Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №6

По дисциплине: «Языки программирования»

Вариант №3

Выполнили:

Студенты 2-го курса

Группы ПО-7

Гункевич И.А.

Проверила:

Дряпко А.В.

Брест 2021

**Цель работы:** Ознакомиться с принципом инкапсуляции, ознакомиться с использованием принципа наследования в Python.

**Ход работы:**

**Задание 1**

# Постановка задачи:

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.

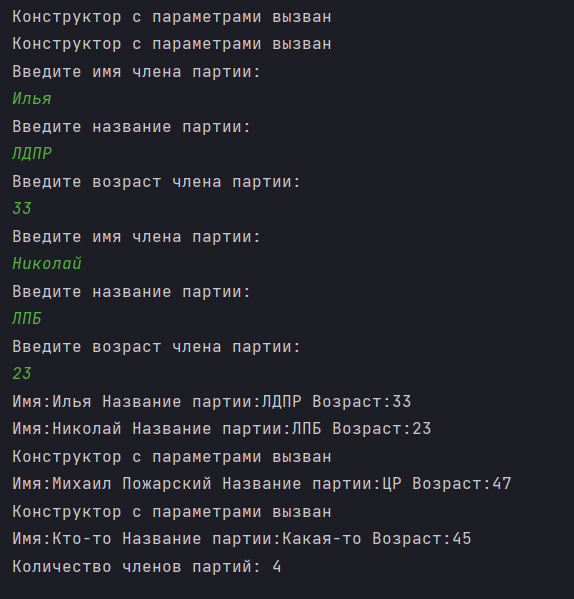
5. Написать демонстрационную программу, в которой объекты пользовательского класса создаются с помощью неявного использования конструкторов без параметров.

6. Показать в программе явное использование конструкторов с параметрами.

**Код программы:**

class PartyMember:  
 count = **0** \_\_name = str()  
 \_\_party = str()  
 \_\_age = int()  
 def \_\_init\_\_(self):  
 PartyMember.count += **1** print(PartyMember.count)  
  
 def \_\_init\_\_(self**,** nm=""**,** pr=""**,** ag=**0**):  
 PartyMember.count += **1** print("Конструктор с параметрами вызван")  
 try:  
 self.\_\_name = nm  
 self.\_\_party = pr  
 self.\_\_age = int(ag)  
 except ValueError as e:  
 print(e)  
  
 def set\_name(self**,** nm):  
 self.\_\_name = nm  
 def set\_party(self**,** pr):  
 self.\_\_party = pr  
 def set\_age(self**,** ag):  
 self.\_\_age = ag  
 def get\_name(self):  
 return self.\_\_name  
 def get\_party(self):  
 return self.\_\_party  
 def get\_age(self):  
 return self.\_\_age  
  
 def read(self):  
 try:  
 self.\_\_name = str(input("Введите имя члена партии:\n"))  
 self.\_\_party = str(input("Введите название партии:\n"))  
 self.\_\_age = int(input("Введите возраст члена партии:\n"))  
 except ValueError:  
 print("Ошибка ввода")  
  
 def show(self):  
 print("Имя:" + self.\_\_name + " Название партии:" + self.\_\_party + " Возраст:" + str(self.\_\_age))  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 y = PartyMember()  
 y1 = PartyMember()  
 y.read()  
 y1.read()  
 y.show()  
 y1.show()  
 x = PartyMember("Михаил Пожарский"**,** "ЦР"**, 47**)  
 x.show()  
 c = PartyMember()  
 c.set\_name("Кто-то")  
 c.set\_party("Какая-то")  
 c.set\_age(**45**)  
 c.show()  
 print("Количество членов партий:"**,** PartyMember.count)

**Результат:**



**Задание 2**

# Постановка задачи:

1. Построить модель предметной области в соответствии со своим вариантом (см. ниже)

2. Для каждого класса создать конструктор и деструктор, выдающий сообщение о своей работе

3. Для каждого класса создать внешние функции установки и получения полей данных

4. Для каждого класса разработать функции, позволяющие представить на экране значения полей данных

5. Для каждого класса разработать функции, позволяющие вводить с консоли значения полей данных

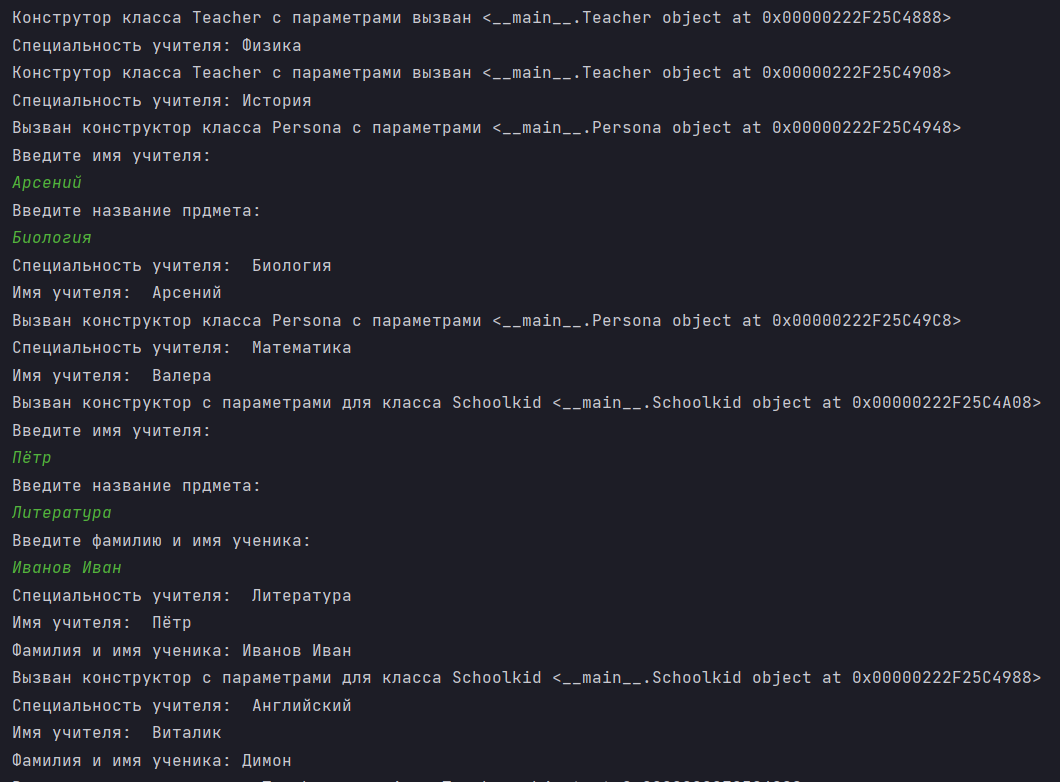
6. Написать демонстрационную программу, иллюстрирующую поочередный вызов конструкторов и деструкторов базового и производного классов

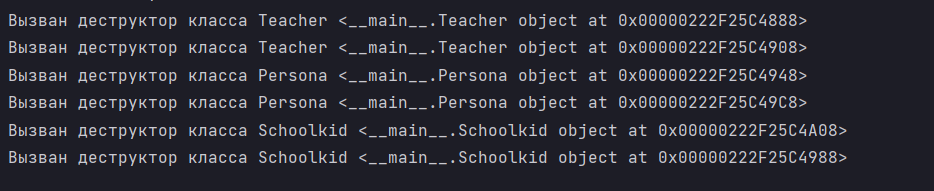
7. Построить диаграмму классов

**Код программы:**

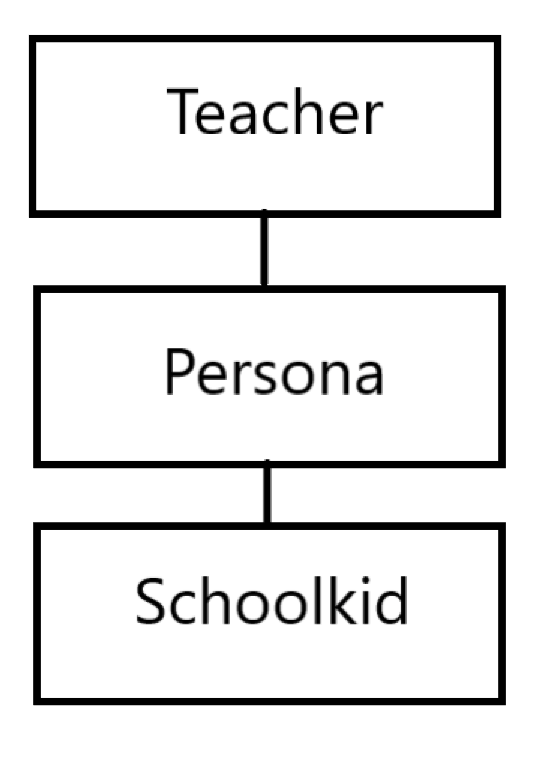
class Teacher:  
 subject = str()  
  
 def \_\_init\_\_(self):  
 print("Вызван конструктор без параметров Teacher"**,** self)  
  
 def \_\_init\_\_(self**,** sb=""):  
 print("Конструтор класса Teacher с параметрами вызван"**,** self)  
 try:  
 self.subject = sb  
 except ValueError:  
 print("Ошибка ввода")  
  
 def set\_subject(self**,** sb):  
 self.subject = sb  
  
 def get\_subject(self):  
 return self.subject  
  
 def \_\_del\_\_(self):  
 print("Вызван деструктор класса Teacher"**,** self)  
  
 def read(self):  
 try:  
 self.subject = str(input("Введите название предмета, которым обучает учитель:\n"))  
 except ValueError:  
 print("Ошибка ввода")  
  
 def show(self):  
 print("Специальность учителя:"**,** self.subject)  
  
  
class Persona(Teacher):  
 name = str()  
  
 def \_\_init\_\_(self):  
 print("Вызван конструктор без параметров класса Persona"**,** self)  
  
 def \_\_init\_\_(self**,** nm=""**,** sb=""):  
 print("Вызван конструктор класса Persona с параметрами"**,** self)  
 try:  
 self.name = nm  
 self.subject = sb  
 except ValueError:  
 print("Ошибка ввода")  
  
 def set\_name(self**,** nm):  
 self.name = nm  
  
 def get\_name(self):  
 return self.name  
  
 def \_\_del\_\_(self):  
 print("Вызван деструктор класса Persona"**,** self)  
  
 def read(self):  
 try:  
 self.name = input("Введите имя учителя:\n")  
 self.subject = input("Введите название прдмета:\n")  
 except ValueError:  
 print("Ошибка ввода")  
  
 def show(self):  
 print("Специальность учителя: "**,** self.subject)  
 print("Имя учителя: "**,** self.name)  
  
  
class Schoolkid(Persona):  
 fio = str()  
  
 def \_\_init\_\_(self):  
 print("Вызван конструктор класса Schoolkid без параметров"**,** self)  
  
 def \_\_init\_\_(self**,** nm=""**,** sb=""**,** fi=""):  
 print("Вызван конструктор с параметрами для класса Schoolkid"**,** self)  
 try:  
 self.name = nm  
 self.subject = sb  
 self.fio = fi  
 except ValueError:  
 print("Ошибка ввода")  
  
 def \_\_del\_\_(self):  
 print("Вызван деструктор класса Schoolkid"**,** self)  
  
 def set\_fi(self**,** fi):  
 self.fio = fi  
  
 def get\_fi(self):  
 return self.fio  
  
 def read(self):  
 try:  
 self.name = input("Введите имя учителя:\n")  
 self.subject = input("Введите название прдмета:\n")  
 self.fio = input("Введите фамилию и имя ученика:\n")  
 except ValueError:  
 print("Ошибка ввода")  
  
 def show(self):  
 print("Специальность учителя: "**,** self.subject)  
 print("Имя учителя: "**,** self.name)  
 print("Фамилия и имя ученика:"**,** self.fio)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 teacher1 = Teacher()  
 teacher1.set\_subject("Физика")  
 teacher1.show()  
 teacher2 = Teacher("История")  
 teacher2.show()  
 person1 = Persona()  
 person1.read()  
 person1.show()  
 person2 = Persona("Валера"**,** "Математика")  
 person2.show()  
 schoolkid1 = Schoolkid()  
 schoolkid1.read()  
 schoolkid1.show()  
 schoolkid2 = Schoolkid("Виталик"**,** "Английский"**,** "Димон")  
 schoolkid2.show()

**Результат:**





**Диаграмма классов:**



**Вывод:** я ознакомился с принципом инкапсуляции, ознакомился с использованием принципа наследования в Python.