Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №6**

По дисциплине: “Языки программирования”

Тема: «Классы. Инкапсуляция. Наследование»**Вариант №10**

**Выполнил**:

студент 2 курса

группы ПО-7

Курмыса Е.Е.

**Проверила:**

Дряпко А. В.

Брест 2021

**Цель работы:** ознакомиться с принципом инкапсуляции, конструкторами и деструкторами, а также с self в Python; принципом наследования классов в Python.

**Задание 1. Классы и инкапсуляция:** Определить пользовательский класс «Квартира», счётчик объектов данного класса, конструкторы с параметрами и без них (должны выводить информацию о количестве объектов), а также внешние компоненты-для получения и установки полей данных объекта класса. Написать программу, создающую объекты класса с помощью неявного использования конструктора без параметров и с ними.

**Код (Python)**

class Flat:

count = 0 # количество созданных объектов класса

\_\_address = str() # адрес квартиры

\_\_area = int() # площадь квартиры

\_\_floor = int() # этаж

# конструктор без параметров (по умолчанию)

# рекомендуется объявлять лишь один конструктор в классе + можно сразу подставлять значения по умолчанию

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

# self.count += 1

# print('Создан новый объект. Количество созданных объектов:', self.count)

def \_\_init\_\_(self, address = "", area = 0, floor = 0): # конструктор с параметрами; если их нет, то ставятся по умолчанию

try:

Flat.count += 1

print('\nСоздан новый объект. Количество созданных объектов:', self.count, end='\n\n')

self.\_\_address = address

self.\_\_area = int(area)

self.\_\_floor = int(floor)

except ValueError as e:

print(e)

def get\_address(self):

return self.\_\_address

def get\_area(self):

return self.\_\_area

def get\_floor(self):

return self.\_\_floor

def set\_address(self, address):

self.\_\_address = address

def set\_area(self, area):

self.\_\_area = area

def set\_floor(self, floor):

self.\_\_floor = floor

def read(self):

try:

self.\_\_address= input('Введите адрес квартиры: ')

self.\_\_area = int(input('Введите площадь квартиры (в кв. м): '))

self.\_\_floor = input('Введите номер этажа, на котором расположена квартира: ')

except ValueError as e:

print(e)

def show(self):

print('\n' + '-' \* 20)

print('Адрес:', self.\_\_address, '\nПлощадь:', self.\_\_area, 'кв. м\nНомер этажа:', self.\_\_floor)

print('-' \* 20)

if \_\_name\_\_ == "\_\_main\_\_":

grad1 = Flat() # вызов конструктора по умолчанию

grad1.read() # запись данных в объект через функцию

grad1.show() # вывод данных в консоль

grad2 = Flat("Москва, ул. Останкинская, д. 14, кв. 81", 129, 9) # вызов конструктора с параметрами

print('Адрес 2-ой квартиры:', grad2.get\_address())

print('Номер этажа 2-ой квартиры:', grad2.get\_floor()) # получение данных при помощи геттеров

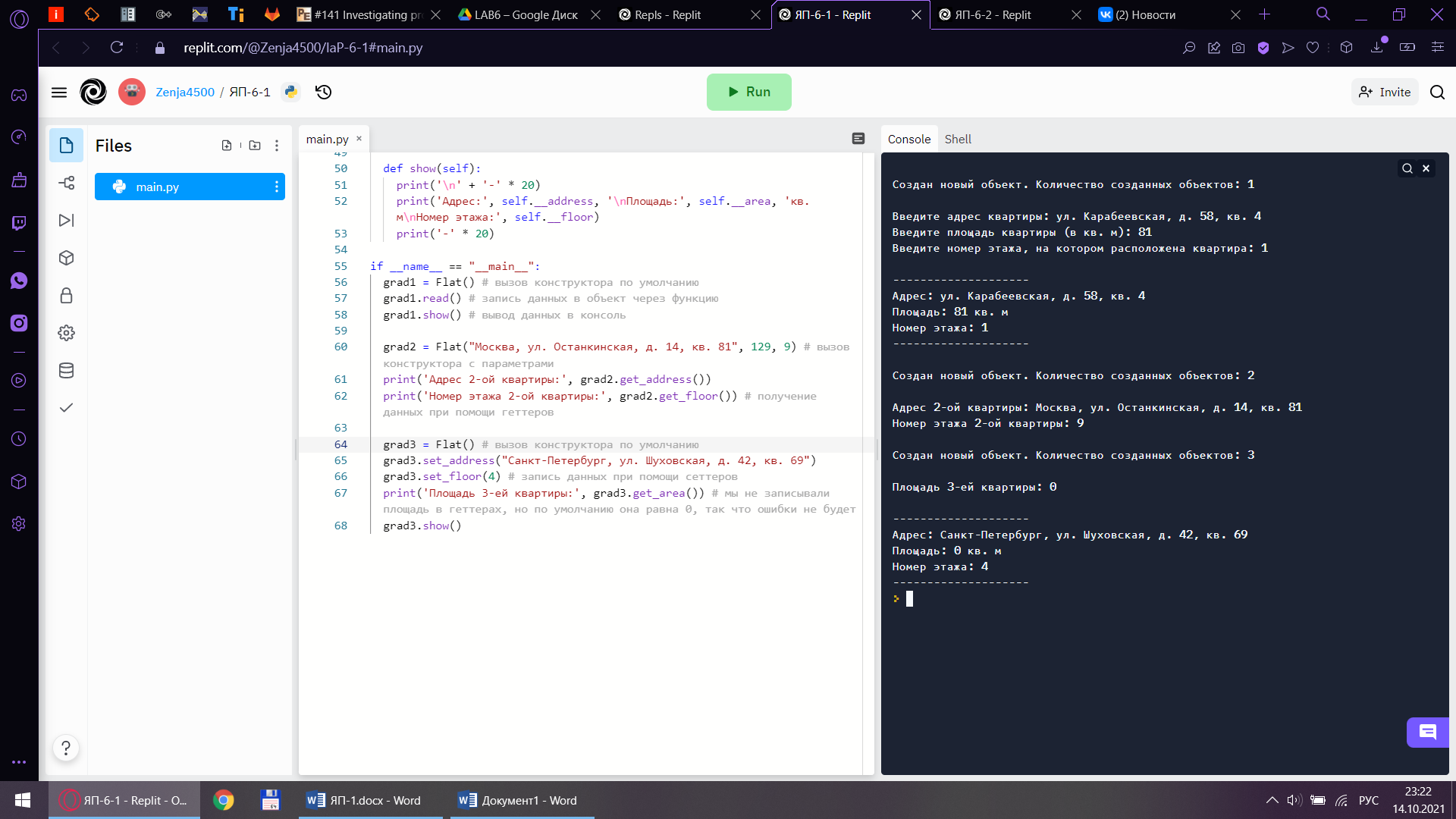
grad3 = Flat() # вызов конструктора по умолчанию

grad3.set\_address("Санкт-Петербург, ул. Шуховская, д. 42, кв. 69")

grad3.set\_floor(4) # запись данных при помощи сеттеров

print('Площадь 3-ей квартиры:', grad3.get\_area()) # мы не записывали площадь в геттерах, но по умолчанию она равна 0, так что ошибки не будет

grad3.show()



**Задание 2. Наследование:** построить модель предметной области «Студент-Учащийся-Магистрант», классы к ней, конструкторы и деструкторы, внешние функции установки/получения данных полей.

В данном случае стоит сделать следующий ряд: Учащийся -> Студент -> Магистрант, причём поля будут следующие:

* Учащийся: имя, возраст, город;
* Студент: поля Учащегося, курс, специальность и университет;
* Магистрант: поля Студента, тема проекта и ФИО научного руководителя.

**Код (Python)**

**pupil.py: // модуль с классом «Учащийся»**

class Pupil:

count = 0

\_\_pupil\_number = int()

\_name = str()

\_age = int()

\_town = str()

def \_\_init\_\_(self, name='Учащийся', age=0, town='Город\_учащийся'):

Pupil.count += 1

self.\_\_pupil\_number = Pupil.count

print('Вызов конструктора класса Pupil No.', self.\_\_pupil\_number)

self.\_name = name

self.\_age = int(age)

self.\_town = town

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

print('Удаляется объект класса Pupil No.', self.\_\_pupil\_number)

def get\_name(self):

return self.\_name

def get\_age(self):

return self.\_age

def get\_town(self):

return self.\_town

def set\_name(self, name):

self.\_name = name

def set\_age(self, age):

self.\_age = age

def set\_town(self, town):

self.\_town = town

def read(self):

try:

self.\_name = input('Введите имя учащегося: ')

self.\_age = int(input('Введите возраст учащегося: '))

self.\_town = input('Введите город учащегося: ')

except ValueError as e:

print(e)

def show(self):

print('Имя:', self.\_name, '\nВозраст:', self.\_age, '\nГород:', self.\_town, end='\n\n')

**student.py // модуль с классом «Студент»**

from pupil import Pupil

class Student(Pupil):

count = 0

\_\_student\_number = int()

\_course = int()

\_specialty = str()

\_university = str()

def \_\_init\_\_(self, name='Студент', age=0, town='Город\_студент', course=0, specialty='Специальность\_студент', university='Университет\_студент'):

Student.count += 1

self.\_\_student\_number = Student.count

print('Вызов конструктора класса Student No.', self.\_\_student\_number)

super().\_\_init\_\_(name, age, town)

self.\_course = int(course)

self.\_specialty = specialty

self.\_university = university

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

print('Удаляется объект класса Student No.', self.\_\_student\_number)

super().\_\_del\_\_()

def get\_course(self):

return self.\_course

def get\_specialty(self):

return self.\_specialty

def get\_university(self):

return self.\_university

def set\_course(self, course):

self.\_course = course

def set\_specialty(self, specialty):

self.\_specialty = specialty

def set\_university(self, university):

self.\_university = university

def read(self):

try:

self.\_name = input('Введите имя студента: ')

self.\_age = int(input('Введите возраст студента: '))

self.\_town = input('Введите город студента: ')

self.\_course = int(input('Введите курс студента: '))

self.\_specialty = input('Введите специальность студента: ')

self.\_university = input('Введите университет студента: ')

except ValueError as e:

print(e)

def show(self):

print(

'Имя:', self.\_name,

'\nВозраст:', self.\_age,

'\nГород:', self.\_town,

'\nКурс:', self.\_course,

'\nСпециальность:', self.\_specialty,

'\nУниверситет:', self.\_university, end='\n\n')

**undergrad.py // модуль с классом «Магистрант»**

from student import Student

class Undergrad(Student):

count = 0

\_\_undergrad\_number = 0

\_project\_theme = str() # тема проекта

\_director = str() # научный руководитель

def \_\_init\_\_(self, name='Магистрант', age=0, town='Город\_магистрант', course=0, specialty='Специальность\_магистрант', university='Университет\_магистрант', project\_theme='Темапроекта\_магистрант', director='Научрук\_магистрант'):

Undergrad.count += 1

self.\_\_undergrad\_number = Undergrad.count

print('Вызов конструктора класса Undergrad No.', self.\_\_undergrad\_number)

super(Undergrad, self).\_\_init\_\_(name, age, town, course, specialty, university)

self.\_project\_theme = project\_theme

self.\_director = director

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

print('Удаляется объект класса Undergrad No.', self.\_\_undergrad\_number)

super().\_\_del\_\_()

def get\_project\_theme(self):

return self.\_project\_theme

def get\_director(self):

return self.\_director

def set\_project\_theme(self, project\_theme):

self.\_project\_theme = project\_theme

def set\_director(self, director):

self.\_director = director

def read(self):

try:

self.\_name = input('Введите имя магистранта: ')

self.\_age = int(input('Введите возраст магистранта: '))

self.\_town = input('Введите город магистранта: ')

self.\_course = int(input('Введите курс магистранта: '))

self.\_specialty = input('Введите специальность магистранта: ')

self.\_university = input('Введите университет магистранта: ')

self.\_project\_theme = input('Введите тему научного проекта: ')

self.\_director = input('Введите ФИО начуного руководителя: ')

except ValueError as e:

print(e)

def show(self):

print(

'Имя:', self.\_name,

'\nВозраст:', self.\_age,

'\nГород:', self.\_town,

'\nКурс:', self.\_course,

'\nСпециальность:', self.\_specialty,

'\nУниверситет:', self.\_university,

'\nТема проекта:', self.\_project\_theme,

'\nФИО научного руководителя:', self.\_director, end='\n\n')

**main.py // основной модуль, в котором вводятся и выводятся все данные**

from pupil import Pupil

from student import Student

from undergrad import Undergrad

if \_\_name\_\_ == "\_\_main\_\_":

person1 = Pupil()

person1.read()

print('Данные по первому учащемуся:')

person1.show()

person2 = Pupil('Егор', 13, 'Кобрин')

print('Имя второго учащегося:', person2.get\_name())

print('Город второго учащегося:', person2.get\_town())

person3 = Student()

person4 = Student('Артём', 18, 'Брест')

person3.read()

person4.set\_specialty('ИИ')

person4.set\_course(1)

person4.set\_university('БрГУ им. Пушкина')

print('Данные по третьему учащемуся:')

person3.show()

print('Данные по четвёртому учащемуся:')

person4.show()

person5 = Undergrad()

print('Данные по пятому учащемуся (дефолтные):')

person5.show()

person5.read()

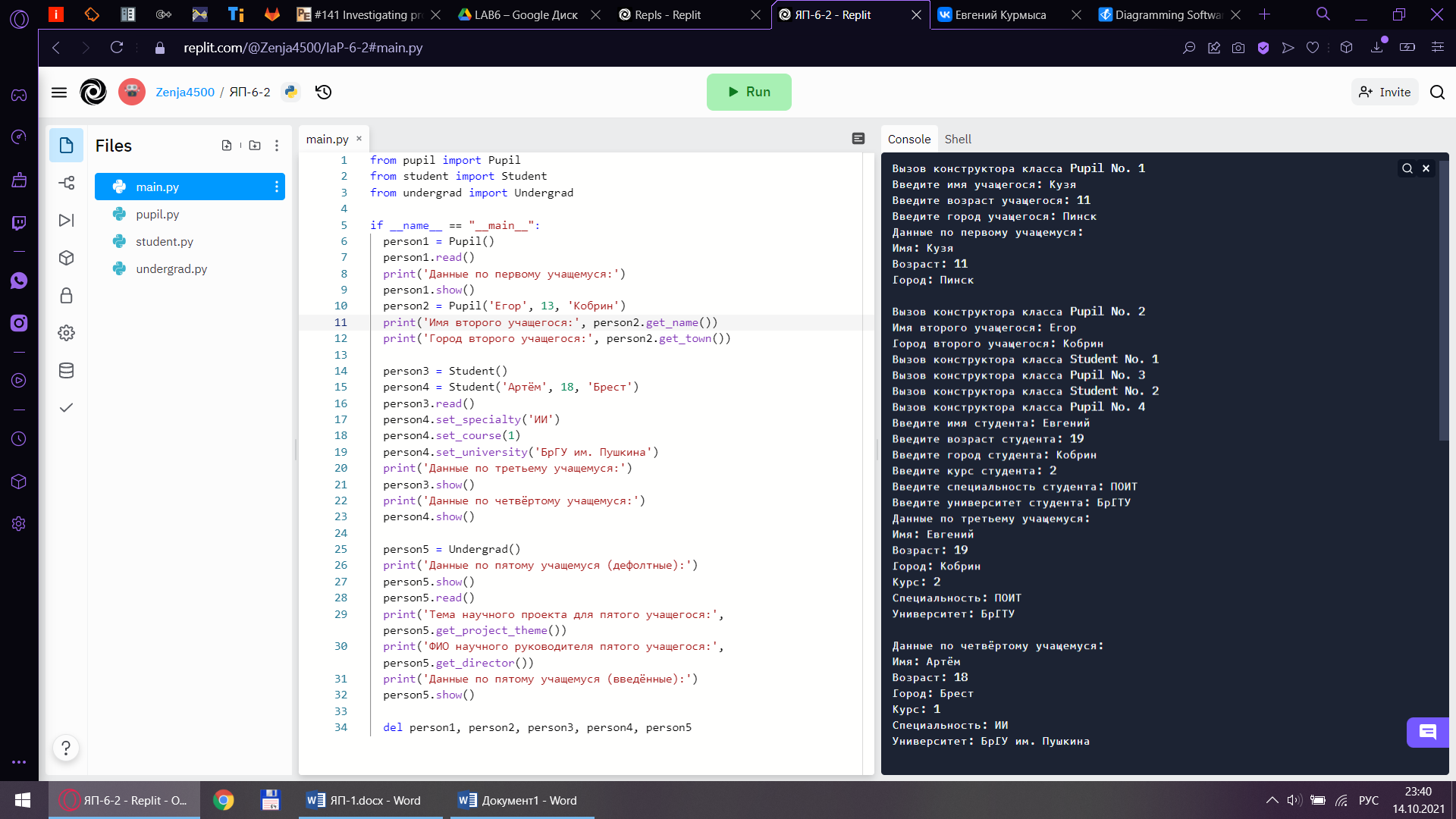
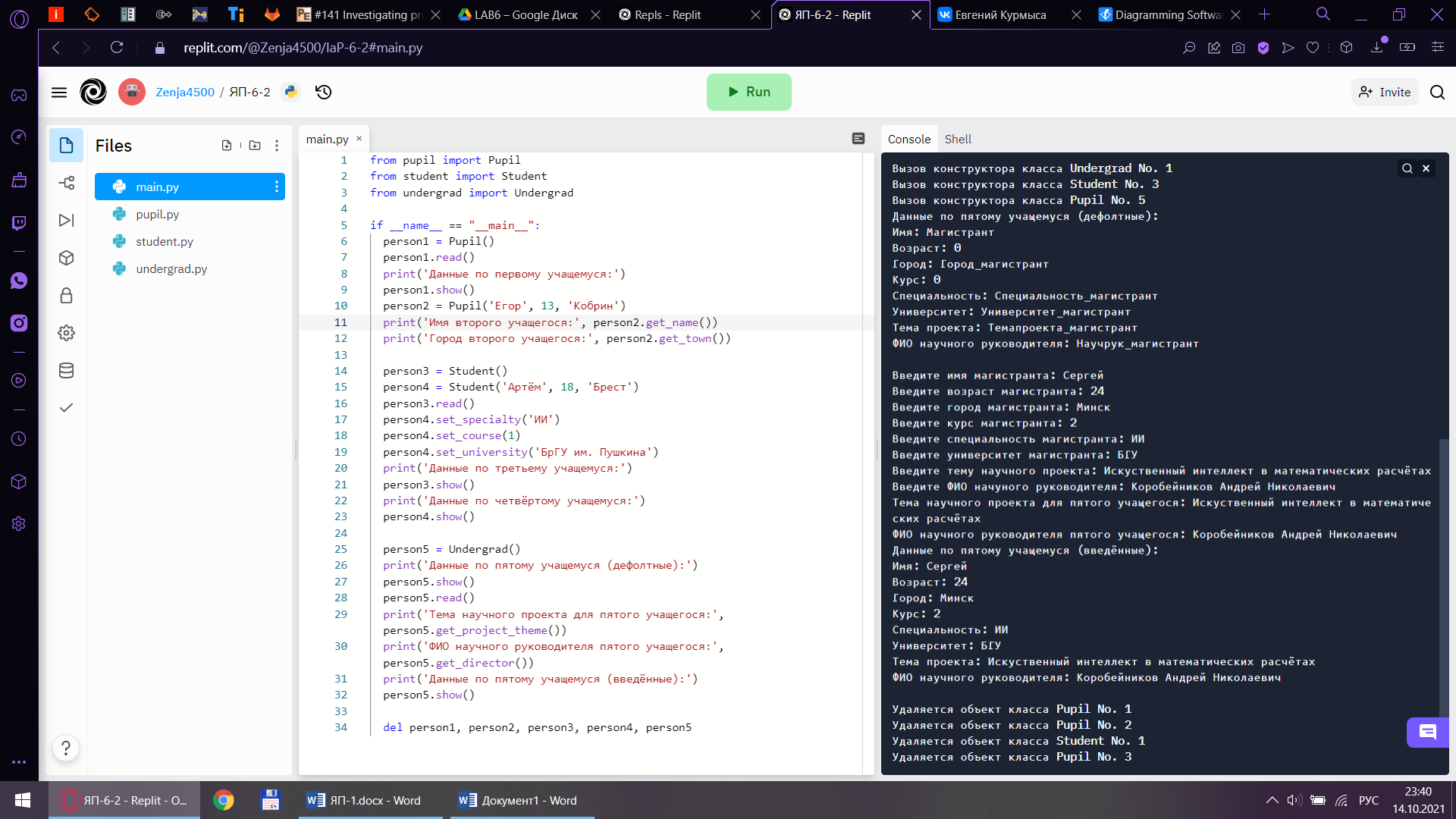
print('Тема научного проекта для пятого учащегося:', person5.get\_project\_theme())

print('ФИО научного руководителя пятого учащегося:', person5.get\_director())

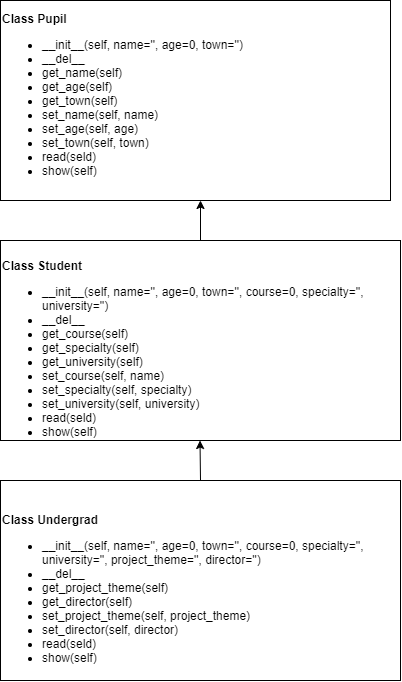
print('Данные по пятому учащемуся (введённые):')

person5.show()

del person1, person2, person3, person4, person5

**Иерархическая схема:**



**Вывод:** яознакомился с принципом инкапсуляции, конструкторами и деструкторами, а также с self в Python; принципом наследования классов в Python.