# CS1003 Week 3 Exercise 1: JSON

As with all lab exercises, this exercise is not assessed. It is intended solely to help you understand the module material. There is probably more here than you will have time for during the lab hours you are encouraged to complete it in your own time if you don't finish it. You will need to look at online resources, especially the documentation of key Java library classes. You are advised to work in Linux.

In this exercise you will work with some quite large data files. Some of the ways you are used to working may fail or be very cumbersome for such files. Basic UNIX tools like `less` and `vi` work well with large files.

The aim of this exercise is to practice writing Java programs that read or write semi-structured data using various libraries. To do this, you will also need to examine Java code written by somebody else and modify it to suit your goals.

## Background

You will be processing a large data file containing information about some of the brighter stars in the sky arising from the Hipparcos satellite mission ( http://sci.esa.int/hipparcos/). The format of the file is documented at http://cdsarc.u-strasbg.fr/viz-bin/Cat?I/311. This is obviously a structured data file and you can confirm this by examining it in a text editor of by executing "`less hip2.dat`" from a terminal. We are going to convert it into two different semi-structured formats. This might be useful for adding additional data or comments about just some of the stars, for instance.

The supplied `HIP2Reader` class provided on `studres` handles reading the file — feel free to have a look at the code. `HIP2Test` is provided to give you an idea of how it is used. `HIP2Test` takes the name of the file to read (`hip2.dat` normally) as a command-line parameter.

## Tasks

1. Copy the supplied files from `studres`. You should have java files HIP2Reader.java and HIP2Test.java, a data file hip2.dat and a Java library file javax.json-1.0.jar. Put all of these files into a new directory inside your home directory (e.g. ~/Documents/CS1003/Exercises/W03Exercise1). From a terminal shell, change into this directory and compile the example files:

```
mkdir -p ~/Documents/CS1003/Exercises/W03Exercise1      (creating the directory)
cd ~/Documents/CS1003/Exercises/W03Exercise1            (navigating to directory)
scp /cs/studres/CS1003/Exercises/W03-1-JSON/* .         (copy from StudRes)
export CLASSPATH=${CLASSPATH}:./javax.json-1.0.jar      (add jar to class path)
javac *.java                                            (compile)
```

   You can then run the provided test file which will process and print the structured data file:

```
    java HIP2Test hip2.dat
```

   Warning: this will produce a lot of output.

2. The instructions above will work if you use any editor and compile from a terminal. If you want to run within an IDE, you will have to instruct it to use the provided library (javax.json-1.0.jar).

3. Write a program to store this data in a file in JSON format. The whole file should be an object with a field "stars" whose value is an array of objects, each with fields "StarID", "right-ascension", "declination", "parallax" and "magnitude". You can adapt the code for producing Json from the output part of `JsonModify.java` which you will also find under examples on StudRes.

You may find it easier to test your program on just part of the data – but once your program is working, do try it on the complete data! You can create a shorter file containing the first 30 lines like this:

```
    head –n 30 hip2.dat > hip2-short.dat
```

One way to write this program is to modify `HIP2Test.java`. You need to replace the printing code with code to write JSON to a file instead. You will also need to add some code at the beginning to setup the `JsonGenerator` (the lines you need for this can be selected from the `JsonModify.java` example on studres) and at the end to write things out.

**Optional Extensions:**

- The distance to a star in parsecs (one parsec is about 3 light-years) is 1000/(parallax in milliarcseconds). Modify your programs to compute the distances of the stars as they read them and include that instead of the parallax in the output. As a further step, if you are interested in astronomy, compute the absolute magnitudes as well.
- The JsonGenerator class that we use for output has capabilities to automatically format the output to make it more human-readable. Look up pretty-printing for JSON.

## Using the Json Libraries from the Command Line

The .jar files will be found by Java if they are put in the same directory and the java compiler run the way described before. If you wish put the .jar file somewhere else (e.g. to re-use it across multiple Java projects), you put it in convenient directory such as `~/jar` and then execute the command

```
export CLASSPATH=${CLASSPATH}:${HOME}/jar/javax.json-1.0.jar
```

`javac` and `java` should now see the extra libraries without any problems.

## Submission

**You are required to submit a zip file containing your exercise attempt to the appropriate slot in the Exercises category on MMS.** Note that if you have done this work on the lab servers as suggested, you will need to upload your submission to your local machine before submitting to MMS.

To recursively zip the contents of a folder on Linux you can do this:

```
cd ~/Documents/CS1003/Exercises/W03Exercise1
```
(if you are not already there)

```
zip -r W03Exercise1.zip *
```

This will create a file called W03Exercise1.zip containing the entire contents of the current directory (files that match *).

Suppose that you have done that operation from this directory: `~/Documents/CS1003/Exercises` on the lab machine called `pc3-002-l`, you can copy the zip file from my local machine using scp as follows:

```
scp pc3-002-l.cs.st-andrews.ac.uk:~/Documents/CS1003/Exercises/W03Exercise1.zip .
```

and you will have a copy of the zip file in the working directory from which you executed the scp command. I can then submit the file to MMS using the web interface.

You could also use FileZilla instead of scp to download the file to your local machine (but you should really master scp).