Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №6

за 1 семестр

По дисциплине: «Языки программирования»

Тема: «Классы. Инкапсуляция»

Выполнила:

Савицкая А.Ю., ПО-7

Проверил:

Бойко Д.О.

Брест, 2021

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

**Задание 1.**

Вариант 16

Для своего варианта (класс Отдел кадров) выполнить следующее:

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.

**Текст программы:**

class Otdel\_kadrov:

count = 0

name = str()

age = int()

salary = int()

def \_\_init\_\_(self):

Otdel\_kadrov.count += 1

print ("Количество: ", Otdel\_kadrov.count)

def \_\_init\_\_(self, name = "", age = 0, salary = 0):

Otdel\_kadrov.count += 1

print ("Количество: ", Otdel\_kadrov.count)

try:

self.name = name

self.age = age

self.salary = salary

except ValueError as error:

print (error)

def set\_name(self, name):

self.name = name

def set\_age(self, age):

self.age = age

def set\_salary(self, salary):

self.salary = salary

def get\_name(self):

return self.name

def get\_age(self):

return self.age

def get\_salary(self):

return self.salary

def reading(self):

try:

self.name = input("Введите имя: ")

self.age = int(input("Введите возраст: "))

self.salary = int(input("Введите зарплату ( в USD ): "))

except ValueError:

print ("Ошибка ввода")

def show(self):

print("\nИмя: ", self.name, "\nВозраст", self.age, "\nЗарплата: ", self.salary, "\n")

member1 = Otdel\_kadrov()

member1.set\_name("Иванов Иван")

member1.set\_age(35)

member1.set\_salary(600)

member1.show()

member2 = Otdel\_kadrov("Петров Пётр", 27, 550)

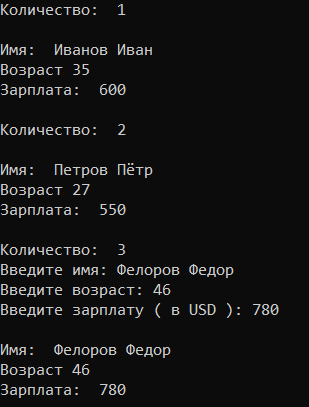
member2.show()

member3 = Otdel\_kadrov()

member3.reading()

member3.show()

**Результаты тестирования:**



**Задание 2.**

Вариант 11

Для своего варианта выполнить следующее:

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

**Текст программы:**

class Employee:

name = str()

def \_\_init\_\_(self):

print ("Конструктор класса Employee без параметров ", self)

def \_\_init\_\_(self, name = ""):

print ("Конструктор класса Employee с параметрами ", self)

try:

self.name = name

except ValueError as error:

print(error)

def \_\_del\_\_(self):

print ("Деструтор класса Employee", self)

def set\_name(self, name):

self.name = name

def get\_name(self):

return self.name

def reading (self):

try:

self.name = input("\nВведите имя: ")

except ValueError:

print("Ошибка ввода")

def show(self):

print("\nИмя: ", self.name)

class Musician(Employee):

last\_film = str()

def \_\_init\_\_(self):

print ("Конструктор класса Actor без параметров ", self)

def \_\_init\_\_(self, name = "", member = ""):

print ("Конструктор класса Actor с параметрами ", self)

try:

self.name = name

self.member = member

except ValueError as error:

print(error)

def \_\_del\_\_(self):

print ("Деструтор класса Actor", self)

def set\_member(self, member = ""):

self.member = member

def get\_member(self):

return self.member

def reading (self):

try:

self.name = input("\nВведите имя: ")

self.member = input("Является ли членом оркестра?: ")

except ValueError:

print("Ошибка ввода")

def show(self):

print("\nИмя: ", self.name)

print("Член оркестра: ", self.member, "\n")

class Violinist(Musician):

subordinates = str()

def \_\_init\_\_(self):

print ("Конструктор класса Violinist без параметров ", self)

def \_\_init\_\_(self, name = "", member = "", melody = ""):

print ("Конструктор класса Violinist с параметрами ", self)

try:

self.name = name

self.member = member

self.melody = melody

except ValueError as error:

print(error)

def \_\_del\_\_(self):

print ("Деструтор класса Violinist", self)

def set\_melody(self, melody):

self.melody = melody

def get\_melody(self):

return self.melody

def reading (self):

try:

self.name = input("\nВведите имя: ")

self.member = input("Является ли членом оркестра?: ")

self.melody = input("Введите сольное произведение для скрипки: ")

except ValueError:

print("Ошибка ввода")

def show(self):

print("\nИмя: ", self.name)

print("Член оркестра: ", self.member)

print("Сольное произведение для скрипки: ", self.melody, "\n")

employee1 = Employee()

employee1.set\_name("Подчиненный 1")

employee1.show()

employee2 = Employee("Подчиненный 2")

employee2.show()

musician1 = Musician()

musician1.reading()

musician1.show()

musician2 = Musician("Музыкант 2", "Да")

musician2.show()

violinist1 = Violinist()

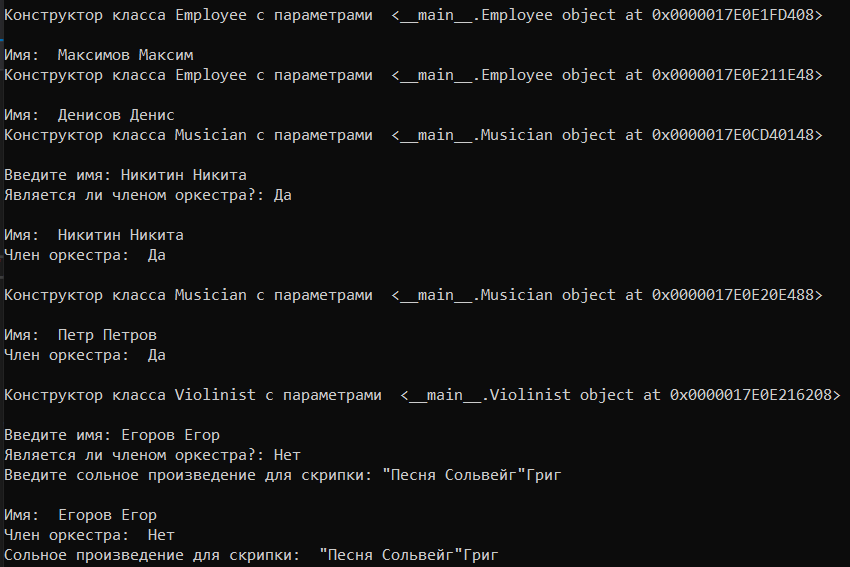
violinist1.reading()

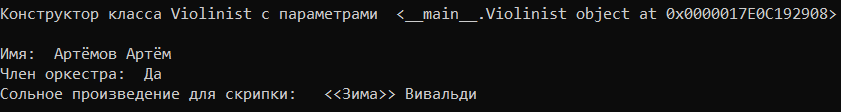
violinist1.show()

violinist2 = Violinist("Артёмов Артём", "Да", " <<Зима>> Вивальди ")

violinist2.show()

**Результаты тестирования:**





**Диаграмма классов:**



**Вывод:** ознакомилась с основами файловой системы языка Python.