CSE 174 – Sections D, E – Spring 2013
PROGRAM #9: 20 points – Due Thursday, April 4

Write a program to model the movement and actions of two trucks. The program will contain two classes, a Truck class and a TruckMover class. The Truck class has already been written and documented.

The main method of the TruckMover class will create two Truck objects (one with capacity 20 tons and one with capacity 10 tons) and process a file containing commands for the movement, loading, and unloading of the trucks. The user should be prompted for the name of the command file. The organization of the file follows. There will be a series of commands using the following keywords:

load x	– load x tons of material into the active truck
move x y	– move the active truck to position x, y
dump	– dump all of the material from the active truck
home	– move the active truck to position 0,0
switch	– change the active truck
quit	– sentinel command to end the command file

As you process the command file, you will display the command given and the result of processing that command (see sample output below). Truck 1 (the 20 ton truck) will initially be the active truck. After processing the command file, there is a brief summary report expected (see sample output below). PLEASE USE THE METHODS OF THE TRUCK CLASS TO PERFORM ALL OF THE COMMANDS. Your program should not need to compute anything.

The Truck class (Truck.java), the Truck documentation (Truck.html), and a file of commands (commands.txt) is provided in S:\EAS\handouts\byrketdl\cse174\Program9.

Incremental scoring option:

I am providing an incremental scoring option for this program similar to Program #8. One of the more difficult aspects of this program is switching back and forth between trucks. If you can implement this program for a single truck, it will be worth 16 of the 20 points. In this case, you will simply create a single truck of capacity 20 and process a command file without any switch commands. I have provided a sample command file called noswitch.txt that you can use to test your program. Again, a good programming strategy might be to implement this version first and then add the two truck feature. A sample run with the "noswitch.txt" is also in the handouts folder (sampleNoSwitch.txt).

Sample run using commands.txt with notations:

```
Enter file name: commands.txt
Command: load 15 ==> 15 tons were loaded                 <- Loads 15 tons on truck 1
Truck 1 state: (0, 0), 15 tons                           <- Current state of truck 1 using toString()
Command: move 12 -4 ==>  truck moved to (12, -4)         <- Move truck 1
Truck 1 state: (12, -4), 15 tons
Command: load 10 ==> 5 tons were loaded                  <- Tried to load 10 tons, but only 5 loaded
Truck 1 state: (12, -4), 20 tons
Command: home ==>  truck moved to (0, 0)                 <- Move truck 1 to (0, 0)
Truck 1 state: (0, 0), 20 tons
Command: dump ==> 20 tons were dumped                    <- Dump truck 1
Truck 1 state: (0, 0), 0 tons
Command: move -1 -1 ==>  truck moved to (-1, -1)
Truck 1 state: (-1, -1), 0 tons
Command: load 25 ==> 20 tons were loaded
Truck 1 state: (-1, -1), 20 tons
Command: switch ==> truck 2 is now active                <- Truck 2 is now active, next commands apply
Truck 2 state: (0, 0), 0 tons                               to this truck
Command: badcommand ==> invalid command                  <- Invalid command ignored
Truck 2 state: (0, 0), 0 tons
Command: load 12 ==> 10 tons were loaded
Truck 2 state: (0, 0), 10 tons
Command: move 6 7 ==>  truck moved to (6, 7)
```

```
Truck 2 state: (6, 7), 10 tons
Command: dump ==> 10 tons were dumped
Truck 2 state: (6, 7), 0 tons
Command: load 5 ==> 5 tons were loaded
Truck 2 state: (6, 7), 5 tons
Command: switch ==> truck 1 is now active
Truck 1 state: (-1, -1), 20 tons
Command: move 4 4 ==>  truck moved to (4, 4)
Truck 1 state: (4, 4), 20 tons
Command: switch ==> truck 2 is now active
Truck 2 state: (6, 7), 5 tons
Command: dump ==> 5 tons were dumped
Truck 2 state: (6, 7), 0 tons
Command: quit ==> command file finished
Truck 2 state: (6, 7), 0 tons
****Truck 1 summary****
Truck 1 state: (4, 4), 20 tons
Total miles travelled = 32
Total tons delivered = 20
****Truck 2 summary****
Truck 2 state: (6, 7), 0 tons
Total miles travelled = 9
Total tons delivered = 15
```

Switch again, truck 1 is active now

Summary report after "quit" command

Turn in:

Using the Cascade LMS (my.csi.muohio.edu) website, turn in your SOURCE code file for TruckMover. The grader will add the Truck class and data file to test your program. If you make a mistake and wish to upload a new version...use the "Archive" option, and then upload your file again.

Journal:

In the journal indicate the option you are selecting – either 1 truck or 2 truck

Formatting:

Please check that your program meets the following formatting style guidelines:

- ✓ Comments at the top of the code and prior to each method
- ✓ Comments inside methods as appropriate
- ✓ Meaningful identifiers
- ✓ Proper indentation
- ✓ camelCasing for variable and method names
- ✓ PascalCasing for class names
- ✓ Lines of code that do not exceed 70 characters

Scoring:

Scoring for this assignment will be based on:

- Proper file processing (2)
- Proper handling of the various commands (4)
- Keeping track of active cab and switching back and forth(4)
- Proper output formatting (4)
- Proper use of the methods of the Cab class (6)

Scoring for this assignment will be as described above. Proper code formatting is expected. Failure to follow the formatting guidelines or to leave a journal entry will result in deductions of up to 4 points.