Software Engineering Project Write-up
Spring 2020

Introduction

Welcome to the Beer Game! In the next few sprints, you will be implementing the beer game using the requirements specification of Team 8: (Find the latest version here: Link).

At the beginning of every sprint, you will be paired up with a new partner and you will receive a new code base which you will have to further develop. You will **not** be given detailed instructions on what to do in each sprint, this is up to the team. Grading of the progress in each sprint is based on any advances you make, and not only the feature implementation. You earn points for adding/improving features, adding/improving documentation, adding/improving test cases or even working on the build process. In line with the definition of "software" discussed in class, all these aspects contribute to the quality of the final software product; and would hence give you points.

As part of your learning process, you are expected to adopt the XP development process to complete the sprints. You're also expected to do "pair programming" with your teammate.

Tools

The project will be implemented in C++ and Qt, supported by the Qt IDE, and CMake. For automated generation of code documentation, you will use Doxygen. You need to install several tools constituting your programming environment, based on freely available components; frequently you can choose between installing the tools by source code for your own compilation or prefabricated packages for installation through a package manager.

The following tools are required:

- GNU C++: https://gcc.gnu.org/install/
- Qt: https://www.qt.io/download
- CMake: https://cmake.org/install
- Doxygen: https://sourceforge.net/projects/doxygen/
- Google Test: https://github.com/google/googletest
- Operating System: Linux, macOS

For version control and code submission, you are expected to use the GitHub repository which you received access to.

Primer

- Qt: https://doc.qt.io/qt-5/qtexamplesandtutorials.html
- CMake: https://cmake.org/cmake-tutorial

- Doxygen: http://www.doxygen.nl/manual/starting.html
- Google Test: https://github.com/janoszen/clion-project-stub/blob/master/gtests/googletest/docs/Primer.md

Criteria checks for grading:

1. Check if code compiles without errors (*will be checked automatically*)
2. Check progress
    a. Check if new functionality been added (*will be checked using TA-provided test cases*)
    b. Check if GUI features have been added (*will be checked manually*)
    c. Check if documentation has been added (*clear description of new implemented features in Doxygen generated documentation; will be checked automatically*)
    d. Check if test cases have been added (*number of tests and its code coverage compared to the last sprint will be used; will be checked automatically*)
3. Check code quality (*including proper commenting, class structuring, modularity, exception handling, etc.; will be checked manually*)
    a. Program should be broken down to smaller logical units as appropriate
    b. Information should be properly passed from one component to another with the appropriate use of parameters and return values
    c. Global variables are not used unless absolutely necessary
    d. Code is readable and reusable
4. Check if improvement have been added to previous sprint's codebase (*checked manually*)
5. Check for plagiarism (*checked automatically using https://theory.stanford.edu/~aiken/moss/. Codebases with a significant amount of plagiarism will immediately receive the lowest possible score.*)

Grading

There isn't any fixed amount of features you have to complete for each sprint. Unlike other programming classes, here, you are free to work on whatever feature you want to first.

The lowest possible score in each sprint is always given when no improvements have been made. For teams that do make improvements, a score is calculated using the criterias detailed above. The more criterias you fulfill, the higher score you get. The score of the highest scoring team will be used as an upper limit for that sprint. The rest of the teams will be placed relatively after that.

Testing

The TAs will provide you with their own test cases (using Google Test). Use these tests to check if the feature you created works or not. This will also be used for grading, in order to automatically check which feature you've completed.

You must also write your own unit tests whenever possible (again, using Google Test). As mentioned in Criteria 2d, the higher the number of test cases and its code coverage, the higher the score you'll receive.

References file

Cite all the help you get in a references file called references.txt. This document must be included in your submissions in the root folder. Append to it as the project progresses. Describe the contribution of each source as a JSON parseable text.

Here is a sample format for the references file:

```
{
      "online-references": [{
                  "url": "https://www.geeksforgeeks.org/bubble-sort/",
                  "contribution": "Implementation for bubbleSort()",
                  "file": "main.cc"
            },
            {
                  "title": "https://www.geeksforgeeks.org/quick-sort",
                  "contribution": "Implementation for quickSort()",
                  "file": "main.cc"
            }
      ],
      "colleague": [{
                  "jacobs-id": "f.mcfreshface1",
                  "sprint": "2",
                  "contribution": "Helped with bubbleSort() algorithm",
                  "file": "main.cpp"
            },
            {
                  "jacobs-id": "f.mcfreshface2",
                  "sprint": "2",
                  "contribution": "Helped with quickSort() algorithm",
                  "file": "main.cpp"
            }
      ]
 }
```

You may use https://jsonlint.com to help format and validate your references file.

Deadline

The usual deadline will be in two weeks after the write-up has been released (on Thursdays, at 22:00). As sometimes there may be technical problems, there will be a grace period of 30 minutes, during which we still accept submissions. After 22:30, the TAs will download your repositories, and the last uploaded version before 22:30 will be graded.

Other information

- If your team member doesn't contribute to the sprint, send an email to the TAs. Grades will be given on an individual basis.
- Office hours will be held on Tuesdays 2 to 3 pm (Rahul) and Saturdays 2 to 3 pm (Iulia) at the IRC GSA.
- If the provided test cases do not cover the feature you're working on, send an email to the TAs. A test case will be added to cover that feature. Then, everyone will be notified through the mailing list. **This is only possible until a week before the deadline!**
- Grading for all sprints is based on an improvement based metric (based on improving the codebase you received) rather than a task based metric (based on completing a fixed amount of tasks). This way, in case you receive a repository that hasn't accomplished much, you will still be graded fairly according to the improvements you make. Otherwise, you would have had to implement tasks for two sprints, the previous one **and** the current one.
- You must not make any code base used in the project available to the public (e.g. store it as a public GitHub repository).
- Using repositories you've already worked on in the previous sprints is not allowed.
- Remember to use the class diagram specified in Team 08's documentation, as our test cases will also be based on it. If you have any issues with the documentation, email the TAs.