Anthony Uitz aru5mf

1. The ArrayList class that I used follows from List ADT.
2. I used an ArrayList to hold the points, which includes both coordinates and label and distance to input point, as well as to sort them based on distance using the Comparable interface coupled with the Collections built in sorting method.
3. I found two problems with the program statement. The first involves how sanitized the inputs are. In order to make the program work, I am assuming that k and M are greater than 0, M is less than the number of values in the data file, and k is less than or equal to M. The second assumption I made was in the case of equal number of category 1 points and category 2 points, the input coordinate would be categorized as the category 2. Another minor assumption I made was that the input to end the program could be any coordinate with both x and y equal to 1, not specifically “1.0 1.0”.
4. There are three program components: reading in input, calculating categories and output, and printing output. The first component uses two Scanner objects to read in the user input and the data file. The M values needed from the data file are stored as Point objects, which are added to an ArrayList. The second component is in a while loop that takes in the two user-inputted coordinates and the ArrayList of Point objects to calculate the distance to each point, which is stored in the Point objects, then sorts the ArrayList based on the distances. Lastly, it calculates the average distances and the category of the user-inputted coordinate. Finally, the last component, which is also in the while loop, prints the sorted Point objects, the category assignment, and the average distances.
5. Due to my assumptions about input sanitation I did very little error checking. However, I could have checked whether or not the inputs were valid, like the file existing or k being less than M. I could have also checked if k was 0 and 1 and changed how the output was printed based on this as if k = 1, there is no category 2.
6. Three possible test cases would be checking when category 1 is dominant, checking when category 2 is dominant, and checking when M is less than the number of values in the data file. The examples given in the homework assignment, with k = 3 and k = 5 respectively, are the first two test cases. The last test case could be done by using the five points given in the homework example as the data file, but setting M = 4 or any other number less than 5 and greater than 0.
7. I spent about half an hour to an hour coding, two to two and a half hours total including understanding the problem and writing the report.