Programming Exam 1



Instructions. You will have 50 minutes to create and submit one program and a readme file. Debug your code as needed. You may use your book, your notes, your code from programming assignments, the code on the COS 126 course website, the booksite, and you may read Piazza. No form of communication is permitted (e.g., talking, email, texting, calling, posting to Piazza) during the exam.

Submissions. Submit your work using the link on the Meetings page.

NAME:

Grading. Your program will be graded on correctness, design, efficiency, and clarity including comments. You will receive partial credit for a program that correctly implements some of the required functionality. You will lose a large number of points if your program does not compile.

Discussing this exam. Due to travel for extracurriculars and sports, some of your peers will take this exam next week. Do not discuss exam contents with anyone who has not taken the exam.

This paper. In addition to submitting your code electronically, you must return this paper. Fill in the information below, then transcribe and sign the Honor Code pledge. You may do so now.

	NETID:		
	PRECEPT:		
	EXAM ROOM:		
"I pledge my	honor that I will	not violate the Honor Code during th	is examination."
SIGNATURE:			

Write a program, DrawPath.java, that draws line segments in the standard drawing window based on data read from standard input. Start with the provided template, found on the Meetings page:

```
public class DrawPath {
   public static void main(String[] args) {

      // starting point
      double x = StdIn.readDouble();
      double y = StdIn.readDouble();

      // YOUR CODE HERE

}
```

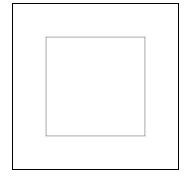
The input to this program consists of pairs of values, where the first pair is the starting point (x, y), and all subsequent pairs are displacements, (dx, dy). You may assume that all input files will have at least four values and have an even number of values (i.e., for every dx, there will be a dy). For example, using the inputs in pathA.txt as listed in the figure on the right, the starting point is (.2, .2) and the first (dx, dy) is (.6, 0), so the first line segment will connect (.2, .2) to (.8, .2), then (.8, .2) to (.8, .8) and so on.

pathA.txt

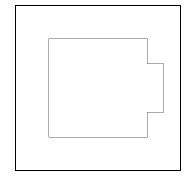
.2 .6	.2
0	.6
6	0
0	6

Under "YOUR CODE HERE", write a single loop that reads double values dx and dy from StdIn, uses StdDraw.line() to draw a line segment from (x, y) to (x + dx, y + dy) and then updates x to x + dx and y to y + dy. Do not use StdDraw.polygon(). Note, use of unnecessary data structures will be penalized.

Your code should produce the following drawings for the input files pathA.txt and pathB.txt:



> java-introcs DrawPath < pathA.txt</pre>



> java-introcs DrawPath < pathB.txt</pre>

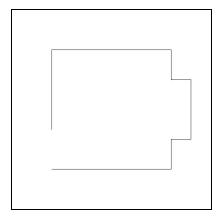
Add code to DrawPath. java to perform the following validity checks:

- Is the path closed (i.e., does it end where it started)?
- Does the path stay within the bounds of the standard drawing window?

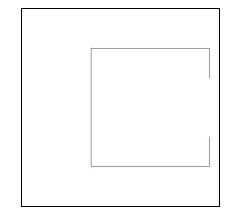
First, test whether the path is closed by checking if the last x and y coordinates are no more than .00001 away from the first x and y coordinates (i.e., compare the last x to the first x and the last y to the first y). If that is not the case, print "Not closed". This test does not affect drawing output.

Second, make sure each coordinate you send to StdDraw.line() is within the bounds of the drawing window. If the ith line to be drawn would cause this condition to be violated, do not draw the offending segment; instead, print the error message "Segment i out of bounds". Do this test for each segment. Your program should still draw all line segments where both points are in bounds.

Your code should produce the following output for the input files pathC.txt and pathD.txt:



> java-introcs DrawPath < pathC.txt
Not closed</pre>



> java-introcs DrawPath < pathD.txt
Segment 3 out of bounds
Segment 4 out of bounds
Segment 5 out of bounds</pre>

Part 3 readme.txt 1 point

In truth, another validity test is needed: Does the path intersect itself? In 50 words or less, describe how you would solve this problem. Write your answer in the provided readme.txt template.