Assign6Tests  
Jennifer Swofford

Implementation:

1. I started with a searchRow search that returns the index of the row located (or -1 if it’s not located). This is defined in the CityTable and in the StadiumTable. I made searchRow an abstract method in the AbstractTable class. I may refactor this to be completely to be a concrete method in the AbstractTable class depending on if time allows.
2. From there I decided that I needed to change findRow so that it leverages the searchRow and displays the row contents to the user. That made me realize what I would rather do is put a concrete displayRow method in AbstratctTable that takes an int (passed from SearchRow) and displays the contents of the row in a user-friendly format.
3. At this point, I removed the bulk of what I had in remove row and leveraged the newly created searchRow. Interestingly, I did find a bug in my program at this point. It turns out that when I removed a row and then saved it to a file, I wasn’t updating row count correctly. This is where I ended up removing the rowCount and started using fullTable.size().
4. The bubblesort was next – this took quite a bit of time in the CityTable. I encountered hiccups around needing to do some casting, and also had been running into issues where I wasn’t hitting the compareTo in my CityRow class. It kept using compareTo from the string class. It turned out I wasn’t passing the object but was passing compRow1.getCityId(). Once I corrected this, I didn’t have any issues.
5. From the sort, I jumped straight to the join. I found this part to be the most daunting, so I spent a lot of time here. I also wasn’t sure where to execute it in the main program. I ended up creating a new menu option for the merging of the tables that outputs a JoinedTables.txt.

I really had a tough time with this one around the types and casting I had to do. I ended up reaching out to tutor for assistance here because I couldn’t grasp how to make the for each loops work.