4.1 Наследование в объектно-ориентированном программировании | 7 out of 11 steps passed | 14 out of 29 points received

Видео-разбор подвига (решение смотреть только после своей попытки):
https://youtu.be/k0xvHbnTnoo (https://youtu.be/k0xvHbnTnoo)

Подвиг 6. Еще один пример, когда в базовом классе прописывается необходимый начальный функционал для дочерних классов.

Известно, что браузер (и не только) может отправлять на сервер различные типы запросов: GET, POST, PUT, DELETE и др. Каждый из этих типов запросов обрабатывается в программе на сервере своим отдельным методом. Чтобы каждый раз не прописывать все необходимые методы в классах при обработке входящих запросов, они выносятся в базовый класс и вызываются из дочерних. Выполним такой пример.

Пусть в программе объявлен следующий базовый класс с именем GenericView:

```
class GenericView:
    def __init__(self, methods=('GET',)):
        self.methods = methods

    def get(self, request):
        return ""

    def post(self, request):
        pass

    def put(self, request):
        pass

    def delete(self, request):
        pass
```

Здесь каждый метод отвечает за обработку своего типа запроса. Параметр methods - это кортеж или список, состоящий из набора разрешенных запросов: строк с именами соответствующих методов (как правило, пишут заглавными буквами).
Вам необходимо объявить дочерний класс с именем DetailView, объекты которого можно создавать командами:

```
dv = DetailView()  # по умолчанию methods=('GET',)
dv = DetailView(methods=('PUT', 'POST'))
```

Для инициализации атрибута methods следует вызывать инициализатор базового класса GenericView.

Далее, в классе DetailView нужно определить метод:

```
def render_request(self, request, method): ...
```

который бы имитировал выполнение поступившего на сервер запроса. Здесь request - словарь с набором данных запроса; method - тип запроса (строка: 'get' или 'post' и т.д.).

Например:

```
html = dv.render_request({'url': 'https://site.ru/home'}, 'GET')
```

должен быть обработан запрос как GET-запрос с параметром url и значением 'https://site.ru/home'. Параметр url является обязательным в словаре request для каждого запроса.

В методе render_request() необходимо выполнить проверку: является ли указанный метод (method) разрешенным (присутствует в коллекции methods). Если это не так, то генерировать исключение командой:

```
raise TypeError('данный запрос не может быть выполнен')
```

Если проверка проходит, то выполнить соответствующий метод (или get(), или post(), или put() и т.д. с возвращением результата их работы).

Подсказка: для получения ссылки на нужный метод можно воспользоваться магическим методом __getattribute__() или аналогичной функцией getattr()).

Наконец, в дочернем классе DetailView следует переопределить метод get() для нужной нам обработки GET-запросов. В этом методе нужно выполнить проверку, что параметр request является словарем. Если это не так, то генерировать исключение:

```
raise TypeError('request не является словарем')
```

Сделать проверку, что в словаре request присутствует ключ url. Если его нет, то генерировать исключение:

```
raise TypeError('request не содержит обязательного ключа url')
```

Если же все проверки проходят, то вернуть строку в формате:

"url: <request['url']>"

Пример (эти строчки в программе писать не нужно):

```
dv = DetailView()
html = dv.render_request({'url': 'https://site.ru/home'}, 'GET')   # url:
https://site.ru/home
```

P.S. В программе нужно объявить только классы. Выводить на экран ничего не нужно.

To solve this problem please visit

https://stepik.org/lesson/701995/step/7