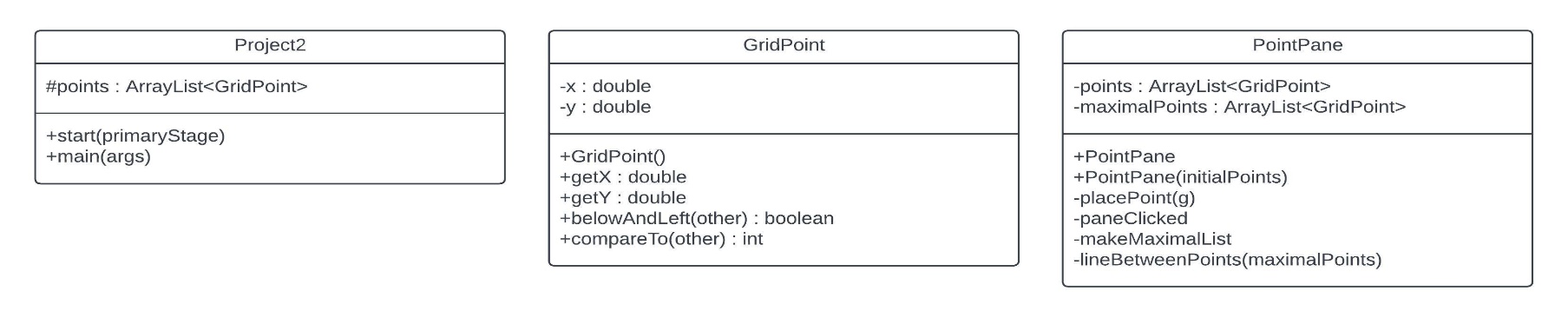
CMSC 315

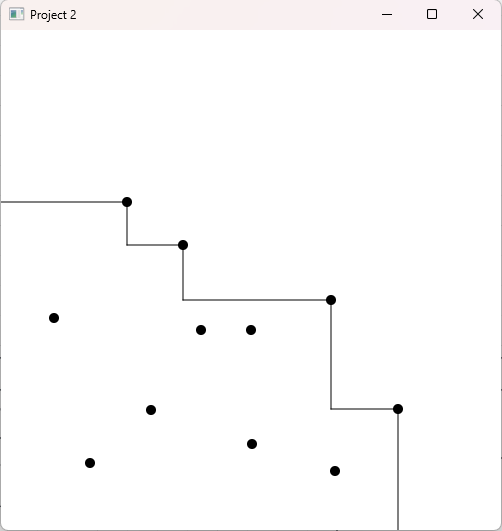
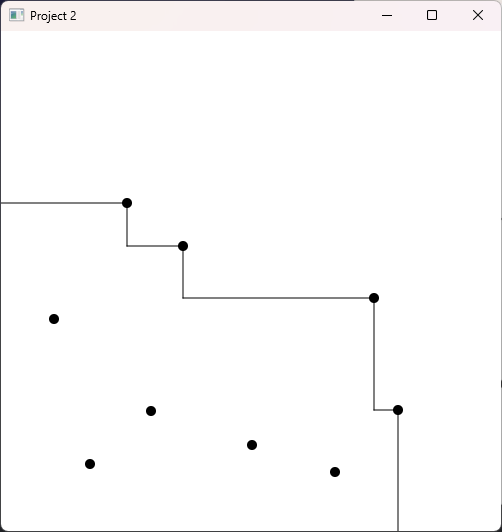
Programming Project 2 – Maximal Points

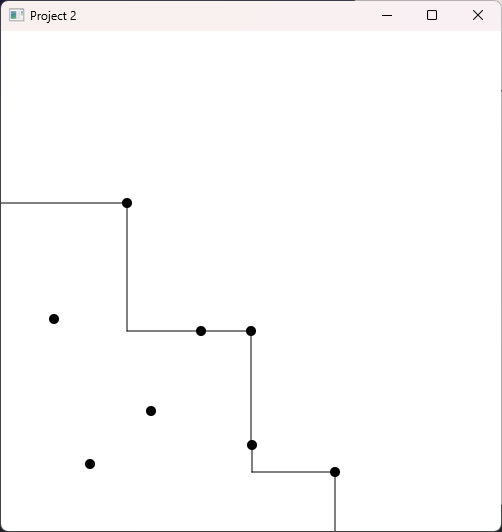
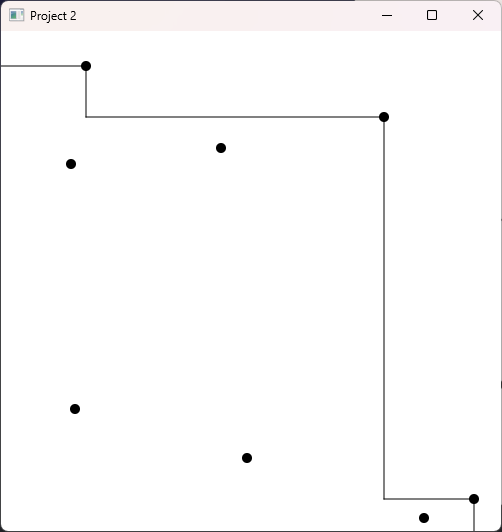
Daniel Smolsky

**UML Diagram:**

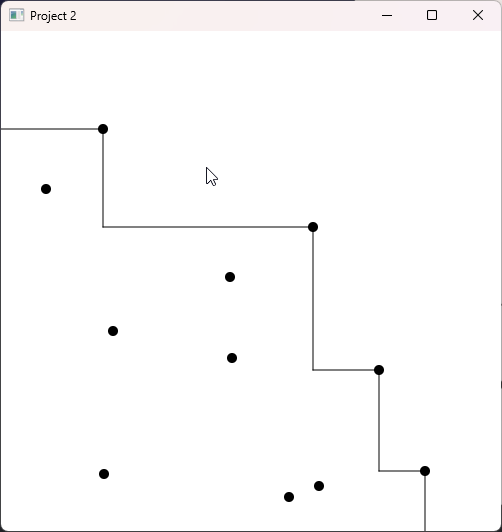
**Test Plan:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test # | Purpose | Pos/Neg Test | Input Values | Expected Result | Pass/Fail |
| 1 | Check program output matches example | Positive | 200.0 300.0  250.0 300.0  330.0 270.0  150.0 380.0  126.0 172.0  397.0 379.0  334.0 441.0  53.0 288.0  89.0 433.0  182.0 215.0  251.0 414.0 | Same image as in guidelines | pass |
| 2 | Left click on pane adds points and right click removes points | positive | 200.0 300.0  250.0 300.0  330.0 270.0  150.0 380.0  126.0 172.0  397.0 379.0  334.0 441.0  53.0 288.0  89.0 433.0  182.0 215.0  251.0 414.0 | Similar image to guidelines with some new points added in by left click and some old points removed | pass |
| 3 | each new point redraws the line between maximal points | Positive | All original values removed with right click and several new points added at left click location | New maximal point line drawn | Pass |
| 4 | Removing any points redraws the line between maximal points | Positive | Original values with some points removed with right click | New maximal point line drawn | pass |
| 5 | Confirm maximal points do not have any points above and to the right | Positive | Randomly clicked points on pane | Maximal points do not have any points above and to the right | pass |

Test plan 1: Test plan 2:

Test plan 3: Test plan 4:

Test plan 5:



**Big-O analysis:**

The entire process to determine maximal points is within the makeMaximalList method in the PointPane class. Therefore, this is the only method I will analyze to determine the time complexity of finding the set of maximal points. Within the makeMaximalList method there are steps to add “corner” or “fake” points to the list, these points assist in drawing lines connecting the maximal points in another method and don’t add to the time complexity or represent real maximal points. The process to determine maximal points is 3 steps:

1. Sort the list of n points by x axis reverse order, Collections sort uses a version of merge sort which is O(nlogn) time.
2. Add the first value from the list (highest x value) to the maximal point list, O(1).
3. Iterate through the list of n points once, if the current point has an equal X value and greater Y value than the most recent point in maximal list, replace the most recent point in maximal list with the current point. Otherwise, if the current point has a greater Y value, add it to the maximal list. O(n)

In total, the time required to complete these steps is O(nlogn) + O(1) + O(n) = O(nlogn).

**Lessons Learned:**

Creating this project, I practiced and improved my inheritance and polymorphism skills with my extension of the Pane class and implementing comparable interfaces. More specifically, I learned how to use JavaFX to create interactive graphical user interfaces and handling mouse events like clicks for adding and removing points in PointPane. Additionally, the reading this week introduced concepts for tracking algorithm efficiency such as big-O analysis. Because of this, I was able to improve my initial O(n2) implementation to a much more efficient O(nlogn).