Problemset 10, Guidelines, idioms, patterns

After your boss at ACME Inc. proudly declared that he made a graphical Ursuppe game, he wonders if the implementation follows the best practices of object-oriented design. You're slightly startled by the question, given that your boss seems to know about as much about programming as your grandmother does, but nonetheless, you concede that the question is fair. To give a fair answer, you decide to read the *design pattern* book by the Gang of Four (ISBN 0–201–63361–2). (It is available online, from within the university of Bern.) Also, you want to take a look at the *cook's tour* paper, to see how patterns can be used to describe software. That paper is in the papers folder in the p2ubungen repository.

Armed with your new knowledge:

- Make sure that the implementation of genes in the Ursuppe game uses at least one pattern from the design patterns book.
- Explain the overall architecture of your implementation of genes, using the patterns that you are using, similar to the cook's tour's explanation. At most 500 words.

To impress the CTO of ACME Software Inc. further, you'd like to tell him that your team has developed the software following strict guidelines. Use these as a starting point, and at at least one more:

> You **must** have at least one test class per non-GUI class, and there should be two to twelve test methods per class that exercise concrete examples of the unit's usage.
>
> Test names **must** be long enough to summarize the entire test case. Have all test methods contain the word with `should`, for example: `blue3ShouldEatRed5`.
>
> External resources **must** be separated from actual code, such that code can be tested in isolation. You **must** make sure that no class is hardwired to use an external resource. You **can** use the "Guice" framework to isolate external resources such as time sources, console output, random generators, files etc. . .
>
> Avoid loops and other abstractions in test code! Tests should exercise concrete examples of program execution.
>
> Tests **must** be silent. There must be no output whatsoever! After running all tests the console must be blank. Avoid interactive tests, please make sure that the tests do not prompt the user for input. Test must run fully automated.
>
> Non-private methods that are used by tests only **must** be marked as such. Use the `ForTestingOnly` annotation to mark such methods.
>
> Each class **must** be commented with at least one sentence that lists *all* responsibilities of the class.