Tests

The following methods are used when testing the program.

Unit Test

A Java framework, JUnit5, has been used within the process. You may consult to com.luox6.conway.test for the written test cases.

Included Sections

- row/col length getters
- class constructor
- getCell cell status with coordinate wrapping
- neighbour live cell counts around specific coordinates
- toString string representation/output format

The above have tested class's construction, mutations, and expected string representation.

Run Unit Test

You may use your favourite IDE integration with JUnit. Please consult JUnit manual However, if you would like to use ConsoleLauncher, after compiling the program following the instruction in readme.md, try the following command for unit test result.

```
$ # The following bash command should run at project root directory
$ java -jar deps/junit-platform-console-standalone-1.7.0.jar -cp ./src --scan-class-path

Thanks for using JUnit! Support its development at https://junit.org/sponsoring


╷
├─ JUnit Jupiter ✔
│  └─ ConwayMapTest ✔
│     ├─ getMapRowLength() ✔
│     ├─ isValidCoordinate() ✔
│     ├─ getMapColLength() ✔
│     ├─ testToString() ✔
│     ├─ getCell() ✔
│     └─ neighbours() ✔
└─ JUnit Vintage ✔

Test run finished after 89 ms
[         3 containers found      ]
[         0 containers skipped    ]
[         3 containers started    ]
[         0 containers aborted    ]
```

```
[        3 containers successful ]
[        0 containers failed     ]
[        6 tests found           ]
[        0 tests skipped         ]
[        6 tests started         ]
[        0 tests aborted         ]
[        6 tests successful      ]
[        0 tests failed          ]
```

Manual Test

However, the unit test cannot cover some corner cases, e.g.:

- Seed file parsing
- IO/Input validation
- …

In order to test these cases, here we have some crafted files, which are also listed in manual->Example Args section. These cases have covered the following topics:

- example1.txt -> Given test file from homework description
- example2.txt -> Wrapping, neighbour counts, simple, mostly static case
- example3.txt -> Wrapping, neighbour counts, complex moving case
- invalid1.txt -> Seed file with < 2 rows
- invalid2.txt -> Seed file with < 2 columns
- invalid3.txt -> Seed file with undefined char
- invalid4.txt -> Seed file with inconsistent rows
- invalid5.txt -> Empty seed file

Commands to run these examples have also listed in the manual. It is believed that these cases cover most of the common mistakes that users would make.

Game rules are also tested at example2.txt and example3.txt by stepping through the generated files.

You may find the expected output in tests/result section. Use diff for the file generated from the program with corresponded command – these should be identical or it means something goes wrong.

Some examples inspired by Wikipedia.