GitHub Flow and Code Review

In this recitation, you will learn GitHub flow and code review to improve your use of GitHub in a team. GitHub flow is a branch-based workflow common for projects that use GitHub. Code review is a systematic examination of code with the purpose of evaluating a specific contribution for correctness, building a shared understanding of the code, and providing learning opportunities for the contributor and the reviewer.

You will work in pairs in this recitation. In your pair, you will make a contribution to a project and also review and accept another's (your partner's) contribution.

There is a sample code review checklist at the end of this document. You might find the documentation on GitHub helpful, especially the documentation on forking repositories and GitHub flow:
https://guides.github.com/activities/forking
https://guides.github.com/introduction/flow

Instructions

1. Find a partner for this recitation.
2. One team member of each pair should fork the CMU-15-214 Recitation 10 repository:
   https://github.com/CMU-15-214/Recitation10
3. Both members should clone the repository. The repository has a copy of the game framework from Recitation 9, and each team member will contribute a new game. One team member will contribute a "Money Door Game" in which players guess the doors with the most money behind them. The other team member will contribute "Sort with Swaps", in which players try to sort numbers while using as few swaps as possible. To ensure you have enough time for code review, we have provided code for each of these games:
   MoneyDoor: https://goo.gl/bszARh
   SortWithSwaps: https://goo.gl/A3fwaU
4. To make your contribution, create a new branch, commit your contribution to that branch, push the branch to GitHub, and create a pull request to the master branch of the repository. Simply copy the appropriate code from the link above to make your contribution; the code is not high quality, but do not fix your submission before your partner has a chance to review it.
5. Perform a code review on your partner's pull request. To provide feedback to your partner, use global comments on the pull request as well as comments on specific lines of code in the pull request.
6. After your partner has provided feedback, pick one feedback point to address. Address the comment by making a local change to your code, commit, and push it to GitHub. This will add your commit to the current open pull request. After you have addressed the feedback, your partner can accept the pull request.

A sample code review checklist

- Functionality is implemented in a simple, maintainable, and reusable manner.
- Code uses descriptive and meaningful variable, method and class names. Code understanding does not rely too much on comments.
- Class and functions should be small and cohesive.
- The contribution avoids duplication of code.
- Functions should not take too many input parameters.
- The contribution uses a standard code format. The contributor should use a standard style template and/or a linter.
- Variables are declared with the smallest possible scope. Code does not preserve or create variables that are not used again.
- Code does not contain needless and/or commented-out code, such as debugging comments.
- Classes are final and objects are immutable where possible.
- The accessibility of the packages, classes and its members are minimized.
- Variables and method declarations use appropriate and correct types.
- Value classes properly override the `equals` and `hashCode` methods. All classes override the `toString` method.
- Code does not force clients to use special processing by, e.g., returning null rather than an empty collection.
- Contributions should include appropriate unit tests that increase our confidence in the correctness of the contribution.