Testing

Lauri Kangassalo

June 16, 2013

General

I have not bothered to test minor things such as do getters work, or do toString-methods return a correct output. Instead I have been concentrating on testing things that could possibly be wrong, such as comparing two byte arrays or returning an element from hash table.

Unit testing

In this project I have used unit testing to test so called helper classes, which are used in many parts of the program and are crucial for the correct function of the program. Examples of helper classes follow: *Reductor*, *CommonHelper*, *HashTable*. I have not used unit testing for the actual hash cracking and table creation, because I don't think that unit testing is suitable for that. I think it would be counter-productive to write tests for them, since the results of hash cracking and table creation are so unpredictable.

The hash table implementation has not been tested with large quantities of data / big tables.

Manual testing

I have tested this project by hand lots and lots. I have tried different combinations of parameters for table creation in search for the perfect ratio between them. For example I've been following the number of different endpoints and different hashes created. You could say that most of this project's testing is done manually. If it does crack hashes, it cracks hashes, if it doesn't, it doesn't.

Performance testing

Most of the time I just ran the program with different parameters and observed how long it would take for it to complete, but lately I've started using NetBeans' built-in profiler.