Министерство образования Республики Беларусь

Учреждения образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

По дисциплине: «Языки программирования»

Тема: «Классы. Инкапсуляция. Наследование»

Выполнила:

Студентка 1 курса

Группа ПО-7

Фурсевич Д.С.

Проверил:

Бойко Д. О.

2021

**Цель:** ознакомиться с принципом инкапсуляции и наследования в Python.

**Ход работы**

Вариант 11

**Задание 1:**

Выполнить следующее:

1. Определить пользовательский класс «Дипломат».

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.

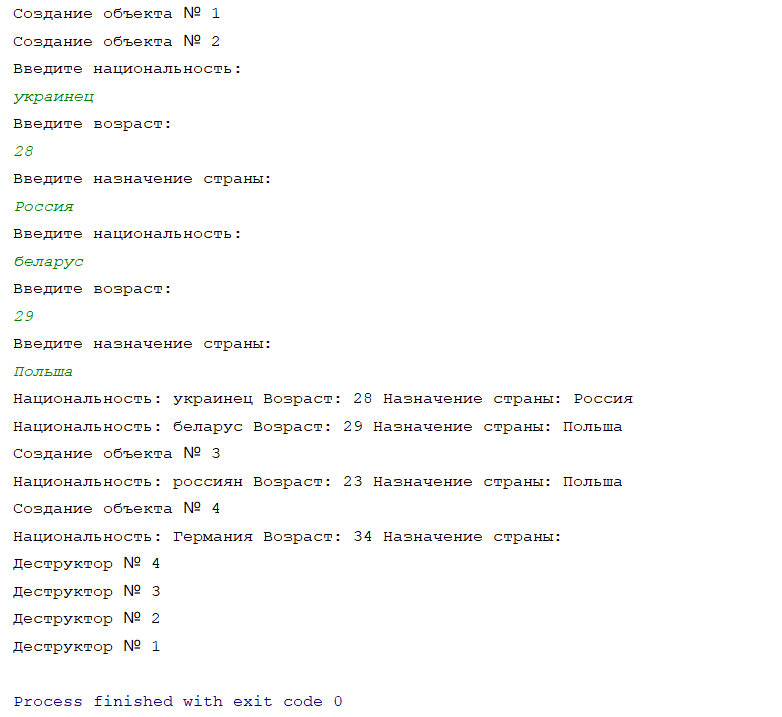
5. Написать демонстрационную программу, в которой объекты пользовательского класса создаются с помощью неявного использования конструкторов без параметров.

6. Показать в программе явное использование конструкторов с параметрами.

**Код программы:**

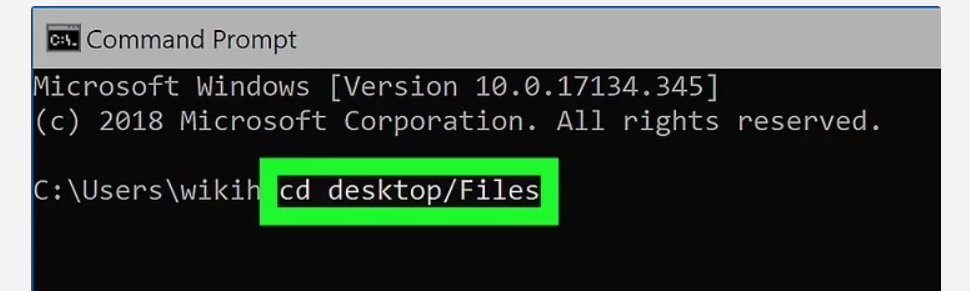
class Diplomat:  
 count = 0  
 \_\_nationality = str()  
 \_\_age = int()  
 \_\_job\_assignment\_country = str()  
  
 def \_\_init\_\_(self):  
 Diplomat.count += 1  
 print("Создание объекта №", Diplomat.count)  
  
 def \_\_init\_\_(self, nt='', ag=0, jac=''):  
 Diplomat.count += 1  
 print("Создание объекта №", Diplomat.count)  
 try:  
 self.\_\_nationality = nt  
 self.\_\_age = int(ag)  
 self.\_\_job\_assignment\_country = jac  
 except ValueError as e: # не совпадение типов данных  
 print(e)  
  
 def set\_natinality(self, nt):  
 self.\_\_nationality = nt  
  
 def set\_age(self, ag):  
 self.\_\_age = int(ag)  
  
 def set\_job\_assignment\_country(self, jas):  
 self.\_\_nationality = jas  
  
 def get\_natinality(self):  
 return self.\_\_nationality  
  
 def get\_age(self):  
 return self.\_\_age  
  
 def get\_job\_assignment\_country(self):  
 return self.\_\_nationality  
  
 def read(self):  
 try:  
 self.\_\_nationality = str(input("Введите национальность: \n"))  
 self.\_\_age = int(input("Введите возраст: \n"))  
 self.\_\_job\_assignment\_country = str(input("Введите назначение страны: \n"))  
 except ValueError as e:  
 print("input error")  
  
 def show(self):  
 print("Национальность: " + self.\_\_nationality + " Возраст: " + str(self.\_\_age) + " Назначение страны: " + self.\_\_job\_assignment\_country)  
  
 def \_\_del\_\_(self):  
 print("Деструктор №", Diplomat.count)  
 Diplomat.count -= 1  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 y = Diplomat()  
 y1 = Diplomat()  
 y.read()  
 y1.read()  
 y.show()  
 y1.show()  
  
 x = Diplomat("россиян", 23, "Польша")  
 x.show()  
 c = Diplomat()  
 c.set\_natinality("итальянец")  
 c.set\_age(34)  
 c.set\_job\_assignment\_country("Германия")  
 c.show()

**Выполнение программы:**

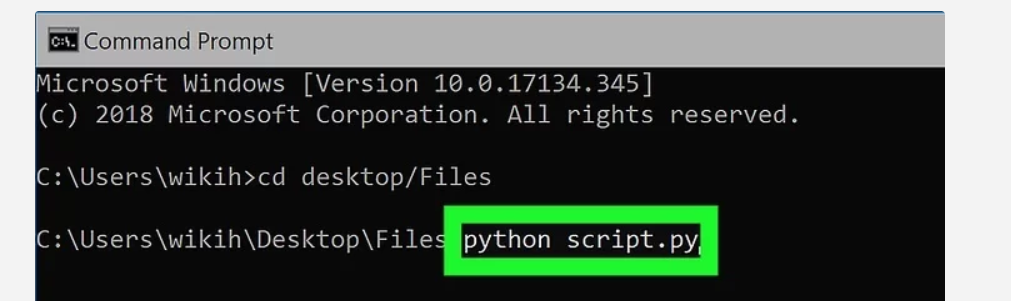


**Запуск приложения через командную строку:**

**1.**



**2.**



**Задание 2:**

Для своего варианта выполнить следующее:

1. Построить модель предметной области **Музыкант-Сотрудник-Скрипач**

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

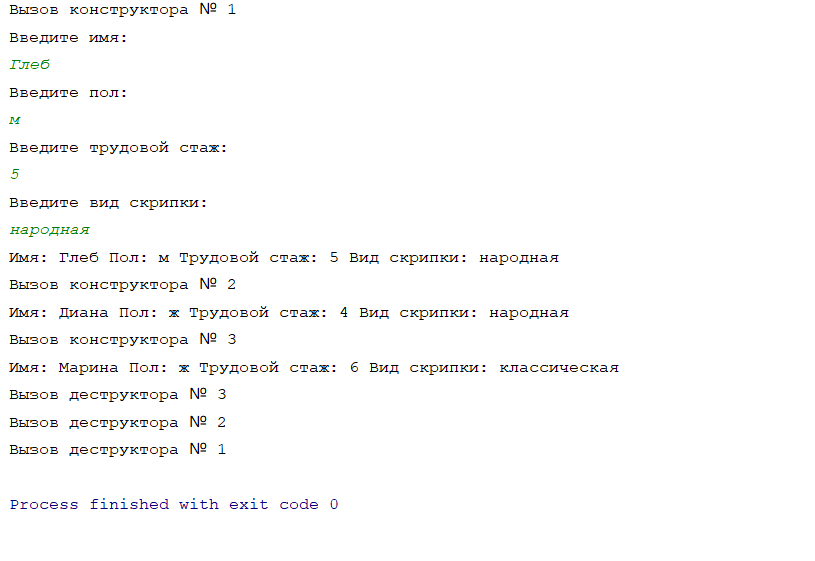
6. Написать демонстрационную программу, иллюстрирующую поочередный вызов конструкторов и деструкторов базового и производного классов

7. Построить диаграмму классов

**Код программы:**

class Worker:  
 count = 0  
   
 def \_\_init\_\_(self, nm = '', fl = ''):  
 Worker.count += 1  
 print("Вызов конструктора №", Worker.count)  
 try:  
 self.\_name = nm  
 self.\_floor = fl  
 except ValueError as e: # не совпадение типов данных  
 print(e)  
  
  
 def set\_name(self, nm):  
 self.\_name = nm  
  
 def set\_floor(self, fl):  
 self.\_floor = fl  
  
 def get\_name(self):  
 return self.\_name  
  
 def get\_floor(self):  
 return self.\_floor  
  
 def read(self):  
 try:  
 self.\_name = str(input("Введите имя: \n"))  
 self.\_floor = str(input("Введите пол: \n"))  
 except ValueError as e:  
 print("input error")  
  
 def show(self):  
 print("Имя: " + self.\_name + " Пол: " + self.\_floor)  
  
 def \_\_del\_\_(self):  
 print("Вызов деструткора №", Worker.count)  
 Worker.count -= 1  
  
class Musician(Worker):  
   
  
 def \_\_init\_\_(self, nm='', fl='', we=0):  
 Musician.count += 1  
 print("Вызов конструктора №", Musician.count)  
 try:  
 self.\_name = nm  
 self.\_floor = fl  
 self.\_work\_experience = int(we)  
 except ValueError as e: # не совпадение типов данных  
 print(e)  
  
  
 def set\_work\_experience(self, we):  
 self.\_work\_experience = int(we)  
  
 def get\_work\_experience(self):  
 return self.\_work\_experience  
  
  
 def read(self):  
 try:  
 self.\_name = str(input("Введите имя: \n"))  
 self.\_floor = str(input("Введите пол: \n"))  
 self.\_work\_experience = int(input("Введите трудовой стаж: \n"))  
 except ValueError as e:  
 print("input error")  
  
 def show(self):  
 print("Имя: " + self.\_name + " Пол: " + self.\_floor + " Трудовой стаж: " + str(self.\_work\_experience))  
  
 def \_\_del\_\_(self):  
 print("Вызов деструткора №", Musician.count)  
 Musician.count -= 1  
  
  
class Violinist(Musician):  
  
 def \_\_init\_\_(self, nm='', fl='', we=0, tp=''):  
 Violinist.count += 1  
 print("Вызов конструктора №", Violinist.count)  
 try:  
 self.\_name = nm  
 self.\_floor = fl  
 self.\_work\_experience = int(we)  
 self.\_type = tp  
 except ValueError as e: # не совпадение типов данных  
 print(e)  
  
 def set\_type(self, tp):  
 self.\_type = tp  
  
 def get\_type(self):  
 return self.\_type  
  
 def read(self):  
 try:  
 self.\_name = str(input("Введите имя: \n"))  
 self.\_floor = str(input("Введите пол: \n"))  
 self.\_work\_experience = int(input("Введите трудовой стаж: \n"))  
 self.\_type = str(input("Введите вид скрипки: \n"))  
 except ValueError as e:  
 print("input error")  
  
 def show(self):  
 print("Имя: " + self.\_name + " Пол: " + self.\_floor + " Трудовой стаж: " + str(self.\_work\_experience) + " Вид скрипки: " + self.\_type)  
  
 def \_\_del\_\_(self):  
 print("Вызов деструктора №", Violinist.count)  
 Violinist.count -= 1  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 B0 = Violinist()  
 B0.read()  
 B0.show()  
  
 B1 = Violinist("Диана", "ж", "4", "народная")  
 B1.show()  
  
 B2 = Violinist()  
 B2.set\_name("Марина")  
 B2.set\_floor("ж")  
 B2.set\_work\_experience(6)  
 B2.set\_type("классическая")  
 B2.show()

**Выполнение программы:**



**Вывод:** ознакомилась с принципом инкапсуляции и наследования в Python.