COS226 – Term Project
Nathan Mathis, Crockett Lalor, Max Geffken

The Hearthstone Deck Builder is a utility application for fans of Blizzard Entertainment's video game Hearthstone: Heroes of Warcraft. The application serves two main purposes. The first of these is to act as an attribute-based search engine for all cards in the game. Upon starting the app, users are presented with a search bar, into which they can specify attributes to search for (the flags and search-able attributes are in the included README file). Hitting the search button populates a list with matching card names, and clicking through this list allows the user to view a picture some extra information about the selected card. The interface is meant to be fairly simple and easy to understand – as a result we mimicked in spirit the command language and argument specification of a typical command line program, such as git, nmap, or top, or any other a user might find and make use of in a typical GNU/Linux distribution. It also simplifies our work, and allows for much less interpretation/parsing logic than the alternative (which would be a Google-style search term). We also added the functionality for the user to add returned cards into a deck list. This gives the user of our application the ability to plan out decks and to tinker with the inclusion of different cards in their list.

The second purpose is to generate high quality decks for the user. The ranking of one card versus another is extremely subjective, and very rarely do two people agree on which cards are better than others in what situations – this is in fact where a lot of the skill in Hearthstone comes from and what makes hearthstone matches interesting/exciting. However, we did not have the time or determination to create a program to intelligently design decks in such a way as a human being would (to do so would be an impressive feat of engineering indeed), so we generate the decks based on a well-respected and heuristically determined so-called “tier list” of relative card rankings from http://heartharena.com/tierlist. The list is fairly accurate and there are many who rely on it when building Hearthstone Arena decks. Therefore, we deigned it “good enough”, and decided to build the deck generation around it. It should be noted that, at this point in time, this feature is not complete.

That being said, our application is very, very close to complete and is starting to look pretty slick. It's fast, and runs smoothly though there are a few bugs that we are aware of:

1) Removing a card from the deck list crashes the application

2) Words in the card description don't wrap correctly and the whole app re-sizes if the text gets too long on a line

3) Some strange cases of strings not being represented correctly (such as having “[...]” around the name of a card, resulting from incorrect JSON parsing)

4) Multiple successive searches causes the application to crash

All things considered, we've come pretty far. These bugs are obviously not at the moment our priority and they aren't huge issues. They seem fairly approachable/straightforward, and we can afford to worry about other things.

Notes:

The API we are using is omgvamp's hearthstone API, whose website is http://hearthstoneapi.com and whose mashape is https://market.mashape.com/omgvamp/hearthstone.

HTTP requests are made using the open source library Unirest, which can be found at http://unirest.io/java.

The build process, including resolution of dependencies, is managed by Maven, whose website is https://maven.apache.org

None of us had ever quite built a project like this, and merely getting our program to a state where it was actually consuming the API was actually quite difficult, and required quite a bit of tedious configuration and reading of Unirest/Maven documentation. Additionally, we had some trouble with version control. We decided on git early on, but didn't actually need or use it until later on in the

process. It wasn't difficult to integrate, and git is simple and pleasant to use, but as one of our group members had never before used git at all, it was necessary to give him a sort-of crash course in git (which of course can be very confusing the first time through). Fortunately, we created a trello board to act as a kind of makeshift scrum board so that we could keep track of what needed to be done and where we were in our project. Because of the effort put into process and project management, everything thus far has gone quite smoothly.