Kevin Leuthold

Professor Kanemoto

CPSC 39

12/7/24

My final project is a text based application that provides information about Magic: the Gathering cards on either a single card or a whole deck basis. When searching for a single card, the program will take an input from the user and search through my map of Card objects for any matches. After prompting the user for a selection from the list, displayed with a small preview of the card's info, the program will display a more in-depth summary of the card with information such as price, mana cost, card type (creature, instant, sorcery, land, enchantment, artifact, etc.), a creature's power and toughness, a planeswalker's loyalty, rulings information, alternate printings, etc. The user can also have the program take in a .txt file containing deck list information, which is easily obtainable from any popular deck hosting website (such as archidekt.com) and provide relevant information about the collection of cards, such as average price, average mana cost, as well as a short bit of information about each card in the deck. This is useful for people who would like a quick reference to a card at a moment's notice, possibly for rulings information in the middle of a game. The deck analysis function can also be useful for building new decks, as distribution of factors such as mana cost and mana color impact what cards you may want to put in a specific deck. In the future, I would like to add the ability to add cards from the card search functionality to a decklist and then be able to export that file to be used elsewhere.

The algorithm that impacted my program the most was probably the one I used to import and objectize my Card objects. To do this, I first had to learn the Google .json library, also called GSON. With GSON, I was able to use its JsonReader class to help parse the more syntax-heavy .json format. After parsing each json object for all of its data fields using a switch(case) statement, I used the Java Collections HashMap class to store my card objects. Another algorithm that I used that was important to my project was my decklist read function. Similarly to my other algorithm I had to read in a file, however the file I have to read is plaintext instead of a .json file. The .txt file is of the format "4x Card Name" for each card present in the user's deck, so I had to find a way to split the string into an integer quantity and a String card name. To do this, I first tried to use String.split(), however this did not end up working for me as I was unable to access both halves of my data after splitting. I asked ChatGPT for assistance, and it told me to use the java.regex library's Pattern and Matcher classes which worked much better. The third algorithm I used in my code was to search a card based on user input. I first took in a string from the user, which would be used as an input to search through the names of each card in the HashMap and return an ArrayList with each matching result using String.contains(). This allows for fuzzier search terms (i.e. searching "elves" will return both "Llanowar Elves" and "Fyndhorn Elves") for users who are unsure of what card they are specifically looking for.

The time complexity order of my algorithms should all be O(n). There is at no point in my data where I have to use a nested for loop to search through a multidimensional array, and each data structure I used prioritized fast access times due to the size of my data. HashMaps access at O(1) speed, meaning that any access that I had to do, if the id of the card was known, would be no detriment to my time complexity. If I search a card without knowing its id, it would

only be O(n) time, as I have to search through each element for a matching name and add the card's key to an ArrayList.

For my program, I used 3 main data structures - HashMaps, ArrayLists, and Arrays. As mentioned previously, I used the HashMap class included with the java.collections library to organize my Card objects after reading them in from my .json file. To hash them, I initially tried to use solely the included unique, alphanumeric id included with each card object from my data source, however quickly realized I would have no feasible way to access my data once put into the map. From there, I went through a few iterations, trying instead to hash based on the name of the card itself but then ran into collision issues, which the Collections HashMap class does not handle. I tried myself to implement buckets using ArrayLists of Card objects but found it to be both a major hassle to code, but also rather inefficient to access my data if needed. Instead, I ended up hashing using an array of Strings as my key, containing the aforementioned unique id along with the name of the card. I found this to be a lot more manageable to deal with, as it allowed me to keep each Card object in a unique place in the HashMap, thus eliminating collisions, but also kept the name of the card intact in case I needed to access the data using only a name. I, then, assemble the buckets after the fact by skimming through the data and collecting each value that matches the name and return the ArrayList that the user can then choose from. I streamlined the process a bit in the deck analysis portion, forgoing the choice for each card and choosing the 0th element in the returned ArrayList.

An opportunity I encountered when developing this program was to learn how to use an outside code library in my own work. When sourcing my data, I initially intended to use a file from a different source, since it was the only way I could source a .csv file which we have been working with all semester. However, after inspecting my data, I found many inconsistencies that

would have made trying to read in my data in any reasonable fashion nearly impossible. It was then that I decided to explore my other option, reading and parsing a .json file. I researched online and found one that seemed good, but was a rather barebones library that hadn't been updated in at least 5 years. After struggling to find useful documentation, and trying to survive on StackOverflow pages, I found someone using the GSON library, which immediately stood out to me for being a lot simpler and a lot more robust than the other library. After converting the start of my code to the new library, I found it much easier to progress and write my code.

An error I encountered in my code was when I encountered a NullPointerException while reading in my data. When a value is not present, the data source file lists the field as null, which often led to errors in my code. When the computer tries to read a value of a specific type, and instead receives null, it throws a NullPointerException and crashes out of the code unless caught. To fix this, I had to research my library and look through the documentation until I found something useful. As it turns out, there's a JsonReader.peek() function, which returns the type of the next value, without actually reading it and throwing the error, which was a quick implementation into my code that immediately fixed all my issues.

If I could change anything for next time, I would have spent more time trying to develop a Graphical User Interface for my code, as a lot of the data skimmed from the file was related to API requests for various images and links to other pages. This would have allowed me to create a very cool interface that actually showed the cards being presented instead of relying solely on the console for output. I would have also put more functionality into the decklist portion of my code, possibly even allowing the user to select cards from the card search option and add them to

a blank decklist, which could be returned to the user at the end of the code.