NBody Simulation

Getting Started with the Simulator (NBody.java)

Create a file named **NBody.java**. NBody is a class that will actually run your simulation. This class will have NO constructor. The goal of this class is to simulate a universe specified in one of the data files. For example, if we look inside data/planets.txt (using the command line **more** command), we see the following:

```
$ more planets.txt
5
2.50e+11
1.4960e+11  0.0000e+00  0.0000e+00  2.9800e+04  5.9740e+24    earth.gif
2.2790e+11  0.0000e+00  0.0000e+00  2.4100e+04  6.4190e+23     mars.gif
5.7900e+10  0.0000e+00  0.0000e+00  4.7900e+04  3.3020e+23  mercury.gif
0.0000e+00  0.0000e+00  0.0000e+00  0.0000e+00  1.9890e+30      sun.gif
1.0820e+11  0.0000e+00  0.0000e+00  3.5000e+04  4.8690e+24    venus.gif
```

The input format is a text file that contains the information for a particular universe (in SI units). The first value is an integer **N** which represents the number of planets. The second value is a real number **R** which represents the radius of the universe, used to determine the scaling of the drawing window. Finally, there are **N** rows, and each row contains 6 values. The first two values are the x- and y-coordinates of the initial position; the next pair of values are the x- and y-components of the initial velocity; the fifth value is the mass; the last value is a String that is the name of an image file used to display the planets. Image files

can be found in the **images** directory. The file above contains data for our own solar system (up to Mars).

readRadius

Your first method is **readRadius**. Given a file name as a **String**, it should return a double corresponding to the radius of the universe in that file, e.g. **readRadius("./data/planets.txt")** should return 2.50e+11.

To help you understand the **In** class, we've provided a few examples for you in the **examples** folder given in the skeleton. The first one is called **BasicInDemo.java**. Take a look at the code, and its input file, **BasicInDemo_input_file.txt**. This program should output: **The file contained 5, 9.0, ketchup, brass, and 5.0**.

There's a slightly more complicated example called **InDemo.java**, which you can also find in the examples folder. While this demo does not perfectly match what you'll be doing in this project, every method that you need is somewhere in this file. You're also welcome to search the web for other examples (though it might be tricky to find since the class name **In** is such a common English word).

NOTE: Do not use System.exit(0) in your code, despite the example using it. This will break the autograder, and you will not obtain a score.

Alternately, you can consult the full documentation for the In class, though it can be found a bit intimidating.

We encourage you (and your partner, if applicable) to do your best to figure out this part of the assignment on your own. In the long run, you'll need to gain the skills to independently figure out this sort of thing. However, if you start getting frustrated, don't hesitate to ask for help!

You can test this method using the supplied TestReadRadius.

```
javac NBody.java TestReadRadius.java
java TestReadRadius
```

readBodies

Your next method is readBodies. Given a file name, it should return an array of **Body**s corresponding to the bodies in the file, e.g. **readBodies("./data/planets.txt")** should return an array of five planets. You will find the **readInt()**, **readDouble()**, and **readString()** methods in the In class to be useful.

You can test this method using the supplied TestReadBodies.

```
javac Body.java NBody.java TestReadBodies.java
java TestReadBodies
```