Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный университет”

Кафедра ИИТ

Лабораторная работа №6

По дисциплине: “ЯП”

Тема: “Классы. Инкапсуляция. Наследование”

Выполнил:

Студент 2-го курса

Группы ПО-7

Смушко О.Р.

Проверил:

Бойко Д.О.

Брест, 2021

**Вариант 8**

**Цель работы:** Ознакомиться с принципом инкапсуляции.

Дополнительные цели:

1. Ознакомиться с конструкторами и деструкторами в Python

2. Ознакомиться с указателем self

3. Инкапсуляция в Python

4. Ознакомиться с использованием принципа наследования в Python

**Задание 1.**

1. Определить пользовательский класс – «Книга»

2. Определить счетчик

3. Определить в классе конструкторы с параметрами и без. Конструктор должен выводить сообщение о количестве объектов.

4. Определить в классе внешние компоненты-функции для получения и установки полей данных.

5. Написать демонстрационную программу, в которой объекты пользовательского класса создаются с помощью неявного использования конструкторов без параметров. 6. Показать в программе явное использование конструкторов с параметрами.

**Задание 2.**

1. Определить пользовательский класс в соответствии с вариантом задания.

2. Определить счетчик

3. Определить в классе конструкторы с параметрами и без. Конструктор должен выводить сообщение о количестве объектов.

4. Определить в классе внешние компоненты-функции для получения и установки полей данных.

5. Написать демонстрационную программу, в которой объекты пользовательского класса создаются с помощью неявного использования конструкторов без параметров.

6. Показать в программе явное использование конструкторов с параметрами.

**Код программы:**

from typing import Optional  
  
  
class Company:  
 \_\_counter = 0  
 \_\_name: str  
 \_\_square: float  
 \_\_worker\_amount: int  
  
 def \_\_new\_\_(cls, \*args, \*\*kwargs):  
 Company.\_\_counter += 1  
 print(f'Create {cls.\_\_name\_\_}')  
 return super().\_\_new\_\_(cls)  
  
 def \_\_init\_\_(self, name: Optional[str] = None, square: Optional[float] = 4500.90, amount: Optional[int] = 50001):  
 self.\_\_name = name  
 self.\_\_square = square  
 self.\_\_worker\_amount = amount  
  
 def \_\_del\_\_(self):  
 print(f'Del {self.\_\_class\_\_.\_\_name\_\_}')  
 Company.\_\_counter -= 1  
  
 def set\_name(self, name\_) -> None:  
 self.\_\_name = name\_  
  
 def set\_square(self, square\_) -> None:  
 self.\_\_square = square\_  
  
 def set\_amount(self, amount\_) -> None:  
 self.\_\_worker\_amount = amount\_  
  
 def get\_name(self) -> Optional[str]:  
 return self.\_\_name  
  
 def get\_square(self) -> Optional[float]:  
 return self.\_\_square  
  
 def get\_amount(self) -> Optional[int]:  
 return self.\_\_worker\_amount  
  
 def show(self):  
 print(f'Name: {self.\_\_name} | Square: {self.\_\_square} m2 | Amount of workers: {self.\_\_worker\_amount} '  
 f'| Counter: {self.\_\_counter}.')  
  
  
class Animal:  
 counter = 0  
 \_\_gender: str  
 \_\_color: str  
 \_\_age: int  
  
 def \_\_new\_\_(cls, \*args, \*\*kwargs):  
 Animal.counter += 1  
 print(f'Create {cls.\_\_name\_\_}')  
 return super().\_\_new\_\_(cls)  
  
 def \_\_init\_\_(self, gender: Optional[str] = '', color: Optional[str] = '', age: Optional[int] = 0):  
 self.\_\_gender = gender  
 self.\_\_color = color  
 self.\_\_age = age  
  
 def \_\_del\_\_(self):  
 print(f'Del {self.\_\_class\_\_.\_\_name\_\_}')  
 Animal.counter -= 1  
  
 def set\_gender(self, gender: str) -> None:  
 self.\_\_gender = gender  
  
 def set\_color(self, color: str) -> None:  
 self.\_\_color = color  
  
 def set\_age(self, age: int):  
 self.\_\_age = age  
  
 def get\_gender(self) -> Optional[str]:  
 return self.\_\_gender  
  
 def get\_color(self) -> Optional[str]:  
 return self.\_\_color  
  
 def get\_age(self) -> Optional[int]:  
 return self.\_\_age  
  
 def show(self) -> None:  
 print(f'Gender: {self.\_\_gender} | Color: {self.\_\_color} | Age: {self.\_\_age} years'  
 f' | Counter: {self.counter}.')  
  
 def input(self):  
 self.\_\_gender = str(input('Enter gender of the animal: '))  
 self.\_\_color = str(input('Enter color: '))  
 self.\_\_age = int(input('Enter age: '))  
  
  
class DomesticAnimal(Animal):  
 \_\_is\_milk: bool  
 \_\_weight: float  
  
 def \_\_init\_\_(self, gender: Optional[str] = '', color: Optional[str] = '', age: Optional[int] = 0,  
 weight: Optional[float] = 0, is\_milk: Optional[bool] = False):  
 super().\_\_init\_\_(gender, color, age)  
 self.\_\_is\_milk = is\_milk  
 self.\_\_weight = weight  
  
 def set\_is\_milk(self, is\_milk: bool) -> None:  
 self.\_\_is\_milk = is\_milk  
  
 def set\_weight(self, weight: float):  
 self.\_\_weight = weight  
  
 def get\_is\_milk(self) -> Optional[bool]:  
 return self.\_\_is\_milk  
  
 def get\_weight(self) -> Optional[float]:  
 return self.\_\_weight  
  
 def show(self) -> None:  
 print(f'Gender: {self.get\_gender()} | Color: {self.get\_color()} | Age: {self.get\_age()} years'  
 f' | Counter: {self.counter} | Milk? {self.\_\_is\_milk} | Weight: {self.\_\_weight}.')  
  
 def input(self):  
 self.set\_gender(str(input('Enter gender of the animal: ')))  
 self.set\_color(str(input('Enter color: ')))  
 self.set\_age(int(input('Enter age: ')))  
 check = str(input('Enter is there any milk: '))  
 if check.startswith('True'.lower()):  
 self.\_\_is\_milk = True  
 elif check.startswith('False'.lower()):  
 self.\_\_is\_milk = False  
 self.\_\_weight = float(input('Enter weight: '))  
  
  
class Dog(DomesticAnimal):  
 \_\_speed: float  
 \_\_breed: str  
  
 def \_\_init\_\_(self, gender: Optional[str] = '', color: Optional[str] = '', age: Optional[int] = 0,  
 weight: Optional[float] = 0, speed: Optional[float] = 0,  
 breed: Optional[str] = '', is\_milk: Optional[bool] = False):  
 super().\_\_init\_\_(gender, color, age, weight, is\_milk)  
 self.\_\_speed = speed  
 self.\_\_breed = breed  
  
 def set\_speed(self, speed: float) -> None:  
 self.\_\_speed = speed  
  
 def set\_breed(self, breed: str):  
 self.\_\_breed = breed  
  
 def get\_speed(self) -> Optional[float]:  
 return self.\_\_speed  
  
 def get\_breed(self) -> Optional[str]:  
 return self.\_\_breed  
  
 def show(self):  
 print(f'Gender: {self.get\_gender()} | Color: {self.get\_color()} | Age: {self.get\_age()} years'  
 f' | Counter: {self.counter} | Milk? {self.get\_is\_milk()} | Weight: {self.get\_weight()}'  
 f' | Speed: {self.\_\_speed} km/h | Breed: {self.\_\_breed}.')  
  
 def input(self):  
 self.set\_gender(str(input('Enter gender of the animal: ')))  
 self.set\_color(str(input('Enter color: ')))  
 self.set\_age(int(input('Enter age: ')))  
 check = str(input('Enter is there any milk: '))  
 if check.startswith('True'.lower()):  
 self.set\_is\_milk(True)  
 elif check.startswith('False'.lower()):  
 self.set\_is\_milk(False)  
 self.set\_weight(float(input('Enter weight: ')))  
 self.\_\_speed = float(input('Enter speed: '))  
 self.\_\_breed = str(input('Enter breed: '))  
  
  
def main():  
 def task\_1():  
 print('Task 1')  
 name = str(input('Enter company\'s name: '))  
 square = float(input('Enter square of the company: '))  
 amount = int(input('Enter amount of workers: '))  
  
 cmp1 = Company()  
 cmp1.set\_name(name)  
 cmp1.set\_square(square)  
 cmp1.set\_amount(amount)  
 cmp1.show()  
  
 cmp2 = Company("Apple", 1725.56, 110500)  
 cmp2.show()  
 cmp2.\_\_del\_\_()  
  
 cmp3 = Company()  
 cmp3.set\_name("Xiaomi")  
 cmp3.show()  
  
 def task\_2():  
  
 def check(val: str) -> bool:  
 if val.startswith('True'.lower()):  
 val = True  
 return val  
 elif val.startswith('False'.lower()):  
 val = False  
 return val  
 print('\nTask 2')  
 gender = str(input('Enter gender of the animal: '))  
 color = str(input('Enter color: '))  
 age = int(input('Enter age: '))  
  
 anm1\_1 = Animal(gender, color, age)  
 anm1\_1.show()  
 anm1\_2 = Animal("Male", "Green", 8)  
 anm1\_2.show()  
  
 is\_milk = check(str(input('Enter is there any milk: ')))  
 weight = float(input('Enter weight of the animal: '))  
 anm2\_1 = DomesticAnimal(gender, color, age, weight, is\_milk)  
 anm2\_1.show()  
 anm2\_2 = DomesticAnimal("Female", "Red", 11, 89.60, False)  
 anm2\_2.show()  
  
 speed = float(input('Enter speed: '))  
 breed = str(input('Enter breed: '))  
 anm3\_1 = Dog(gender, color, age, weight, speed, breed, is\_milk)  
 anm3\_1.show()  
 anm3\_2 = Dog("Male", "Brown", 2, 14.50, 55, "Labrador")  
 anm3\_2.show()  
 anm3\_2.set\_color('White')  
 print(f'Color: {anm3\_2.get\_color()}.')  
  
 animal\_3 = Dog()  
 animal\_3.input()  
 animal\_3.show()  
  
 task\_1()  
 task\_2()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Вывод:** В ходе лабораторной работы освоил основные способы работы с классами. Познакомился с понятиями инкапсуляция и наследование, а также реализовал их на практике.