CSCI-UA.0101-002: Assignment 5 – Virtual Moped

Due Tuesday, November 14th at 11:59 p.m.

Instructions:

- The project directory folder for this assignment is called A5_project_directory_NYUnetID. Rename NYUnetID with your own NYU NetID. For example, my NetID is gp2442, so I would rename my project folder "A5_project_directory_gp2442".
- The project directory contains a project directory containing four subdirectories, namely bin, lib, test and src. The source files are in src/edu/nyu/cs/NetID. Make sure to rename the subdirectory /NetID to your actual NYU Net ID.
- Complete the code according to the instructions in this document as well as those written as comments within the .java source files (if any).
- **Important:** In addition to completing all problems, you are also expected to compile and run your source code using the Command line from the project directory. Refer to **Lectures 3 and 4** for how to do this. You will lose **five points** if we cannot compile and run your compiled code from the project directory.
- Submit a zip file named "A5_complete_NYUnetID" containing your project folder called "A5_project_directory_NYUnetID". Again, NYUnetID should be replaced with your NYU NetID.

Programming a virtual Moped

In this assignment, you will create a program that allows a user to drive a virtual moped around the virtual streets of Manhattan using the Java command line.

Requirements

The program accepts the following commands from the user (all withoute the double quotes “”):

- “go left”
- “go right”
- “straight on”
- “back up”
- “how we doin’?”
- “fill it up”
- “park”
- “go to Xi’an Famous Foods”
- “help

Driving

- The user simply instructs the Moped to "go left", "go right", "straight on", or "back up", and the moped moves one block in the chosen direction.
- The Moped always starts out its life at 10th St. and 5th Ave., facing South, going forwards.

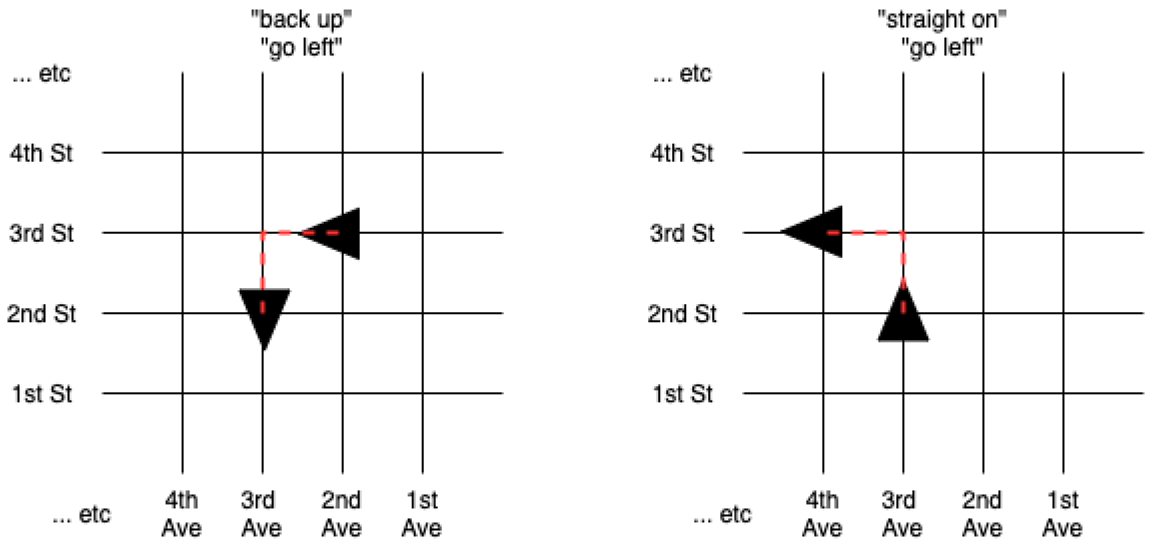

Figure 0.1: Pictorial description of the virtual moped. On the left hand side, the Moped "back up" and then "go left" in reverse. On the right hand side, the Moped goes "straight on" and then "go left".

- The program **must** announce the Moped's current location and orientation (which cardinal direction it is facing), when it starts, and with every move.
- If the user commands the Moped to "park", then the program outputs a message, "We have parked", indicating the moped has been parked on the sidewalk, and **quits**.
- Telling the moped to "back up" puts it into reverse **until it is explicitly told to go "straight on"** again, and vice versa.
- **Important:** Turning left while going forwards is a different thing from turning left while in reverse, hence _your moped must keep track of its orientation_.

Gas

- The moped's gas tank stores up to 1 gallon. It comes pre-filled.
- Driving the moped burns 1/20th of gallon per city block.

- If the user enters the command, "how we doin'?", the program outputs the current level of gasoline in the tank, as a percentage.
- If the Moped runs out of gas, the user is notified that the Moped has run out of gas ("We have run out of gas. Bye bye!") and the program quits.
- To refill the gas tank, the user simply instructs the program to "fill it up".

Homing

The Moped has a special "'go to Xi'an Famous Foods'" command that automatically drives the Moped from wherever it happens to be to **Xi'an Famous Foods** restaurant at 8th Ave. and 15th St.

- The Moped should auto-drive itself one block at a time to the address of Xi'an Famous Foods, outputting its location with each block.
- The Moped must use the same methods to move block-by-block and same amount of gas when homing as when being manually driven.
- **Important**: If the Moped needs to fill up gas while homing, it should do so automatically and then continue.
- Essentially, the only difference between homing and manual driving is that the user is not being asked what the Moped should do, the Moped is deciding for itself.

Location-based advertising

Our Moped is paid for by advertising, and should output ads when it reaches the locations of our current clients:

- At 79th St. and 8th Ave., the Moped should output an ad for the American Museum of Natural History.

- At 74th St. and 1st Ave., the Moped should output an ad for Memorial Sloan Kettering.
- At 56th St. and 3rd Ave., the Moped should output an ad for Tina's Cuban Cuisinerestaurant.
- At 12th St. and 4th Ave., the Moped should output an ad for The Strand book shop.

Help

If the user enters the command, "help", the program should display a list of commands that the program understands.

Assumptions and details

Our program makes a few assumptions:

Geography

- Assume that Manhattan is a 200x10 grid of numbered city streets.
- **Important**: Users must not be allowed to drive their Moped off the grid. If a user tries to go off the grid, the Moped should not move and the program should print a message saying that the Moped has hit a boundary. In addition, if a user tries to move off the grid using "go left" or "go right", the orientation should change as usual. Example 1: If you face north, go north and hit the north boundary, the Moped should not move and still point north. Example 2: If you face north, go left and hit the (west) boundary, the Moped should not move but now point west.

Street naming

- Assume that Streets are named 1st St, 2nd St, 3rd St, and so on up to 200th St.

- Assume that Avenues are named 1st Ave, 2nd Ave, 3rd Ave, 4th Ave and so on up to 10th Ave.

Street directions

- Assume that Street numbers increase as **you go North**.
- Assume that Avenue numbers increase as **you go West**.

Details

Your assignment must adhere to the following requirements:

- Include a class named "Moped" that encapsulates all of the attributes and methods that our moped model needs in order to perform its functions.
- Include a class named "TestDrive" which includes a "main()" method that starts up the program and facilitates the interaction between the user and the Moped.
- Adhere to the object-oriented design principles of encapsulation and abstraction: Make all data fields private; provide 'getter' and 'setter' methods as needed to control the behavior of your objects.