Riccardo Grieco, Anisja Mayer
Lina Mehrle, Dominique Ostermayer

Detailed Quality Assurance Conecept

1 Constructive Quality Management

1.1 Technical Procedure

1.1.1 Guidlines

Our software's guide is the requirements file authored by Lina for milestone 1. It encompasses crucial rules governing the game logic, GUI, and network protocol. This document serves as the cornerstone of our software development, providing a comprehensive reference for all code. As we progress with the project and delve into finer details, each team member is responsible for updating this file. All decisions that need to be done are written down in this file.

1.1.2 Documentation

We place a strong emphasis on thorough documentation throughout our codebase to ensure clarity and ease of understanding for all team members.
Our approach mainly involves using JavaDoc comments to provide concise summaries of the main purpose and functionality of code. These comments serve as a quick reference point for developers seeking to understand the role of a particular method. Moreover, consistency in our documentation style ensures ease navigation in our codebase, promoting collaboration and efficiency in the developement process.

1.1.3 Coding Conventions

We adhere to coding conventions to maintain cohesive, visually appealing, and easily reviewable code. Following these conventions ensures that our code is not only aesthetically pleasing but also facilitates smoother code reviews. We stick to the IntelliJ default coding and formatting conventions.

1.1.4 Meaningful Exception Handling

We find ourselves navigating exception handling much like a game of catch – tossing them where necessary and ensuring we catch them as well. It's a fundamental aspect of our coding practice, allowing us to address expected issues that can occur promptly and effectively. Whenever exceptions occur, we take care to handle them thoughtfully, considering the specific context in which they arise. This approach ensures that our code remains robust and resilient, even in the face of unexpected errors.

1.1.5 UNIT Tests

To assure the functions and quality of the game, especially the networking and the game logic, we test every class with unit tests. Here is our unit test conventions list:

1) Only important/Fundamental methods get tested.
2) Exceptions always get tested. Those tests are named `classname + ËxceptionsTest"`.
3) The test class is named like the original class + `Test"`, for example: `Dice.java` has the test class `DiceTest.java`.
4) The method that gets tested is called like the original method + `Test"`, for example: `fullHouse()` is the original method and `fullHouseTest()` is the test method.
5) The correct structure and naming of the test files is ensured with the `CRTL + Shift + t` combination, that gets pressed in the class we want to get a test class for.
6) No happy testing and ensure random unit tests.
7) We use lambda expressions in tests so they run through and do not stop when the first test fails.

1.2 Organisation

1.2.1 Shared Allocation of Tasks

We maintain a Google Excel sheet accessible only to our group, serving as a central hub for task assignment and tracking across milestones. Each task is assigned to a responsible individual, who updates the status as “will do,” “in progress,” or “done.” This system ensures that every task is accounted for and progress is transparent to all team members.

1.2.2 Bug Control

Within our project management toolkit, we maintain a ticketing list dedicated for tracking and addressing bugs within our software.