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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Введение в информационные технологии»

Тема: «Парадигмы программирования»

Студент гр. 9383 _____ Чумак М.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2019

Цель работы.

Изучить ООП в python. Научиться создавать классы, строить их иерархию и уметь создавать исключения в них для особых ситуаций.

Задание.

Система классов для градостроительной компании

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().

'''

В отчете укажите:

1. Иерархию описанных вами классов.

2. Методы, которые вы переопределили (в том числе методы класса object).

3. В каких случаях будет вызван метод __str__().

4. Будут ли работать непереопределенные методы класса list для CountryHouseList и ApartmentList? Объясните почему и приведите примеры.

Выполнение работы.

Описание пунктов из задания:

1. Описание иерархии классов:
   - объект → класс HouseScheme → класс CountryHouse
   - объект → класс HouseScheme → класс Apartment
   - объект → класс list → класс CountryHouseList
   - объект → класс list → класс ApartmentList
2. Методы, переопределённые в данной работе:
   - __init__() – переопределение конструктора;
   - __str__() – переопределение метода вывода строки в нужную для нас форму определённого класса;
   - __eq__() – переопределение метода, сравнивающего в нашем случае экземпляры одного класса, а именно CountryHouse;
   - append() – переопределение метода list, который добавляет только экземпляры класса CountryHouse;
   - extend() – переопределение метода list, который добавляет только экземпляры класса Apartment.
3. Метод __str__() будет вызван в том случае, если пользователь захочет его использовать для присваивания значения к другой переменной, также при передаче его в функцию print(), либо же для других своих целей, введя в этот метод необходимое количество переменных определённого класса.
4. Непереопределённые методы класса list будут работать так же и для классов CountryHouseList и ApartmentList. Это обусловлено тем, что Все классы-наследники имеют такие же методы, как и у классов-родителей, так как из самого названия понятно, что они наследуют эти методы. Мы как бы загружаем в класс-наследник все методы класса-родителя и имеем право использовать и переопределять все необходимые нам методы класса-родителя под наши нужды. Мы можем использовать, например, метод pop() класса CountryHouseList или ApartmentList, чтобы извлечь необходимый нам элемент.

Выводы.

Были изучены принципы ООП в python. Были созданы классы и необходимые для них исключения, построена иерархия классов, а также был решён ряд задач связанный с переопределением методов классов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

Код:

```
class HouseScheme:
    def __init__(self, count_of_rooms, life_area, is_bathroom_included):
            if isinstance(count_of_rooms, int) and (life_area > 0) and isinstance(is_bathroom_included, bool):
                self.count_of_rooms = count_of_rooms
                self.life_area = life_area
                self.is_bathroom_included = is_bathroom_included
            else:
                raise ValueError('Invalid value')


class CountryHouse(HouseScheme):
    def __init__(self, count_of_rooms, life_area, is_bathroom_included, count_of_floors, site):
        super().__init__(count_of_rooms, life_area, is_bathroom_included)
        self.count_of_floors = count_of_floors
        self.site = site

    def __str__(self):
        return "Country House: Количество жилых комнат {}, Жилая площадь {}, Совмещенный санузел {}, Количество " \
               "этажей {}, Площадь участка {}.".format(self.count_of_rooms, self.life_area,
                                                       self.is_bathroom_included, self.count_of_floors, self.site)
```

```
    def __eq__(self, other):
        if self.life_area == other.life_area and self.site == other.site and abs(
                self.count_of_floors - other.count_of_floors) <= 1:
            return True
        else:
            return False


class Apartment(HouseScheme):
    def __init__(self, count_of_rooms, life_area, is_bathroom_included, floor, windows):
        super().__init__(count_of_rooms, life_area, is_bathroom_included)
        self.floor = floor
        self.windows = windows
        if 16 > self.floor > 0 and self.windows in ['N', 'S', 'W', 'E']:
            pass
        else:
            raise ValueError('Invalid value')

    def __str__(self):
        return "Apartment: Количество жилых комнат {}, Жилая площадь {}, Совмещенный санузел {}, Этаж {}, " \
               "Окна выходят на {}.".format(self.count_of_rooms, self.life_area, self.is_bathroom_included,
                                            self.floor, self.windows)


class CountryHouseList(list):
```

```
    def __init__(self, name):
        super().__init__()
        self.name = name

    def append(self, p_object):
        if isinstance(p_object, CountryHouse):
            self = super().append(p_object)
        else:
            raise TypeError("Invalid type {}".format(p_object.__class__))

    def total_square(self):
        sum_of_square = 0
        for i in self:
            sum_of_square += i.life_area
        return sum_of_square


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

    def extend(self, iterable):
        for x in iterable:
            if x.__class__ == Apartment:
                self.append(x)

    def floor_view(self, floors, directions):
```

```
            for x in filter(lambda i: floors[0] <= i.floor <= floors[1] and i.windows
in directions, self):
                print('{0}: {1}'.format(x.windows, x.floor))
```