**Project 4 Write Up: Dolev Peleg**

**Approach, Design, and Algorithm**

I started this project by first designing the most basic class, CourseDBElement, because all the other classes are based on this data class. This was not complicated and I quickly moved to working on CourseDBStructure.

CourseDBStructure was much more challenging to write, especially because I never created my own hash table before. Specifically, it was confusing for me to understand how to coerce the CRN into a string, and how to create a hash code for that string. After going back to the notes from the hashing PowerPoint and lecture, I saw that by using the algorithm: key = G \* key + s.charAt(counter), I can get a hash code from any string. After that, I decided to use this algorithm in a separate method that I created, calculateHashCode. This is because this algorithm is needed both in the add() and the get() methods.

Another challenge that I faced during the creation of CourseDBStructure was understanding what 4k + 3 prime is, and how to use it. After going to office hours with my instructor, he explained to me how to use this, and why it was created.

After I finished working on CourseDBStructure, I made sure to run the provided JUnit test. This is because during my previous project, I postponed testing until the very end, which caused me to waste a lot of time debugging. This project, I decided to test after each class creation. This was a good decision, because not a lot of debugging was needed later on due to my testing.

CourseDBManager was not too complicated to create, due to it using mostly the same methods as CourseDBStructure. First I was not sure how to call the CourseDBStructure’s methods, but after confirming with my instructor, I understood that I had to create a CourseDBStructure data field, and set a default value for its table size in its default constructor.

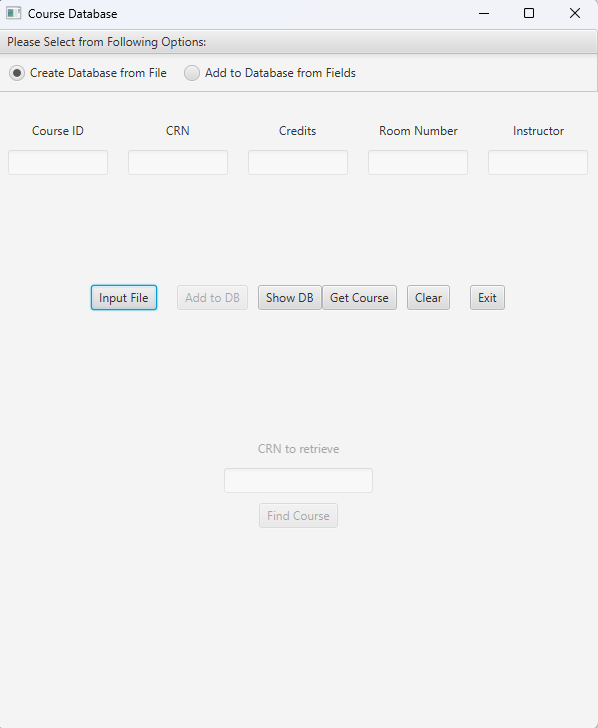
Most of my effort while working on this class was spent on creating the readFile method. While I already know how to read from text files, it was challenging to understand how to ignore invalid input. I took my time, and created a significantly large “if” block, which covered all the requirements for a valid input. This took a lot of testing and debugging work, but eventually I figured it out.

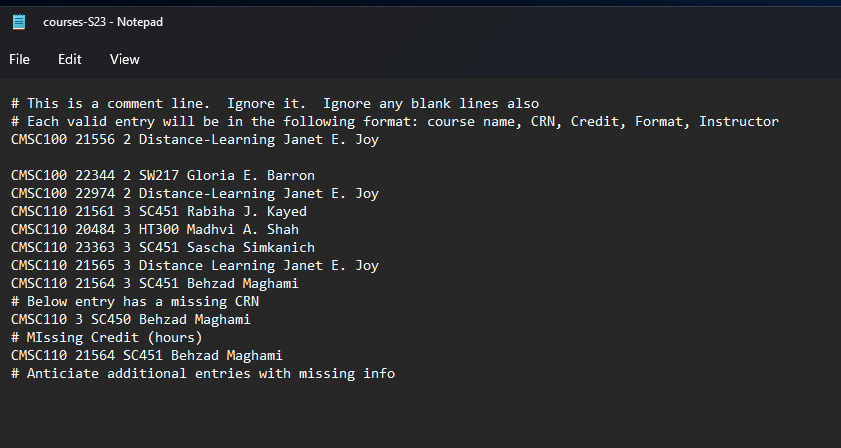
After finishing BasicDoubleLinkedList and its inner classes, I started working on SortedDoubleLinkedList. This class was more complicated to design due to the add() method having many if-else blocks, and nested if-else blocks.

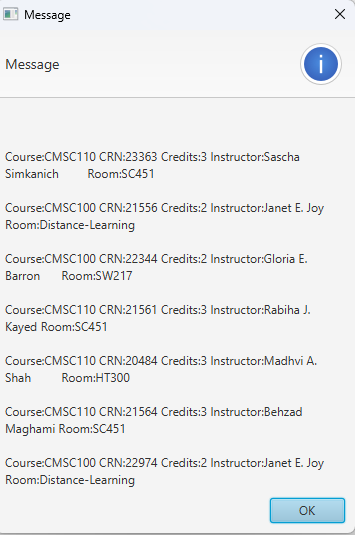
Lastly, after passing all the JUnit tests that were provided to me, I created my own test class. My main effort was to assure that my readFile() method was working correctly. I tried to write two invalid inputs into the file, and see if they will be added to my course structure. All the tests were successful.

**Test Runs and Cases**

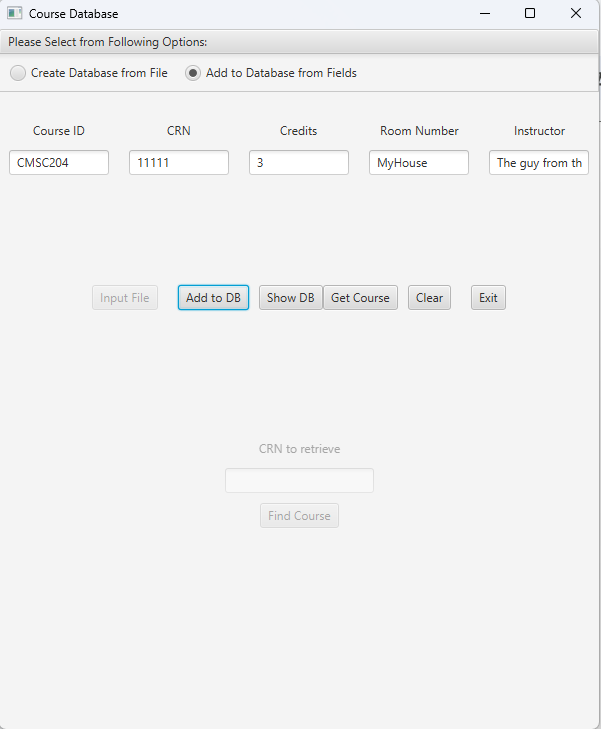
**Creating a CourseDBStructure from a file**

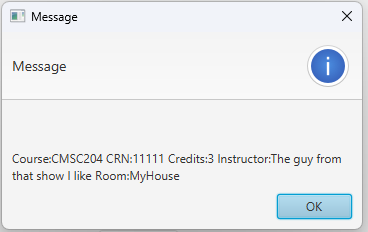
****

****

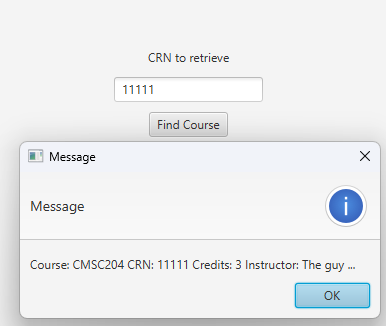
****

**Creating a CDE from text fields**

****

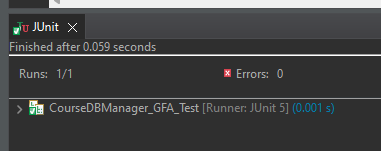
****

**Getting a CourseDBE by using its CRN**

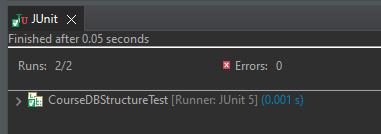
****

**JUnit Testing**

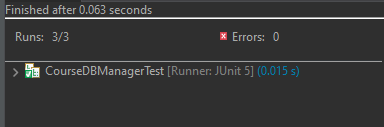
**CourseDBManager\_GFA\_Test**

****

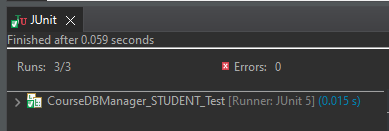
**CourseDBStructureTest**

****

**CourseDBManagerTest**

****

**CourseDBManager\_STUDENT\_Test**

****

**Learning Experiences**

The main lessons that I haveI learned were how to create my own hash table, how to create a hashcode for strings, and how to validate strings’ format.

One thing that I will do in my next projects is feel more comfortable when using hashing, and hash codes. When I first learned this concept, it was confusing and I did not understand why it is better than regular lists. Now I understand that the ease of which one can get items from a hashtable without having to search for them, or iterate through the entire data structure, can be very useful.

All the parts of my project were successful, and besides my notes and the textbook, I used two more websites which will be cited below.

**Assumptions**

1. The user will be using JUnit 5, Java, and JavaFX
2. Valid input in text files will follow the same format that was in the text file that was provided to me.

**Enhancements**

No enhancements were made.

**Citations**

Sameek MishraSameek Mishra 9, aioobeaioobe 410k112112 gold badges808808 silver badges825825 bronze badges, RaviRavi 30.5k4141 gold badges119119 silver badges170170 bronze badges, AndyAndy 4, AbayAbay 7111 silver badge33 bronze badges, Gherbi HichamGherbi Hicham 2, AndyAndy 18911 silver badge22 bronze badges, AbuNassarAbuNassar 1, denolkdenolk 7661414 silver badges2727 bronze badges, Soudipta DuttaSoudipta Dutta 1, Felix A Marrero PentónFelix A Marrero Pentón 21322 silver badges1010 bronze badges, PhilippePhilippe 48511 gold badge77 silver badges1818 bronze badges, sabujpsabujp 98911 gold badge1111 silver badges1111 bronze badges, &amp; Karthik GunasekaranKarthik Gunasekaran 122 bronze badges. (1958, June 1). How to convert a string into an arraylist? Stack Overflow. Retrieved April 3, 2023, from https://stackoverflow.com/questions/7347856/how-to-convert-a-string-into-an-arraylist

YouTube. (2019, April 2). Hashing - hash codes. YouTube. Retrieved April 3, 2023, from https://www.youtube.com/watch?v=bwiYI3S\_iHs