МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Информатика»
Тема: Парадигмы программирования

Студент гр. 0382 _____ Сергеев Д.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург
2020

Цель работы.

Изучить парадигмы программирования, научиться работать с ООП. Создать систему классов для градостроительной компании.

Задание.

<u>Система классов для градостроительной компании:</u>

Базовый класс -- схема дома *HouseScheme*:

class HouseScheme:

- Поля объекта класса HouseScheme:
    - количество жилых комнат
    - площадь (в квадратных метрах, не может быть отрицательной)
    - совмещенный санузел (значениями могут быть или False, или True)

При создании экземпляра класса HouseScheme необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'

Дом деревенский *CountryHouse*:

class CountryHouse: # Класс должен наследоваться от HouseScheme

- Поля объекта класса CountryHouse:
    - количество жилых комнат
    - жилая площадь (в квадратных метрах)
    - совмещенный санузел (значениями могут быть или False, или True)
    - количество этажей
    - площадь участка

При создании экземпляра класса CountryHouse необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'

Метод __str__()

Преобразование к строке вида:

Country House: Количество жилых комнат <количество жилых комнат>, Жилая площадь <жилая площадь>, Совмещенный санузел <совмещенный санузел>, Количество этажей <количество этажей>, Площадь участка <площадь участка>.

Метод __eq__()

Метод возвращает True, если два объекта класса равны и False иначе. Два объекта типа CountryHouse равны, если равны жилая площадь, площадь участка, при этом количество этажей не отличается больше, чем на 1.

Квартира городская Apartment:

class Apartment: # Класс должен наследоваться от HouseScheme

- Поля объекта класса Apartment:
  - количество жилых комнат
  - площадь (в квадратных метрах)
  - совмещенный санузел (значениями могут быть или False, или True)
  - этаж (может быть число от 1 до 15)
  - куда выходят окна (значением может быть одна из строк: N, S, W, E)

При создании экземпляра класса Apartment необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'

Метод __str__()

Преобразование к строке вида:

Apartment: Количество жилых комнат <количество жилых комнат>, Жилая площадь <жилая площадь>, Совмещенный санузел <совмещенный санузел>, Этаж <этаж>, Окна выходят на <куда выходят окна>.

Переопределите список **list** для работы с домами:

Деревня:

class CountryHouseList: # список деревенских домов -- "деревня", наследуется от класса list

- Конструктор:

1. Вызвать конструктор базового класса

2. Передать в конструктор строку name и присвоить её полю name созданного объекта'"

- Метод append(p_object):

Переопределение метода append() списка.

В случае, если p_object - деревенский дом, элемент добавляется в список, иначе выбрасывается исключение TypeError с текстом: Invalid type <тип_объекта p_object>'"

- Метод total_square():

Посчитать общую жилую площадь

Жилой комплекс:

class ApartmentList: # список городских квартир -- ЖК, наследуется от класса list

- Конструктор:

1. Вызвать конструктор базового класса

2. Передать в конструктор строку name и присвоить её полю name созданного объекта

- Метод extend(iterable):

Переопределение метода extend() списка.

В случае, если элемент iterable - объект класса Apartment, этот элемент добавляется в список, иначе не добавляется.

- Метод floor_view(floors, directions):

В качестве параметров метод получает диапазон возможных этажей в виде списка (например, [1, 5]) и список направлений из ('N', 'S', 'W', 'E').

Метод должен выводить квартиры, этаж которых входит в переданный диапазон (для [1, 5] это 1, 2, 3, 4, 5) и окна которых выходят в одном из переданных направлений. Формат вывода:

<Направление_1>: <этаж_1>

<Направление_2>: <этаж_2>

...

Направления и этажи могут повторятся. Для реализации используйте функцию filter().

Выполнение работы.

Разработанный программный код см. в приложении А.

1) Класс HouseScheme

В метод-конструктор передаются аргументы count, area, toi, далее создаются поля объекта класса self.count=count (количество комнат), self.area=area (жилая площадь), self.toi=toi (наличие совмещенного

санузла), при этом проверяется, соответствуют ли аргументы требованиям(площадь>=0, type(toi)==bool), и в случае несоответствия выбрасывается исключение ValueError с текстом: “Invalid Value”.

2) Класс CountryHouse

   Наследуется от класса HouseScheme.

   В метод-конструктор помимо аргументов класса-родителя поступают аргументы floor_count, sec_area, далее в конструкторе создаются поля объекта класса self.floor_count=floor_count (количество этажей), self.sec_area=sec_area (площадь участка). Также вызывается конструктор класса-родителя. При этом проверяется, соответствуют ли аргументы требованиям, и в случае несоответствия выбрасывается исключение ValueError с текстом: “Invalid Value”. Также переопределен метод __str__, который теперь возвращает строку с информацией об объекте. Кроме этого переопределен метод __eq__, который возвращает True в случае, если жилая площадь и площадь участка равны у двух объектов и разница в количестве этажей не превышает единицы.

3) Класс Apartment

   Наследуется от класса HouseScheme.

   В метод-конструктор помимо аргументов класса-родителя поступают аргументы floor, window, далее в конструкторе создаются поля объекта класса self.floor=floor (номер этажа), self.window=window (направление на которое выходит окно). Также вызывается конструктор класса-родителя. При этом проверяется, соответствуют ли аргументы требованиям, и в случае несоответствия выбрасывается исключение ValueError с текстом: “Invalid Value”. Также переопределен метод __str__, который теперь возвращает строку с информацией об объекте.

4) Класс CountryHouseList

Наследуется от класса list.

В метод-конструктор передаётся аргумент name, далее создаётся поле объекта класса self.name=name (Название деревни). Также вызывается конструктор класса-родителя. Переопределяется метод append, в который поступает объект p_object и если (type(p_object)==CountryHouse), то объект p_object добавляется в конец списка, в противном случае выбрасывается исключение TypeError с текстом 'Invalid type {}'.format(type(p_object)). Также определяется метод total_square, который возвращает общую сумму жилой площади всех домов в деревне.

5) Класс ApartmentList

Наследуется от класса list.

В метод-конструктор передаётся аргумент name, далее создаётся поле объекта класса self.name=name (Название жилого комплекса). Также вызывается конструктор класса-родителя. Переопределяется метод extend, в который поступает объект iterable, далее формируется список, в который попадают объекты, содержащиеся в iterable, которые являются объектами класса Apartment. После этого сформированный список склеивается с объектом класса ApartmentList. Переопределяется метод floor_view, который выводит этаж и направление на которое выходят окна в квартире на данном этаже, если этаж входит в данный диапазон и направление соответствует заданным.

Иерархия классов:

- Класс: HouseScheme

  Наследники: CountryHouse, Apartment
- Класс: list

  Наследники: CountryHouseList, ApartmentList

Переопределенные методы:

- __init__(self,…)
- __str__(self)
- __eq__(self, other)
- append(self, p_object)
- extend(self, iterable)

Метод __str__ будет вызван в случае применение функции str() или print().

Непереопределенные методы класса list для CountryHouseList и ApartmentList будут работать, так как эти классы являются наследниками класса list. Например: self.reverse() – развернёт список.

Выводы.

Были изучены парадигмы программирования, также освоена работа с ООП и создана система классов для градостроительной компании.

Разработана программа, в которой реализуется система классов, все необходимые поля классов инициализированы, все необходимые методы классов переопределены, представлена иерархия классов

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.py

```
class HouseScheme():
    def __init__(self,count,area,toi):
        if type(count)==int and count>=0 and (type(area)== float or
type(area)==int) and (area>=0) and type(toi)==bool:
            self.count=count
            self.area=area
            self.toi=toi
        else:
            raise ValueError('Invalid value')


class CountryHouse(HouseScheme):
    def __init__(self, count, area,toi,floor_count, sec_area):
        super().__init__(count,area,toi)
        if (floor_count>=0 and sec_area>=0):
            self.floor_count=floor_count
            self.sec_area=sec_area
        else:
            raise ValueError('Invalid value')
    def __str__(self):
        return 'Country House: Количество жилых комнат {}, Жилая
площадь {}, Совмещенный санузел {}, Количество этажей {}, Площадь участка
{}.'.format(self.count,self.area,self.toi,self.floor_count,self.sec_area)
    def __eq__(self,other):
        if              ((self.area==other.area)              and
(self.sec_area==other.sec_area)        and        (abs(self.floor_count-
other.floor_count)<=1)):
            return True
        else:
            return False

class Apartment(HouseScheme):
    def __init__(self,count,area,toi,floor,window):
        super().__init__(count,area,toi)
        if (floor>=1 and floor<=15 and (window == 'N' or window=='S'
or window=='E' or window=='W')):
            self.floor=floor
            self.window=window
        else:
            raise ValueError('Invalid value')
    def __str__(self):
        return 'Apartment: Количество жилых комнат {}, Жилая площадь
{},   Совмещенный   санузел   {},   Этаж   {},   Окна   выходят   на
{}.'.format(self.count,self.area,self.toi,self.floor,self.window)

class CountryHouseList(list):
    def __init__(self,name):
        super().__init__()
        self.name=name

    def append(self, p_object):
        a=0
```

```
            if (type(p_object)==CountryHouse):
                super().append(p_object)
            else:
                raise TypeError('Invalid type {}'.format(type(p_object)))

        def total_square(self):
            sum=0
            for i in self:
                sum+=i.area
            return sum

    class ApartmentList(list):
        def __init__(self,name):
            super().__init__()
            self.name=name

        def extend(self,iterable):
            ext=list(filter(lambda x: type(x)==Apartment,iterable))
            super().extend(ext)

        def floor_view(self,floors, directions):
            w_fl=list(filter(lambda x: (x.window in directions) and
(x.floor>=floors[0]) and x.floor<=floors[1],self))
            for i in w_fl:
                print(i.window,i.floor,sep=': ')
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Если результаты тестирования велики (больше 1 страницы), то их выносят в приложение.

Процесс тестирования можно представить в виде таблицы, например:

Таблица Б.2 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.			
2.			
3.			
...			

Обратите внимание, что в нумерации таблицы в приложении обязательно должен быть в качестве префикса номер самого приложения: А.