CS109 2025Spring Assignment6 ParkingLot

Designer: ZHU Yueming

Junit: RUI Ziming   Tester: FENG Haibo

Keyword Annotation:
Field(s): You can see the field(s) you need to define after it.
Method(s): You can see the method(s) you need to define or modify after it.

What to Submit

Time.java

Vehicle.java

Car.java

Bus.java

ParkingLot.java

ConcreteParkingLot.java

Introduction

In the Parking lot, we can add parking space for **Car**, **Bus** introduced in assignment 5. When vehicles enters the parking lot, the size of specific kind of vehicle will be deducted. When vehicles leave out of the parking lot, parking fees should be paid.

It is only a simple Parking System for exercising the interface, abstract class, inheritance, polymorphism, wildcard, etc. The implement of Parking System in real world would be more complex than our assignment.

Other Important parameters and Test cases

1. We can ensure that the same vehicle may enter and exit the parking lot multiple times with no overlapping.
2. The initial **Time** is **00:00**, and we can make sure that the time cannot be exceeded 24 hours in this assignment.
3. We can ensure that the **arrivalTime** and **plateNumber** of each vehicle will not be repeated.
4. We ensure that the test case does not allow a vehicle that **has not left** the parking lot to enter the parking lot again.
5. When we mentions the **current Time** in this assignment, which means the current time of parking in our assignment, but not the actual time in the real world.
6. We can make sure that the test cases of plate numbers for each arriving vehicle are all valid.

Classes

1. Interface ParkingLot

The ParkingLot interface describes a common method for parking lots. Each abstract method will be described in detail in the implementation of the interface.

The interface we provided as follows:

```java
import java.util.List;

public interface ParkingLot {
    public Time getTime();

    public void minutesPassed(int minutes);

    public String parkingLotStatus();

    public void driveInto(int type, String... plateNumbers);

    public void driveOut(String ... plateNumbers);

    public Vehicle getVehicleByPlateNumber(String plateNumber);

    public List<Vehicle> getVehiclesByNumber();

    public List<String> getParkingRecordByArriveTime(Time start, Time end);

    public int currentIncome();
}
```

2. class ConcreteParkingLot

> - Do not modify or remove any methods or fields that have been already defined.
> - You can add other methods or attributes that you think are necessary.

Fields:

- `private List<Vehicle> vehicles;`

  Contain all vehicles, including all different types such as Car and Bus.

  This attrivte should add all vehicles that have been **entered into** the parking lot, but if the vehicle enter the parking lot **for more than one time**, you do not add the vehicle again.

- `private Time currentTime`

  The time of out assignment, we use 24-hour clock. The initial value of currentTime is `00:00`

- `private final int CAR_CAPACITY`

The capacity of car.

- `private final int BUS_CAPACITY`

  The capacity of bus.

Methods:

- **constructor**

  ```
  public ConcreteParkingLot(int carCapacity, int busCapacity)
  ```

  You need to initialize two parameters in the constructor.
  You also need to initialize the vehicles attribute and the currentTime attribute. The initial value of currentTime is `00:00`.

- **getTime()**

  ```
  public Time getTime();
  ```

  The method is used to return the attribute `currentTime`.

- **minutesPassed(int minutes)**

  ```
  public void minutesPassed(int minutes);
  ```

  The method is used to record the passing minutes. You need to add the passing minutes to `currentTime` and then update it.

  parameter: `minutes` - an integer which means how much minutes is passed. It will be like "55", "300", "600" etc.

- **parkingLotStatus()**

  ```
  public String parkingLotStatus();
  ```

  The method is used to return a format `String` include the count and capacity of two type of vehicles about parking spot.

  The return format like:

  ```
   [car]:[car count]/[car capacity] [bus]:[bus count]/[bus capacity]
  ```

  For example:

  ```
  car:5/8 bus:2/4
  car:10/12 bus:0/3
  ```

- **currentIncome()**

```java
public int currentIncome()
```

Return the total parking fee for current time.

- **driveInto()**

```java
 public void driveInto(int type, String... plateNumbers)
```

The method is used to record the information of a series of new vehicles, which prepare to park.

You need to create each corresponding `Vehicle`:

1. If the count of current vehicle equivalent to the capacity of the type of current vehicle, the vehicles cannot be driven into the parking pots.

   For example, if the capacity of "Car" is 5, and the current count of "Car" in parking lot is 4, now there are 3 "Car" (A22222,A33333,A44444) want to drive into the parking lot, in this case, only A22222 can be driven but other two cannot.

2. Please note that in test cases the newly entered vehicles may include ones that have entered but have exited the parkinglot, which means the vehicle may enter the parking lot for more than one times.

3. If the vehicle can drive into the parking pot succefully, then

   - Set its `arriveTime` and `plateNumber`. The hour and minute of ArriveTime is the same as `currentTime` of the current parking lot. If multiple vehicles enter the parking lot at the same time, the entry time of each vehicle increases by 1 minute in the order of the index of the parameter `plateNumbers[]`.

     For example: If currentTime is 10:10, and plateNumbers[] are "A11111", "B11111", "C11111", the arriveTime for each vehicle are 10:10, 10:11 and 10:12 respectively.

   - The hour and minute of attribute `currentTime` must be consistent with the hour and minute of arriveTime of the last vehicle entering the parking lot.

   - Add them to vehicles list and add count number to corresponding parking Spot. If the vehicle enter the parking lot again, do not add it again.

Parameters:

1. `type` - an integer which represents the type of vehicle. It only has two values: 0 and 1.
   -- `type==0` represent the Car
   -- `type==1` represent the Bus

2. `plateNumber` - an array with a series of new cars with their plates, it will be like ("A30001", "A10088") or ("A30001", "B10088", "C11111") . We can make sure that our test cases are all valid.

   Example 1

```java
ParkingLot parkingLot = new ConcreteParkingLot(4, 2);
parkingLot.minutesPassed(35);
parkingLot.driveInto(0,"A11111","A11112","C11113","C11114","D11115");
System.out.println(parkingLot.getTime());
```

The attribute `vehicles`:

Type	PlateNumber	ArriveTime	IsInside
Car	A11111	00:35	true
Car	A11112	00:36	true
Car	C11113	00:37	true
Car	C11114	00:38	true

The result is 00:38

- **driveOut()**

```java
public void driveOut(String... plateNumbers)
```

This method is used to record the information of the vehicle leaving the parking lot. When the vehicle leaves the parking lot, the `inside` property of the vehicle is changed to **false** and the parking fee is calculated.

1. Minus corresponding count number in corresponding parking spot
2. Calculate the money that the vehicle need to pay. Then add all to your `totalIncome` and update it
   We can make sure that our test cases of plateNumbers are all vehicles in the parking lots.
3. The hour and minute of leave time of each car is the same as `currentTime`.

Parameters:

`plateNumbers` - an array with a series of vehicles with their plates, it will be like ("A30001", "A10088"), ("B30001", "D10088", "E11111"). We can make sure that our test cases are all valid.

Example 2

Continue to Example1 add following code:

```java
parkingLot.minutesPassed(29);
parkingLot.driveOut("A11111","A11112");
```

The attribute `vehicles` and parking fee:

Type	PlateNumber	ArriveTime	IsInside	Parking Fee
Car	A11111	Null	False	15
Car	A11112	Null	False	15
Car	C11113	00:37	True	Haven't submit
Car	C11114	00:38	True	Haven't submit

Example 3

Continue to Example2 add following code:

```
parkingLot.minutesPassed(60);
parkingLot.driveInto(0,"B11111","A11112","B11112","B11113");
```

The attribute `vehicles` :

Type	PlateNumber	ArriveTime	IsInside
Car	A11111	Null	False
Car	A11112	02:08	true
Car	C11113	00:37	true
Car	C11114	00:38	true
Car	B11111	02:07	true

- **getVehicleByPlateNumber()**

```
public Vehicle getVehicleByPlateNumber(String plateNumber)
```

The method is used to find and return a `Vehicle` by the parameter `plateNumber`.
You need to find the vehicle by its `plateNumber` and then return the `Vehicle`.

If no vehicle match the parameter plateNumber, return null.

- **getVehiclesByNumber()**

```
public List<Vehicle> getVehiclesByNumber()
```

You need to sort the attribute `vehicles` by `plateNumber` in ascending order, and then the method is used to return the `List` of the attribute `vehicles`.

- **getParkingRecordByArriveTime()**

```
public List<String> getParkingRecordByArriveTime(Time start, Time end)
```

This method needs to return all vehicle entry records between parameter start time and end time (included those two time), sorted in ascending order of arriveTime of vehicle.

The return type is a `List<String>` , for each string in List represents the arrive record, and with a format as:

```
[Vehicle name] [plateNumber] [arrive Time] [leave Time]
```

- If the vehicle hasn't left the parking lot , the leave time is null.
- If the vehicle arrived into the parking lot for many times, you need record it again.

Example 4

Continue to Example3 add following code:

```
parkingLot.getParkingRecordByArriveTime(new Time(0,0),new Time(3,20))
                .forEach(System.out::println);
```

Result:

```
Car A11111 00:35 01:07
Car A11112 00:36 01:07
Car C11113 00:37 null
Car C11114 00:38 null
Car B11111 02:07 null
Car A11112 02:08 null
```

The car `A11112` arrive into the parking lot for two times, then the result should have two records for A11112.

> Hint: In this method, you can create another class like ParkingRecord to store data of the arrive record.