Git Quality Standard

Branches

Individual branches: Each person on the team has a branch named after them. Any code you write should be pushed to one of these branches. While you are not required to only work on your own branch, they are primarily made for the person it is named after. In other words, it is perfectly acceptable for two or more people to work on the same branch if they are working on the same task, or if it is more convenient to do so. The code in these branches does not necessarily have to be bug-free, but bugs should still be cleaned up before the code is merged to the Production branch.

Production: When a task in an individual branch is done, a pull-request is used to merge to this branch. This is effectively where our work for a given week is summarised. Any runtime conflicts or bugs caused by combining the code from different branches should be resolved here. Direct additions to the code in this branch is generally only allowed in the case of fixing these conflicts/bugs or adding/altering documentation.

Main: The main “output” of our project. Any code additions to Main must be done via pull-request from the “production” branch. Do NOT push any code directly to this branch. The code in main must be compileable, and should be as bug-free as possible. It should also be free of any debugging code (lines that print “hello” to the console, for example).

Commits & Pull Requests

The message/description of a commit should not contain gibberish or vague statements (“asdf” or “some changes”). Instead, write one sentence that accurately describes the changes/additions. The frequency of commits should also be kept somewhat high; instead of using one commit for an entire task (“added & integrated class B”), try to commit your code after one or a few major step(s) (“Created class B”, “Integrated methods from class B into class A”, “Bugfixing regarding class B”, “Added documentation for class B”, for example).

Pull Requests feature both a title and a description; These should describe the overall task that you have performed, but the description does not have to be particularly lengthy.
When you make a pull request, your code should comply with the documentation guidelines below.
You yourself should not be the one to finalise the pull request. Instead, let it be reviewed by other team members.

Documentation

In terms of in-code documentation, all java files (or other files we actively work on) should contain the following:

- The top of the file should contain a comment block describing the file (class) as a whole: its general function, as well as its connection to other files. This comment should also contain the names of the people who created/worked on the file.
- Each method should have a comment describing its purpose, unless this is blatantly obvious (single line get-methods, for example). *Should further details (expected inputs, details of outputs) be required/recommended?*
- If you think any part of the code is especially complex/not obvious, a comment is recommended.
- variables and methods should be named in camelCase (ie. variableNameExample)
- Similarly, variable and method names should be relevant to the code by the time it is merged to Production/Main. (avoid/rename ‘test’, ‘foo’ and so on)

Git Quickstart Guide

Git is a system used that allows multiple people to work on the same programming project at the same time, using a server to store the code. In our case, the server we’re using is GitHub.

For the sake of this guide, I’ll be assuming you’re accessing GitHub using GitHub Desktop. There are other ways of using git, but those tend to be more complicated to use.

Honestly, It's been a while since I first installed GitHub Desktop, so I’m not too sure about the startup process. However, you should be asked to log in using your GitHub account. If it doesn’t happen automatically, go to File -> Options using the top menu and sign in there.

With that done, you’ll need to download our repository (“repo”) to your machine. On the right side of the window, there should be a list of all repositories that are available to you. Click on puma-telex-machine/AgileSoftwareProject and press the clone button. In the next menu, you can select what folder you want the project placed in, if you care about that sort of thing. You’ll now be on the main screen of github desktop. At the top, you’ll see three large buttons: Current repository (AgileSoftwareProject), Current branch (Main) and Fetch origin (Never fetched). If you read the Git Quality Standard above, you should know that actually working in Main is a bad idea. Go ahead and press the Current branch button and switch to your own branch, for the sake of safety.

Changes

Whenever you add, remove or alter any files in the project folder, the altered files will be displayed in the “changes” tab on the left-hand side of the screen. Left-clicking on the file will show you what changes were made, and right-clicking will, unsurprisingly, show a menu with additional options, including the option to undo the changes to the file.

Commit

In order to properly save your changes, you have to perform a “commit”. At the bottom of the Changes tab is a box containing two text boxes and a button. Write down a sentence that

you feel describes the changes that you've made in the "Summary" textbox, make sure that all files you want to save changes to are marked with a check mark, then press the "commit to **[branch name]**" button. The "history" tab should now also be updated with your new commit.

The Fetch button, Pushing and Pulling

Any commits you create will technically still only be local: other team members can't see them yet. If you have a local-only commit on your system, the "Fetch origin" button will change to "Push origin". Press it to update the online version of the branch. If the version of the branch on your machine is outdated (someone else has pushed commits to the branch), the "Fetch origin" button will change to "Pull origin". (This change may not happen automatically, so go ahead and press the fetch origin button whenever you start up the app or switch branches, just to make sure). Push the button to update your local files.

If two people have made changes to the same file, there's a chance that a conflict will occur when you try to push/pull. To resolve it, you'll be shown both versions of the file and will be asked to combine the two. It can be a bit confusing to describe, so my advice is to just google it if you need more info.

Pull Requests

When your task is done, you'll have to send the code from your branch into the Production branch. This is done using Pull Requests. In the top menu, go to Branch -> Create pull request. This'll open up a browser window. On the screen, you'll see a bar with two buttons: "base" (with Main selected) and "compare" (with your current branch selected). Change "base" to the Production branch. After that, you can fill in the request description according to our standards above, then press the green "create pull request" button. After that, your work is more or less done; the rest of the team will review your code and integrate it into the Production branch.