4.6 Множественное наследование 8 out of 10 steps passed 13 out of 20 points received

Подвиг 5. В программе объявлены два класса:

```
class ShopItem:
    ID_SHOP_ITEM = 0

    def __init__(self):
        super().__init__()
        ShopItem.ID_SHOP_ITEM += 1
        self._id = ShopItem.ID_SHOP_ITEM

    def get_pk(self):
        return self._id


class Book(ShopItem):
    def __init__(self, title, author, year):
        super().__init__()
        self._title = title
        self._author = author
        self._year = year
```

Затем, создается объект класса Book (книга) и отображается в консоль:

```
book = Book("Python ООП", "Балакирев", 2022)
print(book)
```

В результате, на экране увидим что то вроде:

<__main__.Book object at 0x0000015FBA4B3D00>

Но нам требуется, чтобы здесь отображались локальные атрибуты объекта с их значениями в формате:

<атрибут_1>: <значение_1>
<атрибут_2>: <значение_2>
...
<атрибут_N>: <значение_N>

Для этого вам дают задание разработать два класса:

ShopGenericView - для отображения всех локальных атрибутов объектов любых дочерних классов (не только Book);
ShopUserView - для отображения всех локальных атрибутов, кроме атрибута _id, объектов любых дочерних классов (не только Book).

То есть, в этих классах нужно переопределить два магических метода: __str__() и __repr__().

Пример использования классов (эти строчки в программе писать не нужно):

```
class Book(ShopItem, ShopGenericView): ...
book = Book("Python ООП", "Балакирев", 2022)
print(book)
# на экране увидим строчки:
# _id: 1
# _title: Python ООП
# _author: Балакирев
# _year: 2022
```

Другой вариант использования классов:

```
class Book(ShopItem, ShopUserView): ...
book = Book("Python ООП", "Балакирев", 2022)
print(book)
# на экране увидим строчки:
# _title: Python ООП
# _author: Балакирев
# _year: 2022
```

P.S. В программе требуется объявить только классы. На экран выводить ничего не нужно.