3/13/16

Created linked list classes for Rooms and Classes. Currently contain some of the more basic variables needed for comparisons and sorting which will be expanded later. Both lists currently have a makeNode method and a sortedInsert method. The sortedInsert methods currently sort by roomNum or sessionNum, and will later change to sorting by attributeNum.

3/14/16

Added Traverse methods to both linked list classes. This revealed that although all nodes were successfully being added to the list, the lists would become fragmented (when ten values were inserted only four values were shown in traverse. Troubleshooting showed that the problem was shared in both the basic add function (adds a node to the end of the list) and the sortedInsert methods. I was not fix the problem at this time.

3/16/16

Continued troubleshooting the linked lists. After many different attempts to make the insert methods function properly I determined that using a linked list may not be the way to store the data. Research on how JavaScript handles arrays showed that it allowed for an array’s size to be changed dynamically. At this point I have decided to change from using a linked list to store data in the program to using an array, inserting all the values into the array then sorting the array. Currently a selection sort has been implemented, but this may be changed to a more efficient sort later on.

4/1/16

Previously, the array of classes was sorted only by total availability of the classes. The agreed upon priority of classes is lowest availability and highest number of attributes, this calls for a secondary sort of the array. At first, I tried sorting sections of the list, where the program would walk through the classes (that have been sorted for availability already) and if several classes in a row had the same availability, it would sort that subset of the array based on the number of attributes. Several attempts were made to get this method of sorting to work, but none of these attempts were successful, and some even caused data corruption (total availability values were being changed when they are not supposed to). I needed a new approach.

Instead of sorting subsets of data, I needed to sort the whole array a second time. However, my current sorting algorithm, the selection sort, cannot be used as a secondary sort, since it is an “unstable” sort and would override the results of the primary sort. An insertion sort on the other hand, is a “stable” sort, and is valid for a secondary sort. I implemented the selection sort for number of attributes to be the primary sort, and then implemented the insertion sort for total availability as the secondary sort. This gave the desired results.

**Note:** if the sorting algorithms are updated to more efficient sorting methods, they must have the same stability of the algorithms being replaced or the data will no longer be sorted correctly.

4/7/16

Fixed an error with sorting classes where the last item in the list was not sorted based on availability correctly. Added a selection sort for the Room array that sorts them from the lowest number of attributes to the highest. Added several more comments meant to help differentiate which methods are meant to work together ex: Class object, class sorting and add class are all in the same section.

4/13/16

Began work on the scheduling algorithm, including functions like:

* checkAttributes function which checks if a room has the appropriate attributes to hold a class
* timeToIndex function which converts army time (how availability is stored in the class object) into an index number that can be used to traverse a room’s schedule’s 2d array
* beginning work on the classLength function which determines how long a class session is
* beginning work on the ScheduleClasses function, setting up conditions for the outermost while loops and some if statements.

Also added more comments to the code as a whole to make it more readable in the future when editing or debugging the program

4/14/16

Continued work on and completed the schedule algorithm. Added some more test code to see if the schedule algorithm was working and found out that there were some syntax errors which I was able to fix as well as logic errors that will needed to be fixed later. So far, when the schedule algorithm is called it only “schedules” the first class, and it only schedules it on Monday even though the class was specified to be held Tuesdays and Thursdays.

4/15/16

Finished fixing bugs in the schedule algorithm. Fixing the bug where only Monday was scheduled went somewhat quickly but uncovered a new bug where the program thought each class was being held on all 6 days despite being declared as being held on Tuesdays and Thursdays. Once that issue was fixed it was found that when a class was being scheduled at the end of the day, if the last time block was open but there was not enough time for the full class the program would make 2 more time blocks, this was fixed by adding a check at the start of a while loop to see if the current time index plus the class length would be later than the last time block. When this bug was fixed the program began an infinite loop. If the program couldn’t schedule every class it would go into an infinite loop as courseCounter was not being incremented unless the class and the room were not compatible, meaning if a compatible class did not fit into the room it would not move on to the next class and continually try to schedule the same class in the same room, making no progress. Once I fixed that bug the scheduling algorithm appeared to run the way it is expected to run. A few changes were also made to the testing code to make sure that the algorithm was functioning properly.

4/16/16

Fixed a few small bugs involving when the classCounter gets incremented.

4/18/16

Collaborated with Cole to allow the program to schedule classes into individual buildings, as well as make the checkAttributes method more efficient. The program can now schedule into 4 different buildings, Tech, Admin, SSB and Humanities.

4/23/16

Began stress testing the program by feeding it 1000 random classes and 100 different rooms and have found that the program takes anywhere from 0.6 seconds to 0.2 seconds to create a schedule depending on the classes and rooms it receives. During this time I was also able to find and fix several bugs

4/25/16

Made several additions to the program by the request of Kevin, including saving the time and room that a class was scheduled into, into the class object named roomScheduled, timeScheduledStart and timeScheduledEnd, as well as a new method called findScheduledClass which takes a string called courseID and searches the courseIDs of all the classes in the scheduledClassList. If it finds the class it will return all the attributes of the class as a single large string with a pipe(|) separating each attribute. If it cannot find the class it will return “Class not found”

4/27/16

Added the ability for more class subjects and assigned them to their proper buildings. Added a Schedule method which will call the scheduleClasses methods with the appropriate arguments automatically, making it easier to implement since the Schedule method takes no arguments. I also removed all the test code so that it will not be run when utilized by the UI. Added another attribute to rooms showing how many classes a room is currently holding.

**Andrew Westhoff**

**Change Log**

**A.C.K.**

**123 Main Street**

**St. Charles, MO 63376**

**4/28/2016**