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-304 Мох М. П.

**Прийняв:**

Олексів М. В.

Львів – 2023

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

1. Написати та налагодити програму на мові Python згідно варіанту. Програма має задовольняти наступним вимогам:

* класи програми мають розміщуватися в окремих модулях в одному пакеті;
* точка входу в програму (main) має бути в окремому модулі;
* мають бути реалізовані базовий і похідний класи предметної області згідно варіанту;
* програма має містити коментарі.

1. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
2. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
3. Дати відповідь на контрольні запитання.

**Силка на GitHub:** https://github.com/MarianMoh/LabsMohKI-304

# Варіант № 12:

# Базовий клас – Водойма

# Похідний клас – Море

**Код програми:**

Файл main.py:

from sea import Sea  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 # Create a sea object with initial attributes  
 sea = Sea("Sea", 1000000, 15, 2500)  
  
 # Rename the sea and display its information  
 sea.rename("Black sea")  
 sea.showInfo()  
  
 # Add a season to the sea and display updated information  
 sea.addSeason()  
 sea.showInfo()  
  
 # Test method fish in a normal and error way  
 sea.fish()  
 sea.fish()  
  
 # Test method feed, after which we can use fish again  
 sea.feed()  
 sea.fish()  
  
 # Simulate the drying up of the sea  
 sea.dry()

Файл reservoir.py:

import re  
import sys  
import time  
  
from season import Season  
  
class Reservoir:  
 # Initialize the Reservoir object with a name, amount of water, and water temperature.  
 def \_\_init\_\_(self, name, amountOfWater, waterTemperature):  
 self.\_\_name = name  
 self.\_\_amountOfWater = amountOfWater  
 self.\_\_waterTemperature = waterTemperature  
 self.\_\_season = None  
  
 # Display information about the reservoir, including name, amount of water, water temperature,  
 # and the season (if available).  
 def showInfo(self):  
 print("\nName: ", self.\_\_name)  
 print("Amount of water: ", self.\_\_amountOfWater)  
 print("Water temperature: ", self.\_\_waterTemperature)  
  
 if self.\_\_season is not None:  
 print("Season: ", self.\_\_season.name)  
  
 # Rename the reservoir if the provided new name is valid (matches the specified pattern).  
 def rename(self, newName):  
 if re.match("^[a-zA-Z ]{3,}$", newName):  
 self.\_\_name = newName  
 print("\nNew name:", newName)  
 else:  
 print(f"\n- Error! New name \"{newName}\" is incorrect :(")  
  
 # Add a season to the reservoir if it doesn't already have one. Validate user input.  
 def addSeason(self):  
 if self.\_\_season is None:  
 while True:  
 try:  
 inputSeason = input("\nEnter a season: ")  
 season = Season[inputSeason.upper()]  
 break  
 except KeyError:  
 print(f"\n- Error! Wrong input \"{inputSeason}\"...")  
  
 self.\_\_season = season  
 else:  
 print("\n- Error! You already have a season: ", self.\_\_season.name)  
  
 # Simulate the drying up of the reservoir with a short delay and exit the program.  
 def dry(self):  
 time.sleep(0.5)  
 sys.exit(f"\nThe {self.\_\_name} dried up...")

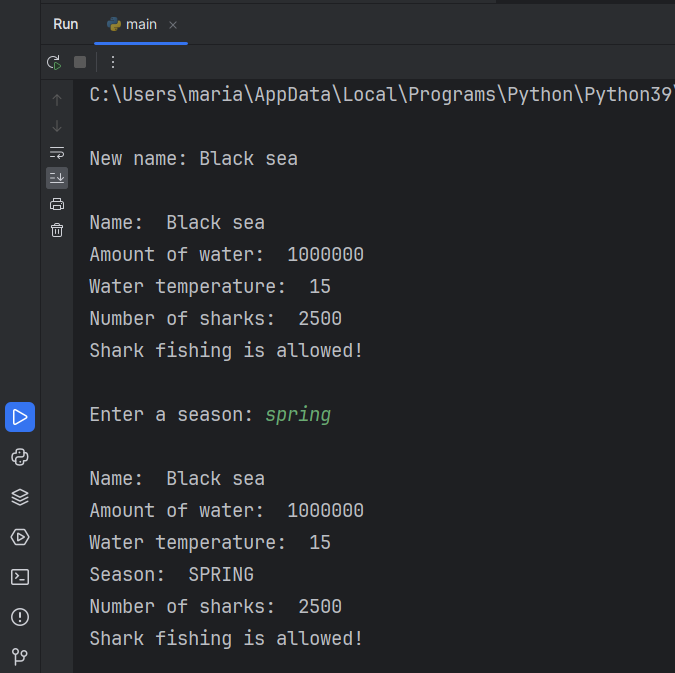
Файл sea.py:

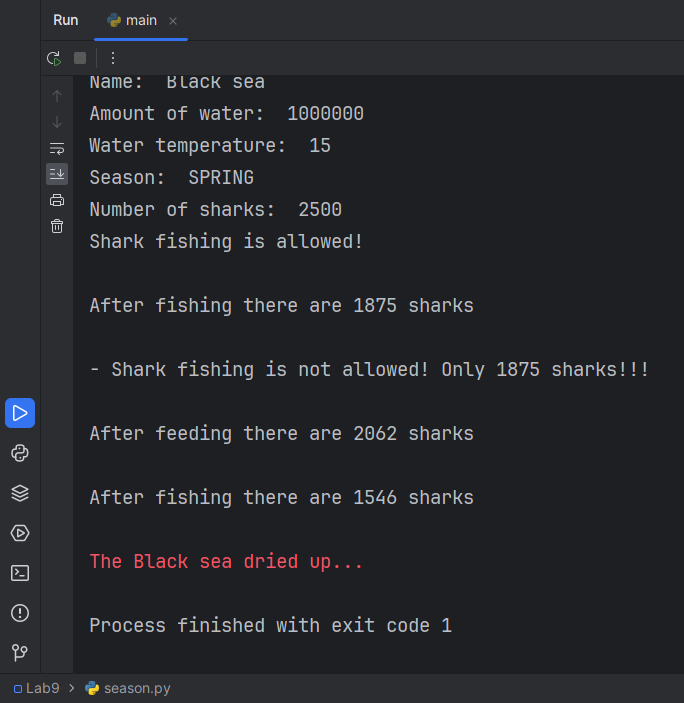
from reservoir import Reservoir  
  
class Sea(Reservoir):  
 # Initialize the Sea object, inheriting from Reservoir.  
 def \_\_init\_\_(self, name, amountOfWater, waterTemperature, numberOfSharks):  
 super().\_\_init\_\_(name, amountOfWater, waterTemperature)  
 self.\_\_numberOfSharks = numberOfSharks  
 self.\_\_sharkFishingIsAllowed = numberOfSharks > 2000  
  
 # Display information about the sea, including inherited reservoir information and  
 # specific sea attributes such as the number of sharks and shark fishing allowance.  
 def showInfo(self):  
 super().showInfo()  
 print("Number of sharks: ", self.\_\_numberOfSharks)  
  
 if self.\_\_sharkFishingIsAllowed:  
 print("Shark fishing is allowed!")  
 else:  
 print("- Shark fishing is not allowed!!!")  
  
 # Perform fishing in the sea, reducing the number of sharks and updating fishing allowance.  
 def fish(self):  
 if self.\_\_sharkFishingIsAllowed:  
 self.\_\_numberOfSharks = int(self.\_\_numberOfSharks \* 0.75)  
 self.isFishingAllowed()  
 print(f"\nAfter fishing there are {self.\_\_numberOfSharks} sharks")  
 else:  
 print(f"\n- Shark fishing is not allowed! Only {self.\_\_numberOfSharks} sharks!!!")  
  
 # Feed the sharks in the sea, increasing the number of sharks and updating fishing allowance.  
 def feed(self):  
 self.\_\_numberOfSharks = int(self.\_\_numberOfSharks \* 1.1)  
 self.isFishingAllowed()  
 print(f"\nAfter feeding there are {self.\_\_numberOfSharks} sharks")  
  
 # Check whether shark fishing is allowed based on the updated number of sharks.  
 def isFishingAllowed(self):  
 self.\_\_sharkFishingIsAllowed = self.\_\_numberOfSharks > 2000

Файл season.py:

from enum import Enum, auto  
  
# Define an enumeration 'Season' to represent the four seasons  
class Season(Enum):  
 WINTER = auto()  
 SPRING = auto()  
 SUMMER = auto()  
 AUTUMN = auto()

# Результат виконання програми:





**Відповіді на контрольні запитання:**

1. Що таке модулі?
   * Модулі в Python - це файли, які містять Python-код. Вони використовуються для організації коду у логічні групи, і можуть містити функції, класи, змінні та інші об'єкти.
2. Як імпортувати модуль?
   * import модуль
3. Як оголосити клас?
   * class МійКлас: # Тіло класу
4. Що може міститися у класі?
   * атрибути (змінні), методи (функції), конструктори, спеціальні методи (наприклад, init , str ), властивості та інше.
5. Як називається конструктор класу?
   * Конструктор класу має ім'я init . Він викликається при створенні нового об'єкта класу і використовується для ініціалізації атрибутів об'єкта.
6. Як здійснити спадкування?
   * class ПідКлас(БазовийКлас): # Тіло підкласу
7. Які види спадкування існують?
   * одиночне спадкування (коли підклас успадковує лише один базовий клас) та множинне спадкування (коли підклас успадковує більше одного базового класу).
8. Які небезпеки є при множинному спадкуванні, як їх уникнути?
   * Небезпеки при множинному спадкуванні включають в себе можливі конфлікти імен методів або атрибутів між базовими класами, що може призвести до непередбачуваної поведінки. Для уникнення цих проблем можна використовувати аліаси, викликати методи базових класів безпосередньо або використовувати композицію замість спадкування.
9. Що таке класи-домішки?
   * це класи, які містять певний функціонал і можуть бути використані для розширення функціональності інших класів. Вони не призначені для створення об'єктів, але можуть бути включені у інші класи за допомогою спадкування, щоб надати їм певну функціональність.
10. Яка роль функції super() при спадкуванні?
    * для виклику методів базового класу з підкласу. Вона допомагає уникнути явного вказівання імен базових класів та робить код більш гнучким при зміні структури спадкування. Наприклад, super(). init () викликає конструктор базового класу.

# Висновок:

У ході виконання даної лабораторної роботи, здобув важливі навички об'єктноорієнтованого програмування мовою Python. Ознайомився з ключовими аспектами цієї парадигми, включаючи створення та використання класів, роботу з об'єктами, та використання спадкування та поліморфізму для покращення ефективності програми.