Data Structures
COMP 2100
Fall 2022

Lab 2: Lunar Lander: Wearing the Implementer Hat

In this assignment you will write your own class definition to complete a game program that allows a user to try to safely land on the moon. Your class will keep track of the current state of the lunar lander.

A working version of the application is available in the downloadable jar file.

Your class will keep track of the lander's current altitude, speed and fuel. To keep things simple, we will use simple ints for all calculations and updates will occur once every second. In the lunar landing simulation, the user will have the option to request thrust from the lander's engines. For each thrust request that is made, a certain amount of fuel will be expended to generate a certain negative acceleration. If the user makes more than one thrust request in a given second, then you apply the appropriate number of fuel units. For example, if the user makes four thrust requests in a given second, then you burn four times the normal amount of fuel and give the lander four times the negative acceleration of a single thrust.

Your class is to be called LunarLander and must include the following public methods:

Method	**Description**
getAltitude()	returns the current altitude in meters
getVelocity()	returns the current velocity in meters/second
getFuel()	returns the current number of thrust units remaining
reset()	resets the lunar lander to the initial situation
thrust()	remember to apply one more unit of fuel on the next update (can be called multiple times between updates)
tick()	called once every second, should update the simulation variables appropriately, applying whatever thrust has been requested since the last update

More specification details are available in the javadoc api document for the LunarLander class.

The source file LunarLander.java is the only one you need to edit/modify. Feel free to examine the other 3 source files if you are curious. LunarLanderMain.java is the main application class for the program, and is the client to your LunarLander class.

Remember that individual work is expected on the lab projects!

To Do

1. Determine the instance variables required (the state of the lander).
2. Stub out the public methods detailed in the specifications.
3. Implement each method according to the specifications.
4. Comment the class and each method with javadoc style comments to approximate the contents of the api specification referenced above. (The command line options I used to generate this were "javadoc -package -notree -noindex -nohelp -nonavbar -d docs LunarLanderMain.java" where docs was a subdirectory I created in the source directory.) Your javadocs may not produce an exact replica because of version differences.

If you need more assistance with writing the code than is available in the specifications, I have provided a page of helpful comments. Try to solve the lab from the specs before going to this page.

Points	Item	Description
5	documentation	Use of comments including Javadoc-style comments, use of self-documenting identifier names, and judicious use of white space (such as indentation) to enhance readability.
5	representation	The instance variables that you declare to represent the state of the lander, along with how you use them to solve the problem.
10	results	The constructor, getAltitude, getVelocity, and getFuel are worth 1 point each. All other methods are worth 2 points each.

To Submit: Email your LunarLander.java source file as an attachment to dstucki@otterbein.edu