Antonio Scalfaro

Project 2 – Maximal Points

CMSC 315

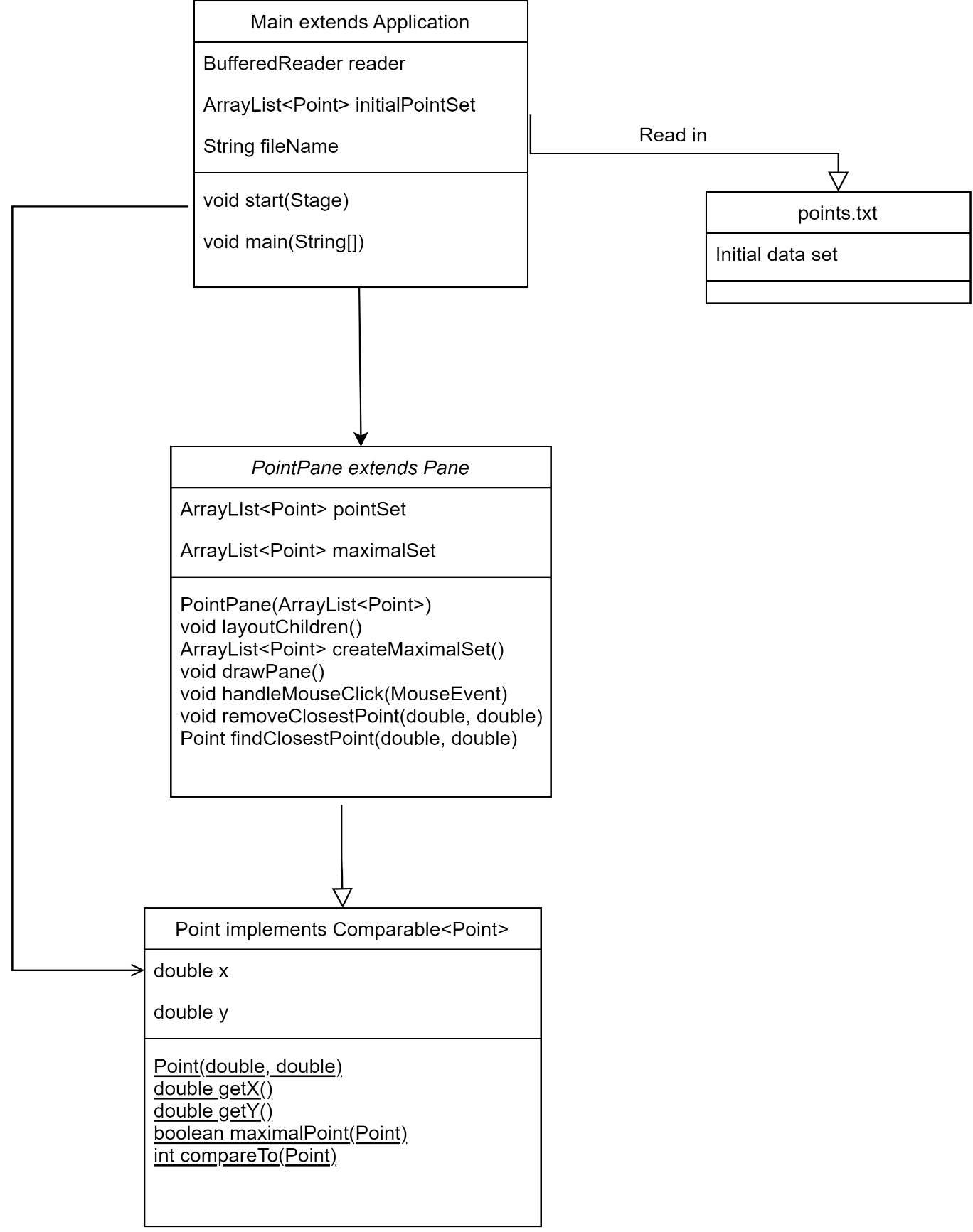
02/13/2024

This is the Maximal Points Program which consists of three classes and one data file. The three classes are Main which extends Application, PointPane which extends Pane, and Point which implements Comparable. This program will produce a GUI with the initial data set of coordinates provided by the points.txt file that will create and display these Points with a line drawn connecting all the maximal points in the set. From there, the user can left click anywhere in the GUI to add another Point and the maximal line will readjust as new Points appear. If the user right clicks on the GUI, the nearest point to that right click will be removed, again the maximal line will readjust as the point set changes. As long as at least one Point is displayed in the GUI, the maximal line will be displayed. As far as time complexity in terms of Big O is concerned, when Main reads in the initial data it is O(n) where n = number of Points. The PointPane class creates each Point as a Circle at a O(n) rate, creates the maximal set of points at O(n2) rate, and lastly must iterate through that list at a O(m) rate where m = number of maximal points which gives the PointPane class an O(n2 + m) rate. Therefor the overall program has an O(n2 + m) rate.

**Test Plan:**

As far as testing goes for this program, it was mostly conducted by using the GUI which does not allow for hard set tests. I tested by running the program, ensuring that all the initial points in the initial dataset were present and that the maximal line was correctly created by connecting the left pane of the GUI to the left most maximal point, connecting all other maximal points together via a PolyLine, and then connecting the right most maximal point to the bottom pane of the GUI. After ensuring all initial points are created and the maximal line was correct, I tested the mouse event functionality of the program. I added multiple points, the first would be a point that does not change the maximal point set and the second would be a point that does change the maximal point set and therefore changes the maximal line. I then removed points, I tested this by keeping the mouse in one spot of the GUI and right clicking. I left the mouse in one spot to test that the removed point would be the next closest point to the click location. I tested this until there were no points left in the GUI. The program passed all tests.

**Class Diagram:**

****

**Initial Data Set:**

A screen shot of a computer

Description automatically generated

**Add Point, maximal line unchanged:**

A screen shot of a computer

Description automatically generated

**Add Point, maximal line changed:**

A screen shot of a computer

Description automatically generated

**Remove Point, maximal line unchanged:**

A screen shot of a computer

Description automatically generated

**Remove all but one Point:**

A screen shot of a computer

Description automatically generated

**Remove all Points:**

A computer screen with a white square on it

Description automatically generated

**What was Learned?**

This program was a really fun and challenging program to create. It gave me an opportunity to grow my knowledge of JavaFx and strengthen my skills in algorithms and file reading. I found the most challenging part to be creating the maximal line which needed to be connected to two edges of the GUI (left and bottom) and all maximal points in the GUI. It led to great frustration when I ran into the problem of the original data set creating a maximal line that was connected nearly perfectly but the last point would connect to the top edge of the GUI instead of the bottom edge. I found that when adding another point this would immediately correct so that gave me the suggestion that the issue lie in the getHeight() call in the maximal line creation. I found it was giving the wrong value because it was not redrawing the line when the layout changed from the original GUI pane to the PointPane. To correct this I had to learn more about the layoutChildren class method and add a protected version in my code that would allow the correct value of getHeight() to be returned so the original maximal line was correct. I also found myself thinking of the time complexity of this program, wondering if there was a good way to reduce it to an O(n2) rate. I did not end up trying to refactor the code to attain this, but I suspect there would be a way to create the maximal line while identifying maximal points which would eliminate the O(m) rate that gets added by having to iterate through the maximal set.