Exercise Sheet #8

Bulcsú Sándor <sandor@th.physik.uni-frankfurt.de>
Hendrik Wernecke <wernecke@th.physik.uni-frankfurt.de>
Laura Martin <lmartin@th.physik.uni-frankfurt.de>
Christopher Czaban <czaban@th.physik.uni-frankfurt.de>

Problem 1 (*Habitable exoplanets*) 0 Pts

The possible existence of other terrestrial planets is bound to the distance from planets to their sun, respectively in relation to the radiant power. Further, one should mainly regard stars in a long lasting status. The burning of hydrogen to helium is the longest lasting cycle of a stars and it is called the main sequence. Your task: download a chart containing a wide range of information on about 3500 exoplanets from `http://exoplanet.eu/catalog/` in the `dat` format.

Problem 2 (*Streams*) 4 Pts

The table you downloaded in Problem 1 contains data about more than 3000 exoplanets, each described by over 80 quantities and corresponding error estimates. As you are at first glance only interested in the given quantities, e. g. mass, radius, excentricity etc., get rid of all the error estimates. Therefore read in the file formatted using a file stream and string streams and print out only those columns that are actual quantities (no error estimates) into a new file.
Hint: In the first line of the data, i. e. the table header with the column names, you have to deal with the comment character '#'. For reading the columns of every line you may use the `getline` function and specify the tab character '\t' as field delimiter.

Problem 3 (*Classes*) 10 Pts

Now define a class called `Planet` with the following properties:

- a constant public variable `myName` for the name of the planet
- a private string array `myData` for all data of the star
- a static string array `dataName` for storing the names of all data (which are effectively the entries of the header line)

- an overloaded constructor which takes a string of data values (a line of the data file) of the planet, and initializes the `myName` and `myData` variables
- a function that gets the name of the data type (e. g. mass, radius, etc.) and returns the quantity: therefore you have to go through the array `dataName` and compare which entry matches the given argument and return the value of the corresponding entry in the array `myData`
- a function to print all data of a planet in a nicely readable way

Problem 4 (*Array of objects*) 6 Pts

With the results from the previous problems now do the following:

- Import the data from the file you created in Problem 2 and create for each of the planets an object of the class `Planet`. Store them all using an array of pointers called `exoplanets`.
- Now go through your array `exoplanets` and sort all planets with respect to the maximal radius `semi_major_axis`.
- Print the planets for which the maximal radius `semi_major_axis` is inside the range of $1.5 - 2$ AU into a file. How many planets are remaining?

Problem 5 (*Map – facultative*) 0 Pts

Extend the class `Planet` form Problem 3 by adding a `static` function such that you can access an array entry by name.