "a") as log_file:
            log_file.write(message + "\n")
```

```
from monitor  import Monitor
class TouchScreen(Monitor):
        x= None
        y= None
        def __init__(self, types,
versions,additionals,additional__devices,prices,counts,is_availables,x_position,y_position):
            super(TouchScreen,self).__init__( types,
versions,additionals,additional__devices,prices,counts,is_availables)
            self.x = x_position
            self.y = y_position
        def status(self):
            super().status()
            print(f"X = {self.x} Y = {self.y}")
        def get_X(self):
            return  self.x
        def get_Y(self):
            return  self.y
        def set_X(self,x):
            self.x = x
        def set_Y(self,y):
            self.y = y
```

## Відповіді на контрольні питання

1. Модулі - це файли в мові програмування Python, які містять функції, змінні та класи, які можна використовувати у інших програмах для полегшення організації коду і підтримки його читабельності та модульності.

2. Для імпортування модуля використовується ключове слово import, наприклад: import module_name.

3. Для оголошення класу використовується ключове слово class, за яким слідує ім'я класу, наприклад: class MyClass:.

4. У класі можуть міститися атрибути (змінні), методи (функції), конструктори, властивості, інші класи та багато іншого.

5. Конструктор класу називається __init__. Це спеціальний метод, який викликається при створенні об'єкта класу і використовується для ініціалізації атрибутів об'єкта.

6. Спадкування в Python здійснюється шляхом вказання базового класу в оголошенні похідного класу, наприклад: class SubClass(BaseClass):.

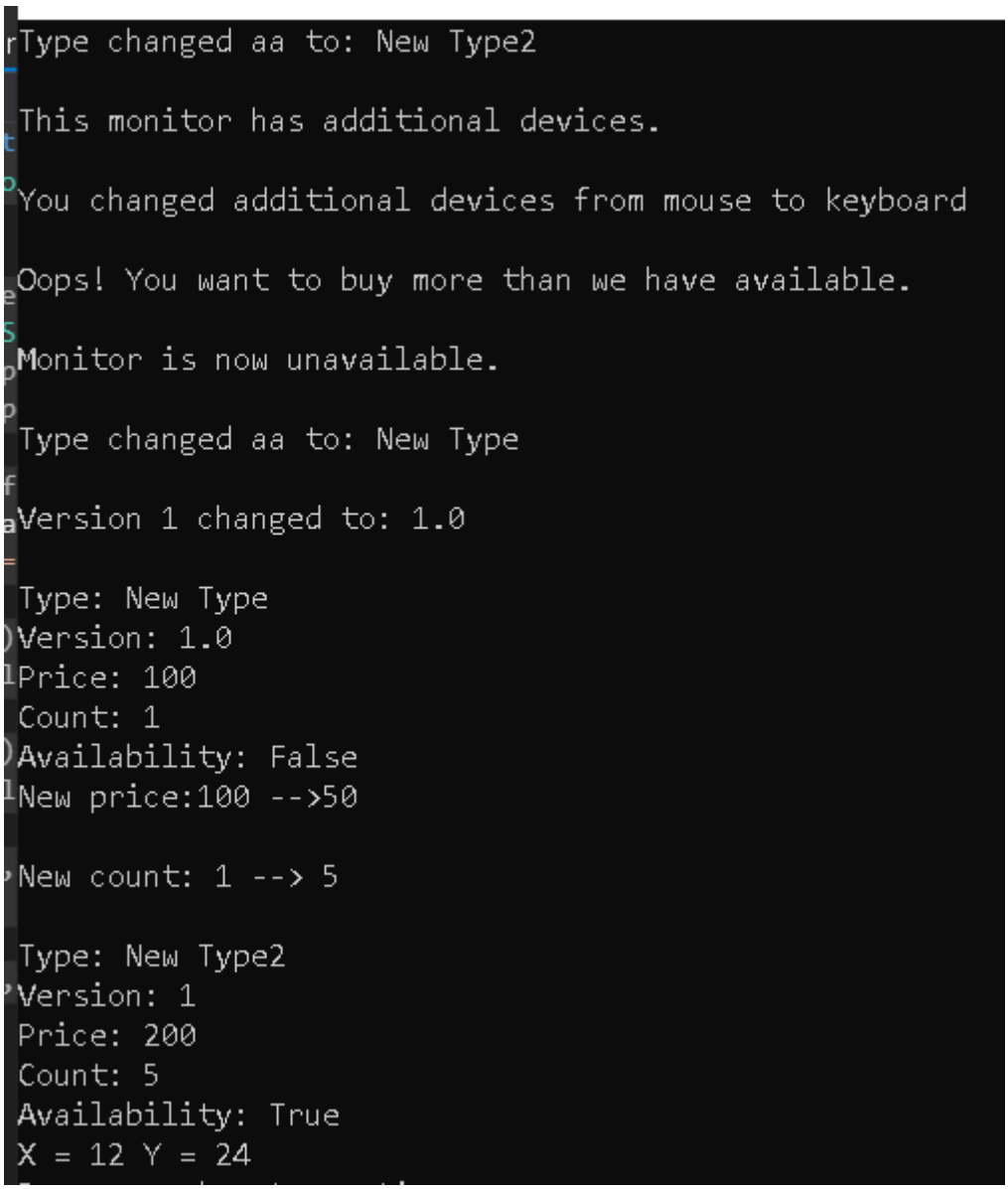
7. Існують одноріневе спадкування (клас успадковує властивості одного базового класу) і багаторівневе спадкування (клас успадковує властивості багатьох базових класів).

8. Небезпеки при множинному спадкуванні включають можливі конфлікти імен методів та атрибутів з різних базових класів. Щоб уникнути цих проблем, можна використовувати аліаси для методів, використовувати діамантовий паттерн (ромбовидну структуру спадкування) або взагалі обмежувати множинне спадкування.

9. Класи-домішки (mixins) - це класи, які містять невелику кількість методів або атрибутів і призначені для розширення функціональності інших класів, не змінюючи їхньої основної структури.

10. Функція super() використовується для виклику методів батьківського класу з класу-спадкоємця. Вона допомагає уникнути конфліктів імен методів при множинному спадкуванні і забезпечує коректне наслідування функціональності від базових класів.

Результат виконання програми




Висновок

На даній лабораторній роботі я навчився основним основам ООП на мові програмування PYTHON.