CSE 403, Project Phase 5: V2 (20 points)
Version 2.0: Maintenance, Refactoring, Localization

For your fifth and final project phase, you will perform work with a focus on code **maintenance**. You will add a **localization** feature to the system and also perform some **refactoring** on your project's code. You have less time to complete this phase than past phases, but it is generally less work to be done and is worth fewer points.

Wiki Page:

Please create a **wiki page** in your repository that contains information to the grader about this phase. Generally speaking this wiki page will contain details and instructions to the grader about the work you have done on this phase. The following sections of this spec will describe the details of exactly what information should appear on that wiki page.

Localization:

Make your application's user interface capable of displaying itself in at least one major spoken/written language besides English. This will require you to make two major changes to your code base:

- **internationalization** (i18n) of your code base so that it is general and not tied to any language
- **localization** (l10n) of your code to the second target language

Internationalizing your project typically involves removing strings and locale-specific references from your source code and converting the code to look up text and other information in **resource files** that are provided for each language. Localization involves creating those resource files and filling them with strings and information for a given locale. As much as possible, please follow the localization idioms and standards of the platform(s) on which you are developing.

To earn full credit for internationalizing your app, all English-specific text, strings, HTML content, etc. should be absent from your source code, databases, text files, and other resources as much as possible. Your database might contain user-submitted data in a specific language, such as messages written by a user in English. It doesn't make sense to try to localize that data; the content to localize is the text that is part of the UI of the app itself.

Your localized text strings can be **fake** or **incorrect**. We do not expect you to actually be fluent in, nor actually translate your entire application to, another written language. So your resources for the second language can be anything you like, as long as they are clearly different from the English ones. For example, a string of "Hello" in English would be properly localized in Spanish as "Hola", but you could simply translate "Hello" to "Howdy". If you do want to try to actually localize your app to correct strings in the other language, you are welcome to do so; but it is not part of your grade.

Most of the internationalization is done by looking up strings in resource files. But you should also localize information such as **dates**, **currency amounts**, and **numbers** for the second locale.

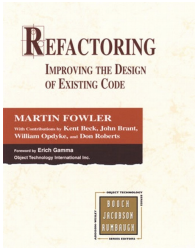
Depending on the nature of your app, if it contains a very large amount of text, it could be a prohibitively large amount of work to localize it. If your group feels that this is the case for your app, please contact your customer TA to discuss the situation. If a particular app is too challenging to localize its text fully, we will come to an agreement with your group about an appropriate subset of the text that should be localized.

If your project is a mobile app, it should **detect the current language** of the device, and if your code is localized into that language, it should display text and other resources in that language as applicable. (Android generally does this for you if you provide your resources files in proper directories.) You should choose to localize into a language that can be easily selected and tested in the mobile emulator. If your project is a web app, auto-detection of the user's locale is difficult, so the site should provide some way to view the site in the other language. For example, you could introduce a user settings page with a menu of available languages. If it is a mix of multiple app types (e.g. mobile+web), localize all components.

In addition to internationalizing the text of your app's UI, we would also like you to internationalize at least one **non-text resource** used by your app. For example, you could choose to localize at least one **image** that appears on the UI, if your UI has any images on it. Or you could localize the app's **layout** or color scheme for the other locale. As with the text, the localization does not need to be correct text, but it should be clearly distinct from the English version of that resource.

On your **wiki page** for this phase, please explain what language you localized your app into, and what non-text resources were localized. Also provide brief step-by-step instructions about how to use and test the app in the second language.

Refactoring:



No app's code is perfect, particularly not at "V1". So as part of this phase, perform a non-trivial **refactor** of some portion of your app's code. Recall from our reading and lecture that a refactoring is when you improve the design quality of a piece of code without changing its external behavior at all. Your refactoring work should have the following properties:

- It should be done over a **non-trivial portion** of the app's code base, say, at least a few significant classes. To be sure that you have chosen a suitably large portion of the code, your group must **contact your customer TA** and briefly confirm what code you will refactor.
- When choosing what part of the code to refactor, choose code that has non-zero **test coverage**. These existing tests should hopefully help to ensure that the refactoring work is less likely to lead to regressions. You may need to refactor the tests' code to match the refactored code under test, if class/method names change.
- At least two members of your team should be involved in the refactoring work.
- Perform your refactor in **code review style**, by making comments in your repository on the files in question about changes that should be made, and then make those changes to the code.
- As you do the refactor, make note of any specific flaws or "**code smells**" that you see in the code that should be changed. Use names such as those found in our reading and lecture on refactoring.
- As you check in fixes for the flaws, make note of specific **refactoring patterns** that you use to improve the code. Use pattern names found in our reading and lecture on refactoring.

On your **wiki page** for this phase, explain what part of the code was refactored and briefly how it was changed. Be specific; list the files and features that are affected and the checkins that correspond to those changes. Also very briefly recap the changes you made to the code, such as listing some of the code problems/smells and refactoring patterns used.

Other Details:

On your repo/wiki, your group should supply a **V2 binary** representing your new fully built version of your app. The previous versions such as V1, Beta, etc. should still remain available at their previous individual URLs or locations.

Our intention is for this phase to be a medium amount of work, not a large work task. So if you feel that the refactoring and localization are taking your group an enormous amount of time, it is possible that you are doing more work than we intended. For example, you may have chosen too large a portion of your app to refactor and could get by with a smaller subset. Please contact your customer TA if you are unsure.

You **do not need to meet with your customer TA in person** during this phase, though you should quickly contact them by email to confirm what portion of your app is to be refactored, as stated previously.

Relationship Between V1 and V2:

We want you to feel that the work you put in on V2 is beneficial to your project. Therefore we will wait until you turn in your V2 before grading your V1 work, and unless you tell us to do otherwise, we will look at your V2 code when grading V1 and TEST2. This gives you an opportunity to avoid some potential deductions, because any problems that existed at the time V1/TEST2 were due but are fixed in V2 will not be counted against your V1/TEST2 score. Note that this is not a "re-submission" of V1 to earn points back on it after grading; this is a chance to make fixes in V2 prior to the V1 grading.

So for example, if part of your code was poorly designed in V1 and would have lost points, but you refactored it to have a good design in this phase, we will give you the higher score on your V1. If you want to perform other improvements to your code during this phase to help improve your V1/TEST2, such as fixing bugs or cleaning up other code, you may do so, though this is not required. You won't know the exact V1/TEST2 grading criteria before submitting this V2 phase, but you can make reasonable guesses about what we will look for based on those corresponding specs.