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

Качан В.В.

Проверила:

Дряпко

Брест, 2021

**Вариант 5**

**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.

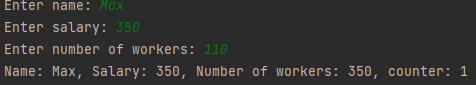
6. Показать в программе явное использование конструкторов с параметрами.

**Код программы:**

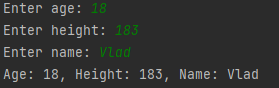
#Задание 1  
class hospital:  
 \_\_counter = 0  
 \_\_name = str()  
 \_\_salary = int()  
 \_\_work = int()  
  
 def \_\_init\_\_(self, name = '', salary = 0, workers = 0):  
 hospital.\_\_counter += 1  
 self.\_\_name = name  
 self.\_\_salary = salary  
 self.\_\_work = workers  
  
 def get\_name(self):  
 return self.\_\_name  
  
 def get\_salary(self):  
 return self.\_\_salary  
  
 def get\_workers(self):  
 return self.\_\_work  
  
 def set\_name(self, name):  
 self.\_\_name = name  
  
 def set\_salary(self, salary):  
 self.\_\_salary = salary  
  
 def set\_workers(self, workers):  
 self.\_\_work = workers  
  
 def show(self):  
 print(f'Name: {self.\_\_name}, Salary: {self.\_\_salary}, Number of workers: {self.\_\_salary}, counter: {self.\_\_counter}')  
  
  
hsp1 = hospital(input('Enter name: '), input('Enter salary: '), input('Enter number of workers: '))  
hsp1.show()  
  
#Задание 2:  
class person:  
 counter = 0  
 \_\_age = int()  
 \_\_height = int()  
 \_\_name = str()  
  
 def \_\_init\_\_(self, age = 0, height = 0, name = ''):  
 person.counter += 1  
 self.\_\_age = age  
 self.\_\_height = height  
 self.\_\_name = name  
  
 def \_\_del\_\_(self):  
 print('Вызов деструктора: '+str(person.counter))  
 person.counter -= 1  
  
 def get\_age(self):  
 return self.\_\_age  
  
 def get\_height(self):  
 return self.\_\_height  
  
 def get\_name(self):  
 return self.\_\_name  
  
 def set\_age(self, age):  
 self.\_\_age = age  
  
 def set\_height(self, height):  
 self.\_\_height = height  
  
 def set\_name(self, name):  
 self.\_\_name = name  
  
 def show(self):  
 print(f'Age: {self.\_\_age}, Height: {self.\_\_height}, Name: {self.\_\_name}')  
  
  
  
class Military(person):  
 \_\_rank = str()  
 \_\_num = int()  
  
 def \_\_init\_\_(self, age = 0, height = 0, name = '', rank = '', num = 0):  
 super().\_\_init\_\_(age, height, name)  
 self.\_\_rank = rank  
 self.\_\_num = num  
  
 def get\_rank(self):  
 return self.\_\_rank  
  
 def get\_num(self):  
 return self.\_\_num  
  
 def set\_rank(self, rank):  
 self.\_\_rank = rank  
  
 def set\_num(self, num):  
 self.\_\_num = num  
  
 def show(self):  
 print(f'Age: {self.get\_age()}, Height: {self.get\_height()}, Name: {self.get\_name()}, Rank: {self.\_\_rank}, Squad number: {self.\_\_num}')  
  
  
  
class recruit(Military):  
 \_\_date = str()  
 def \_\_init\_\_(self, age = 0, height = 0, name = '', rank = '', num = 0, date = ''):  
 super().\_\_init\_\_(age, height, name, rank, num)  
 self.\_\_date = date  
  
  
 def get\_date(self):  
 return self.\_\_date  
  
 def set\_date(self, date):  
 self.\_\_date = date  
  
 def show(self):  
 print(f'Age: {self.get\_age()}, Height: {self.get\_height()}, Name: {self.get\_name()}, Rank: {self.get\_rank()}, Squad number: {self.get\_num()}, Date of call: {self.\_\_date}')  
  
p1 = person(int(input('Enter age: ')), int(input('Enter height: ')), input('Enter name: '))  
p1.show()  
p2 = Military(int(input('Enter age: ')), int(input('Enter height: ')), input('Enter name: '), input('Enter rank: '), int(input('Enter squad number: ')))  
p2.show()  
p3 = recruit(int(input('Enter age: ')), int(input('Enter height: ')), input('Enter name: '), input('Enter rank: '), int(input('Enter squad number: ')), input('Enter date: '))  
p3.show()

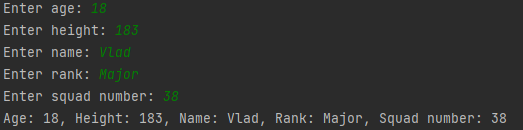
**Результат выполнения:**

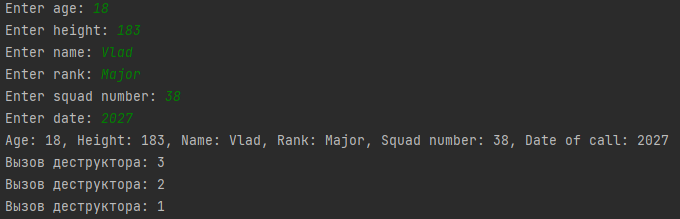
**Задание 1:**



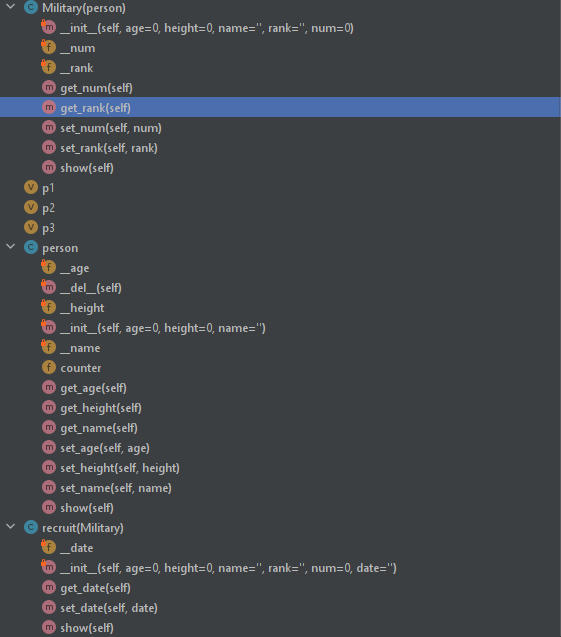
**Задание 2:**







**Диаграмма классов:**

****

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