This Java application is built to model the parking system used in Christchurch International Airport and some features in the Christchurch International Airport website. It is interacting with the user through a GUI called NosyParker. The user gets to see the available parking lots the airport has to offer. If the user wishes to park in any lot, they have to enter the registration number representing the Vehicle object and click ‘admit’. To leave, the user must click ‘release’. The application also lets the user find out how much it costs to park in a chosen parking lot for a given period.

A normal array might work since the size of the parking lot is fixed (eg. initialCapacity). Adding the vehicle is efficient but removing the vehicle would require a lot of transversal through the whole list. This makes the code complicated in comparison to using a Collection which is equipped with adding and removing elements. Having an array of fixed size will also result in wasted memory, as not all parking spaces will be occupied. This is inefficient compared to a dynamic data structure like ArrayList in Collection.
Most of the methods in the parking lot classes rely on methods in the interface Parkable. An extra array list was added to each of the four parking lot classes (e.g. ‘occupiedShortStayParkingLot’ of type ArrayList). This makes it easy to keep track of Vehicle objects in each parking lot. I choose ArrayList because it is simple and works well at the stage when there is no constraint such as the need to be fast. The problem with using ArrayList is that when the number of vehicle increases, it takes longer to traverse through the list to check whether the vehicle has been parked or not (using the added parked() method in the ParkingSystem class). However, that would depend on the scale to which the software is used. These requirements will be discussed with the client.

I also consider using HashSet because it offers constant time performance for simple operation like add, remove, contains, and size. But I have to observe the airport parking system in more details and talk to the client and find out what they want(eg. fast process or less memory). This is important as with HashSet, I have to concern about load factor of hash table as this will have a great effect on the iteration performance.

In class ‘Vehicle’, I try to emulate real life situations. This is done by having the constructor check whether the registration number is a combination of numbers and/or letters or not. If it is not, the exception called IllegalArgumentException is thrown. This checking can also be done in the GUI instead of the constructor, however this method means the class Vehicle can be used in other tasks and not specifically for airport parking alone.

There is a special period which needs special handling in computeCharge() method. Example 3:00AM 03/06/15 – 2:00AM 04/06/15 the obtained value from getHourOfDay() would be 1 and getMinuteOfDay() would be -60 although the actual period is 23 hours. This special case is observed when the difference between getMinuteOfDay() from and getMinuteOfDay() to is negative. It is then handled by adjusting the value before doing any checking for charge. This is done by finding out how far away the arrival time to 1440 minutes(24 hours), then adding the result to the departure time and finally subtracting one day to compensate the added minutes.

Prevent adding vehicle in two different places
The property (keepTrackVehicles) and methods(parked, checkIn, and checkOut) were added in class ParkingSystem to assist the GUI code ‘SimpleStatusPanel’ and ‘SimpleStatusTranscriptPanel’. This is to prevent the same vehicle parking in two different places. In ‘SimpleStatusPanel’, before the vehicle is admitted, the parked() method is used to check whether the vehicle is in the parking system or not. If it is, the program will let the user know, otherwise the vehicle is admitted. Before the vehicle can be released, the system will check using occupants() method whether the vehicle belongs to that parking lot or not. If it is, then the vehicle can be released.

In Junit test for all computeCharge methods, I use getAmountMajorInt() to convert Money to integer. I assume this is okay for this project as most of the money is integer however if they decide to change the charge to involve cent then I may have to use other method.

Extensions

Adding new features

In ‘SimpleStatusPanel’, I changed the code so that before the vehicle is admitted, the user has to input both their name and address. This information will be stored in the class Vehicle to use in method charge(), along with the arrival time which is recorded when the vehicle is admitted. I ensured that the incomplete information is discarded by checking for null. This will prevent Java throwing the exception called NullPointerException when the user clicks ‘cancel’ or closes the panel. I also make sure we have successfully retrieved their information by repeating input if the user has not put down their name or address. When the vehicle leaves, the departure time is recorded and the system will send the charge to the address if the owner has not paid by another method like cash. I ensured that only the vehicle with that registration number is released and no other vehicles, by checking if the parking lot contains that vehicle.

Adding new panel ‘AdvicerPanel’

I added another feature called ‘AdvicerPanel’ to NosyParker. This ‘AdvicerPanel’ will call ‘ChargeAdvicer’ which will find out what is the cheapest option during the given time period and how much the user will save in absolute value and percentage. The ‘AdvicerPanel’ requires the user to select the period and click the button ‘Advice’ to see the solution.

Idea for further extensions

In the future, I would try to restrict the name and address so that the system only accepts sensible name and address. E.g. names can’t be all number and can accept only an address which appears in Google Maps.
I was also thinking of having a radio button to ask what payment methods the owner wants to pay for their parking when the vehicle is released. If the user chooses ‘cash’, then the user will be navigated to another panel which will use something similar to CashRegister.java in Tutorial Lab5 to work with cash. I also want to add a graphic of cash changing depending on how much the user puts their money in.