Programming Assignment 4

Getting Started

This assignment emphasizes conditional statements, described in Chapter 3 of the textbook, but you will also use a lot of tools and techniques learnt earlier in the course. Review in-class examples and Handout 4.

Programming Project

ParkingRates: Calculate the cost of parking. **worth 18 points**

An owner of a parking garage in New York has contracted you to write a program that will compute the amount due for parking. The parking rates are as follows:

1. *Regular*: Unless the *early bird* rule (see below) is applicable, the cost of parking equals
   - an entry fee of $2.00, plus
   - $3.00 per each full half hour before 11:00 a.m., plus
   - $5.00 per each full half hour after 11:00 a.m.

   The *early bird* rate applies if the car has entered the garage between 6:00 a.m. and 8:30 a.m. and left between 3:00 p.m. and 6:00 p.m. The early bird parking is charged a flat fee of $15.00.
2. *Validated*: The first three hours cost $5.00. After that, every full half-hour is billed an additional $3.00.

You can assume that each vehicle entered and exited the garage on the same day. More specifically, the program should:

1. Ask the user to enter a character representing the parking rate: V - for validated and R - for regular.
2. Ask the user to enter the entry and exit time in the garage in a format shown in the interaction below.
3. Display the amount due for parking.

The following sample interactions illustrate the execution of the program.

```
Please enter type of rate:  V or R: R
Please enter time in and time out in hh:mm form, separated by spaces:  8:30 16:07
Charging the EARLY BIRD rate.  Please pay $15.00.
```

In the above example, based on the entry and exit time, the vehicle qualified for the early bird rate of $15.00.

Next interaction demonstrates one case of validated parking:

```
Please enter type of rate:  V or R: V
Please enter time in and time out in hh:mm form, separated by spaces:  12:35 17:52
Charging VALIDATED rates.  Please pay $17.00.
```

In the above case, the vehicle was parked for 317 minutes, which is 137 minutes over three hours (since $317 - 3 * 60 = 137$). Thus, the charges include \$5.00 for the first 3 hours plus \$3.00 per each additional half hour, i.e. $\$5.00 + (137/30) * \$3.00 = \$5.00 + 4 * \$3.00 = \$17.00$. Note that / refers to the operator of *integer* division.

The third interaction shows one case of regular, non early bird parking:

```
Please enter type of rate:  V or R: R
Please enter time in and time out in hh:mm form, separated by spaces:  4:15 7:22
Charging REGULAR rates.  Please pay $20.00.
```

In this interaction, the vehicle entered and exited before the 11:00 rate change time, staying exactly 187 minutes. Thus, the amount charged equals the entry fee of \$2.00 plus \$3.0 per each full half hour (30 minutes) in the garage, i.e. $\$2.00 + (187/30) * \$3.00 = \$2.00 + 6 * \$3.00 = \$20.00$.

Notes and Hints: Note that the above examples do not represent all possible scenarios involved in computing the amount due. You should work out the possible scenarios and gradually write the Java code for them. **It is essential that you do all comparisons and computations that involve time using the all-minute representation, otherwise you may run into great obstacles.**

I suggest you proceed as follows:

1. At first, work out the algorithm and Java code that will convert the time values entered by the user into all-minute representation using an integer. You will need to use method Integer.parseInt() shown in one of the practice problems in class, to convert a string into an integer. Test your code to verify its correctness.
2. Develop the algorithm and code for the regular early bird parking. Test it.
3. Develop the algorithm and code for the validated parking. Test it.
4. Develop the algorithm and code for the regular non-early-bird parking. Test it.
5. Put on finishing touches and test the whole code.

There are a lot of data parameters, such as rates, fees and cut-off times, which you should represent in your code using named constants. For example, to define the time when the regular rate changes, I suggest the following declaration, which makes it easy to understand that it occurs at 11:00:

```
final int RATE_CHANGE_TIME = 11*60; // time when regular rate switches
```

Grading. Partial credit will be awarded, as always, but your program must compile without errors and run. The grading schema is as follows:

- 2 points for correctly reading the input and displaying the amount due with exactly 2 digits after the decimal period.
- 3 points for correctly processing the early bird parking,
- 5 points for correctly processing the validated parking,
- 6 points for correctly processing the regular non-early-bird parking,
- 2 points for good programming style. The **style requirements** have been introduced in class and in the earlier homework assignments.

Good luck and Happy Programming!

(produced on September 24, 2013 by Michael Banks (original by Tamara Babaian))