Programmierpraktikum

Exercise Sheet #11

WS, 2012/2013

Maps

Recall the `Planet` class from a few weeks back. Write a program for manipulating user defined objects within a Map.

Use your planet program and the class `Planet` from exercise #9 for storing the properties of the planets when reading the planet data from a file, as done previously,

- This time, the data is to be stored in an appropriate Map, in key-value pairs {`"PlanetName", PlanetObject`}.
- Pass the Map with the planet data to a static function where the printing of all Planet objects should be done.

Collections

Solve the following tasks using the most useful classes or primitives (according to your judgement) from the Java library. Justify in each case.

1. A large number of random numbers is to be stored. The number is not known in advance.
2. A large number N (known) of random numbers is to be stored, then accessed sequentially for printing.
3. A large number N (known) of random numbers is to be stored, then accessed for printing in an increasing order.
4. A large number of random numbers is to be stored, then accessed for deletion in a random order.
5. A large number of random words is to be stored, then a word is searched for and its index printed, the procedure repeats itself.
6. A large number of random words is to be stored and later printed, but no word should be repeated (the order of printing does not matter).

Note: Check the java documentation to obtain more information about the advantages and disadvantages of each Class.

Large number of threads

Write a program generating a large number of threads and measuring the resulting computational performance.

- Make a runnable class that does some simple calculations if not interrupted. You can use the code in http://itp.uni-frankfurt.de/~gros/Vorlesungen/ProgPrak/Java-threads.html#%286%29
- Instantiate a number from N=1 to N=1000 copies of the runnable class and start their threads.
- Measure the overall performance, the total number of computations performed as a function of the number of threads N.
- Make plots of your results as a function of N. This procedure is useful to find out what is the optimal amount of threads for a given problem in a given machine.

Check with your process manager (in linux, “top” or “gnome-system-monitor”, in mac os the “activity manager”, and windows “task manager”) what is happening with your processor(s) load. This might be interesting only if you have more than one core.