QA CONCEPT

To assure the quality of our game we have the following plan which is quickly summarized and then looked at in detail below.

1. CONSTRUCTIVE QUALITY MANAGEMENT

1.1 TECHNICAL PROCEDURE

1.1.1 STICK TO THE RULES

For the software itself we closely stick to the requirements list Lina wrote. It gets updated as the project continues and the requirements get more detailed.

1.1.2 WHAT DID YOU DO?!

We ensure that the code is sufficiently documented and complex thoughts and methods are explained in detail. The Javadoc covers the main function of the method while the single-line or multi-line comments explain the implementation in detail. We stick to the conventions here.

1.1.3 LET'S BE A LITTLE LAME

Speaking of conventions: we stick to the coding conventions to ensure cohesive, Printerest-worthy, sexy code. It also makes reviewing easier.

1.1.4 NO EXCEPTIONS FOR THE EXCEPTION HANDLING

We like to play catch, so we throw around exceptions a lot. And with a lot we mean: where it is needed.We catch them as well, of course, and handle them in a meaningful way (meaningful depends on the context).

1.1.5 UNI-T TESTS

No, we do not test the uni dance group but the most important methods and its exceptions for each class. We want to ensure that our code runs properly with writing tests while we are coding. When we do both simultaneously, the workload gets evenly distributed. Proper energy and (human-) resource management we do here, y'all. We also have some conventions set for our unit tests.

1.2 ORGANISATION

1.2.1 NO TASK IS LEFT BEHIND

There is a Google Excel sheet (which is only meant for our group to see) with a responsibility assignment of each task of the milestones. The person responsible for a task can then update the status to: will do, in progress or done. This ensures that each task is handled and everyone can see if someone can't handle the workload or has not enough work to do.

1.2.2 THE LIST GOES ON

We have a checklist for reviewing the code:

1. Did the team member sufficiently document the code?
2. Did the team member stick to the code structure we agreed on?
3. Did the team member stick to the conventions we agreed on?
4. Did the team member write only code we can actually use?
5. Did the team member structure the code in a useful and logical way?
6. Did the team member think of exceptions and throw them?
7. Did the team member catch exceptions?
8. Did the team member write unit tests for the code?
   Is there a unit test for the most important methods in the class?
   Is there a unit test for the exceptions in this class?

We then either fix the code right away or give the developer feedback.

1.2.3 BUG AND BEES

We keep a ticketing list (also a Google Sheet) for the bugs in the software that need to be handled and that should be our to-do-list for more time-consuming bugs.

2. ANALYTICAL QUALITY MANAGEMENT

2.1 ANALYTICAL PROCEDURE

2.1.1 A SECOND PAIR OF EYES

Every important snippet of code will be reviewed by Riccardo and Anisja. We are the QA-police that do not play any games (except for playing catch with exceptions ... and of course Kniffeliger).

2.1.2 "IT'S A TEAM EFFORT"

We plan to go through the most important parts of our code every second team meeting. That way we can all get an idea of what the other person coded and can easily review it if necessary.