Задания про классы и наследование

Как правило задачи про классы носят не вычислительный характер. Обычно нужно написать классы, которые отвечают определенным интерфейсам. Насколько удобны эти интерфейсы и как сильно связаны классы между собой, определит легкость их использования в будущих программах.

Предположим есть данные о разных автомобилях и спецтехнике. Данные представлены в виде таблицы с характеристиками. Обратите внимание на то, что некоторые колонки присущи только легковым автомобилям, например, кол-во пассажирских мест. В свою очередь только у грузовых автомобилей есть длина, ширина и высота кузова.

Тип (`car_type`)	Марка (`brand`)	Кол-во пассажирских мест (`passenger_seats_count`)	Фото (`photo_file_name`)	Кузов ДхШхВ, м (`body_whl`)	Грузоподъемность,Тонн (`carrying`)	Дополнительно (`extra`)
car	Nissan xTtrail	4	f1.jpeg	-	2.5	-
truck	Man	-	f2.jpeg	8x3x2.5	20	-
car	Mazda 6	4	f3.jpeg	-	2.5	-
spec_machine	Hitachi	-	f4.jpeg	-	1.2	Легкая техника для уборки снега

Вам необходимо создать свою иерархию классов для данных, которые описаны в таблице.

- `BaseCar`
- `Car(BaseCar)`
- `Truck(BaseCar)`
- `SpecMachine(BaseCar)`

У любого объекта есть обязательный атрибут `car_type`. Он означает тип объекта и может принимать одно из значений: `car`, `truck`, `spec_machine`.

Также у любого объекта из иерархии есть фото в виде имени файла — обязательный атрибут `photo_file_name`.

В базовом классе нужно реализовать метод `get_photo_file_ext` для получения расширения файла (".png", ".jpeg" и т.д.) с фото. Расширение файла можно получить при помощи `os.path.splitext`.

Для грузового автомобиля необходимо разделить характеристики кузова на отдельные составляющие `body_length`, `body_width`, `body_height`. Разделитель — латинская буква `x`. Характеристики кузова могут быть заданы в виде пустой строки, в таком случае все составляющие равны `0`. Обратите внимание на то, что характеристики кузова должны быть вещественными числами.

Также для класса грузового автомобиля необходимо реализовать метод `get_body_volume`, возвращающий объем кузова в метрах кубических.

Все обязательные атрибуты для объектов `Car`, `Truck` и `SpecMachine` перечислены в таблице ниже, где `1` - означает, что атрибут обязателен для объекта, `0` - атрибут должен отсутствовать.

-	Car	Truck	SpecMachine
`car_type`	1	1	1
`photo_file_name`	1	1	1
`brand`	1	1	1
`carrying`	1	1	1
`passenger_seats_count`	1	0	0
`body_width`	0	1	0
`body_height`	0	1	0
`body_length`	0	1	0
`extra`	0	0	1

Далее необходимо реализовать функцию, на вход которой подается имя файла в формате **csv**. Файл содержит данные аналогичны строкам из таблицы. Вам необходимо прочитать этот файл построчно при помощи модуля стандартной библиотеки **csv**. Затем проанализировать строки и создать список нужных объектов с автомобилями и специальной техникой. Функция должна возвращать список объектов.

Не важно как вы назовете свои классы, главное чтобы их атрибуты имели имена:

```
car_type
brand
passenger_seats_count
photo_file_name
body_width
body_height
body_length
carrying
extra
```

И методы:

`get_photo_file_ext` и `get_body_volume`

У каждого объекта из иерархии должен быть свой набор атрибутов и методов. У класса легковой автомобиль не должно быть метода `get_body_volume` в отличие от класса грузового автомобиля.

Функция, которая парсит строки входного массива, должна называться `get_car_list`. Для проверки работы своей реализации функции `get_car_list` и всех созданных классов вам необходимо использовать следующий csv-файл: `coursera_week3_cars.csv`.

Первая строка в исходном файле — это заголовок **csv**, который содержит имена колонок. Нужно пропустить первую строку из исходного файла. Обратите внимание на то, что исходный файл с данными содержит некорректные строки, которые нужно пропустить. Если возникают исключения в процессе создания объектов из строк **csv**-файла, то требуется их корректно обработать стандартным способом. Проверьте работу вашего кода с входным файлом.

Пример чтения **csv** файла:

```
import csv

with open(csv_filename) as csv_fd:
    reader = csv.reader(csv_fd, delimiter=';')
    next(reader)  # пропускаем заголовок
    for row in reader:
        print(row)
```

Также обратите внимание, что все значения в **csv** файле при чтении будут python-строками. Нужно преобразовать строку в `int` для `passenger_seats_count`, во `float` для carrying, а также во `float` для `body_width` `body_height`, `body_length`.

Также ваша программа должна быть готова к тому, что в некоторых строках данные могут быть заполнены некорректно. Например, число колонок меньше . В таком случае нужно проигнорировать подобные строки и не создавать объекты. Строки с пустым значением для `body_whl` игнорироваться не должны. Вы можете использовать механизм исключений для обработки ошибок.

Ниже приведен пример с заготовкой кода для выполнения задания.

```
class CarBase:
    def __init__(self, brand, photo_file_name, carrying):
        pass


class Car(CarBase):
    def __init__(self, brand, photo_file_name, carrying, passenger_seats_count):
        pass


class Truck(CarBase):
    def __init__(self, brand, photo_file_name, carrying, body_whl):
        pass


class SpecMachine(CarBase):
    def __init__(self, brand, photo_file_name, carrying, extra):
        pass


def get_car_list(csv_filename):
    car_list = []
    return car_list
```

Вам необходимо расширить функционал исходных классов, дополнить методы нужным кодом и реализовать функцию `get_car_list`.

Успехов!