John Mobarry

CMSC 204

Assignment 4 Reflection

The implementation of Assignment 4 was a relatively straightforward exercise to help the class understand how to implement a linked list bucket hash table. When presented with the task of implementing this assignment I was anxious regarding creating an inner class for the linked list or creating a node class. Reflecting on what I had learned last semester and this semester I remembered that there are packages that can be imported in order to achieve this data structure more simply. This assignment also made me reflect on how important it is to maintain consistent syntax throughout your classes or you will experience critical failures preventing your code from running correctly.

I initially struggled attempting to start this assignment prior to completing my UML diagram and therefore did not approach the assignment in an organized manner. Due to this I attempted to complete the CourseDBStructure class first as I assumed that the structure of the database would be an essential starting point. This was a foolish errand as I was not able to design the Structure efficiently as I had not actualized the CourseDBElement and therefore was ignorant of its relevant properties. After spending a few hours making no headway into the assignment I decided to step back and approach the UML diagram in order to visualize the three required classes: CourseDBElement, CourseDBStructure, and CourseDBManager.

When considering the CourseDBElement class I had to first think about what were the inherent qualities that needed to be defined prior to creating the constructors and the methods. Each CourseDBElement will need three Strings: CourseID, roomNum, and instructorName and two int: CRN and credits. The most important attribute of the CourseDBElement class is the integer CRN as it will be utilized in determining the hashcode for the specific CourseDBElement which is the basis of the entire assignment. If the hashCode(): method does not function appropriately, then the entire structure of the assignment will cease to function appropriately. I experienced issues regarding the same as when I created my CourseDBManager class. I erroneously utilized the inherited hashcode() method instead of the hashCode(): method I created inside of my class. This resulted in the code simply taking the hashcode of the CRN integer instead of following the hashcode algorithm that was specified within the assignment. In the assignment it specifies that you will need to convert the CRN to a string then complete the hashcode of the same. Once I corrected the syntax and utilized the correct hashCode() method my JUnit tests resulted in passes.

When approaching implementing the CourseDBStructure I had to consider how I was going to implement the LinkedList hashcode buckets. I initially considered making a subclass within the CourseDBStructure where I defined a linked list and then created an array of the same. After reflecting on assignments and labs completed over the last year I remembered that there is a package that you can import that allows the programmer to utilize the linked list data structure without having to create it themselves. By utilizing import java.util.LinkedList I was able to easily import the class which can be used in an array manner and focus on the specifics of the assignment with the confidence that my linked list will function properly. Although it was not specified in the assignment directly I came to the conclusion that a method would need to be implemented within the CourseDBStructure class to assist with the CourseDBManager implementation of the showAll(): method. In order to simplify the showAll(): method I created a nested for loop method which would create string representations of the elements that are present so that this method can simply be called in the CourseDBManager.

Finally I had to implement the CourseDBManager class, which was definitely the easiest to implement class once I had a functioning CourseDBElement and CourseDBStructure class. There was much less creativity required when creating this class as there were no additional methods required besides those outlined in the interface to be implemented. Due to efficient design within CourseDBElement and CourseDBStructure the four required methods could be completed efficiently by calling the methods of the other classes, resulting in clean and succinct code. The readFile() method called back to my experiences in CMSC 203 and I had to search the internet for a good way to parse the data that was being read in the file. The implementation of the bifurcated split method I was able to parse the elements based on spaces up to a limit of 5 items for the element. This is definitely a method that I will utilize in the future as it sets a limit on what your read file can take in and prevents unneeded errors.

Overall this assignment helped me remember that some approaches to design are much more efficient than others. I initially attempted to complete this assignment in a top down manner and it resulted in a lot of needless stress. I found that the most efficient way to complete the assignment was to consider the UML diagram in a top down manner and flesh it out prior to engaging with the IDE. When I moved towards programming in the IDE I then attempted to code from the ground up going from CourseDBElement → CourseDBStructure → CourseDBManager as each higher class requires the implementation of the lower class.