Управление парком автомобилей

Описание задачи:
Создайте программу для управления парком автомобилей. В парке есть различные автомобили, каждый из которых имеет уникальные характеристики.

Требования:

1. Создайте базовый класс для автомобилей и несколько подклассов для различных типов автомобилей (например, легковые и грузовые автомобили).
2. Реализуйте систему добавления автомобилей в парк и отображения информации о каждом автомобиле.
3. Реализуйте метод для аренды автомобилей и отслеживания их наличия.

Решение:

```
class Car:
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year
        self.is_rented = False

    def rent(self):
        if not self.is_rented:
            self.is_rented = True
            print(f"{self.brand} {self.model} has been rented.")
        else:
            print(f"{self.brand} {self.model} is already rented.")

    def return_car(self):
        if self.is_rented:
            self.is_rented = False
            print(f"{self.brand} {self.model} has been returned.")
        else:
            print(f"{self.brand} {self.model} was not rented.")

class Sedan(Car):
    def __init__(self, brand, model, year, trunk_size):
        super().__init__(brand, model, year)
        self.trunk_size = trunk_size

class Truck(Car):
    def __init__(self, brand, model, year, cargo_capacity):
        super().__init__(brand, model, year)
        self.cargo_capacity = cargo_capacity

class CarRental:
    def __init__(self):
        self.cars = []

    def add_car(self, car):
        self.cars.append(car)

    def show_cars(self):
        for car in self.cars:
            status = "rented" if car.is_rented else "available"
            print(f"{car.brand} {car.model} ({car.year}) - {status}")

    def rent_car(self, brand, model):
        car = next((c for c in self.cars if c.brand == brand and c.model == model and not c.is_rented),
None)
        if car:
```

```
            car.rent()
        else:
            print(f"{brand} {model} is not available for rent.")

    def return_car(self, brand, model):
        car = next((c for c in self.cars if c.brand == brand and c.model == model and c.is_rented),
None)
        if car:
            car.return_car()
        else:
            print(f"{brand} {model} was not rented.")

# Пример использования
rental = CarRental()

sedan = Sedan("Toyota", "Camry", 2021, "15.1 cu ft")
truck = Truck("Ford", "F-150", 2022, "3270 lbs")

rental.add_car(sedan)
rental.add_car(truck)

rental.show_cars()
rental.rent_car("Toyota", "Camry")
rental.show_cars()
rental.return_car("Toyota", "Camry")
rental.show_cars()
```

Описание решения:

1. **Класс `Car`**: Базовый класс для всех автомобилей, содержит общие свойства и методы.
2. **Классы `Sedan` и `Truck`**: Наследуют от `Car` и добавляют специфичные свойства для каждого типа автомобилей.
3. **Класс `CarRental`**: Управляет списком автомобилей и реализует методы для добавления, отображения, аренды и возврата автомобилей.