Katie Macalintal

Programming Assignment 1: Assessment

Harm/Benefit Analysis: I feel like this closest school algorithm could cause more harm than good. I think that the ones who would benefit the most from this algorithm would be those at the schools that students will be consolidating at. Since these schools will be gaining more students, they may receive more funding and money from the town. However, this sudden change in the size of students may also be overwhelming for the school to handle. Those who are most likely to be harmed are those that have their schools shut down. Families may not only be stressed about logistics on how to get to the school but also adjusting to a new atmosphere and many teachers may be left unemployed. I would only be willing to implement an algorithm like this if these areas of concern are addressed.

Correctness Assessment: The code compiles and runs without errors. Through test cases, I was able to verify that my code correctly found the two closest School objects when the input had an even and odd number of Schools, the two closest School objects were in the same half and when they weren't, when there were multiple school objects with the same x-coordinate or y-coordinate, and when Schools have the same x value as the x midline. To handle when School objects have the same x value as the x midline, I created a new attribute in the School object to track which side that School object was sorted on.

Modularity/Extensibility/Flexibility: In my code, I tried to break down certain parts of the algorithm into methods, such as finding the distance between two Schools, our brute force approach, and getting the yDelta array of points. Creating a separate method for finding the distance between two schools was really useful as it allowed for easy reusability and better organized my code. In the combine part of the algorithm, there are a handful of repeated lines, however, with the number of variables it had to return and modify in the calling function, I felt as if it was better to just have the code repeat rather than place it in another function. I am currently not aware of anything that I hard-coded that may present challenges if my input were modified or additional functionality was required.

Readability: In my code, I've added a handful of comments around my methods, loops, and conditionals. I also added comments to break up where the brute force, divide and combine steps of my algorithm take place. I stuck with the variable names that we used in our pseudo-code in class as I found them fairly descriptive of their purpose, but I also

added comments around their declaration just in case it was still ambiguous. I think that my method names are also fairly descriptive and inform the reader of their purpose.

Effort: Overall, I spent a few hours working on this code. Looking back on my approach to this project, I think that starting early and breaking the algorithm up into parts allowed me to adequately implement and debug the algorithm. Throughout this project, I would implement and test each section before moving on to the next part of the algorithm. When getting stuck, I would first insert print statements where errors were getting thrown. If this wasn't helping me then I would turn to pen and paper and draw out what the code was doing. Throughout the project, I also referred to the Java API and would ask Maja and Professor Kimmel for some advice on how to approach certain parts of the algorithm. Sometimes I would share the knowledge gained from those conversations with other students in the class as well. If I could go back and do it again, I would consider the edge cases earlier on in the testing process.