Improvement of Design

In phase 2, we put in extra effort to make our program more extendable. This was achieved by major refactoring in two directions.

1. **Using HashMap For Storing Rating**

   In phase 1, we used to have the ratings of different review categories stored as separate attributes in each review entity. This was largely inconvenient if we wanted to edit a review and made the classes extra crowded with functions that would only be used once. This also meant that whenever we want to add a new review, we'd have to pass in a very long list of arguments to the review constructor - which is a code smell. Hence, we decided to store the ratings in a hashmap of strings (all of which collected from user input). This not only allowed us to eliminate the code smell of having two many parameters, but it also made edit review a lot easier to implement. Instead of calling the setter of a review category, we can simply use the HashMap.put(key,value) method to update the new rating of this review category. Instead of having 30ish individual setter methods, we now have only one editRview method in each ReviewManager, which takes in a String specifying the review category that the user wants to edit. The code looked a lot cleaner and readable after the change.

2. **Organizing Prompts Into Dictionaries**

   Our presenter from phase 0 and phase 1 stored all the prompts and feedback messages as hard-coded Strings. This was very inconvenient if we wanted to make our program accessible to non English-speakers. In order to achieve this, we decided to make an abstract class called PromptDictionary, which has enum classes that store the prompt categories, and a HashMap that maps these prompt enum types to the actual Strings that we want to print out to the user. We made two child classes of Prompt Dictionary, a Chinese and an English version, so that these prompts are stored in both languages. All the hard-coded print messages in presenters are replaced with getPrompt calls that gets the corresponding prompt in the appropriate language depending on which PromptDictionary is used (which is specified in the beginning of the program using the LanguageSwitcher class asks the user to select their preferred language). This design change made our program a lot more extendable because if we want to add more languages, the presenter classes will not have to change at all. All we have to do is add a new PromptDictionary subclass and add that option to the LanguageSwitcher.