Project 4 (Java): Morphology. Implementation of the four basic Morphology Operations (dilation, erosion, opening, and closing) and its applications.

Your program will take an input image, a structuring element and a choice of morphological operation via commend-line (args []). Choice of operation are as follows:

1 for dilation
2 for erosion
3 for opening
4 for closing
5 for all four operations.

Summery of what you need to do:

1) Implement your program according to the specs given below. Debug your program until it passes compilation.
2) You will be given three image files: data1, data2, and data3 and three structuring elements, elm1, elm2, and elm3. You also will be given an answer key, data1_answer to verify the correctness of your program.

*** Please pay attention to the three image files and the structuring elements.

3) You will run your program multiple time as follows:

*** 1st run of your program:

4) Run your program using data1, elm1, and choice 5 (all 4 operations); debug your program until your program produces the same results of dilation, erosion, opening and closing as those shown in data1_answer.

*** If your answer is correct, you will receive 8 out of 12 points of this project. Save all the files produced by the 1st run in a separate sub-folder (or re-name them) prior to the next run; so they won't be over-written by the sub-sequence runs, since your program will use the same file names in the sub-sequence runs; and you need these files for your hard-copy (pdf file).

*** 2nd run of your program:

5) Run your working program using data2, elm2 and choice of 3 (opening). data2 contains some large blob-shape objects and some random noise (peppers) in the background. There are also some small holes (salts) inside those larger blobs.
*** your 2nd run will produce a file, openingOutFile. Make observation and compare the openingOutFile with data2. You should see the disappearance of peppers in the background, but the holes (salts) remain within the objects. If you don't see that, you may need to design you own elm2.

*** 3rd run of your programs as follows:

6) Run your working program using the openingOutFile from your 2nd run with elm2 and choice of 4 (closing).
*** your 3rd run will produce a file, closingOutFile. Make observation and compare the closingOutFile with data2. You should see the disappearance of most of the holes within the objects.

*** after your 3rd run, you need to save the openingOutFile (from 2nd run) and closingOutFile (from 3rd run) in a separate sub-folder (or re-name them), so they won't be over-written by the sub-sequence runs, since your program will use the same file names in the sub-sequence runs. ; and you need these files for your hard-copy (pdf file).

*** 4th run of your programs as follows:

7) Run your working program using data3 with elm3 and choice of 4 (closing). data3 contains 3 or 4 clusters of points.
*** your 4th run will produce a file, closingOutFile. Make observation and compare closingOutFile with data3. You should see the merging of points in those clusters. If you don't see that, you may need to design you own elm3.

*** 5th run of your programs as follows:

8) Run your working program using the closingOutFile from your 4th run with elm3 and choice of 3 (opening).
*** your 5th run will produce a file, closingOutFile. Make observation and compare the closingOutFile with data2.

Include in your hard copy (pdf file):
- cover page
- source code
- print dilationOutFile, erosionOutFile, openingOutFile, closingOutFile, prettyPrintFile from 1st run // with captions
- print openingOutFile and prettyPrintFile of 2nd run // with captions
- print closingOutFile and prettyPrintFile of 3rd run // with captions
- print closingOutFile and prettyPrintFile of 4th run // with captions
- print openingOutFile and prettyPrintFile of 5th run // with captions

**********

Language: Java

**********

Project Name: Morphology
Project points: 12pts
Due Date: Soft copy (*.zip) and hard copies (*.pdf):
+1 (13/12 pts): early submission, 10/9/2024, Wednesday before midnight
(12/12 pts): on time, 10/12/2024 Saturday before midnight
(-12/12 pts): non-submission, 10/12/2024 Saturday after midnight

*** Name your soft copy and hard copy files using the naming convention given in Project Submission Requirements. (-2 if not.)

*** All on-line submission MUST include Soft copy (*.zip) and hard copy (*.pdf) in the same email attachments with correct email subject as below; otherwise, your submission will be rejected.

Email subject: (CV) first name last name <Project 4: Morphology (Java)>

*** Inside the email body include your answer to the 4 questions. Optional screen recording if you wish.

========== Specs ========

**********

I. Inputs:

a) inFile (args [0]): a txt file representing a binary image with header.
b) structFile (args [1]): a txt file representing a binary image of a structuring element with header and the origin.
The format of the structuring element is as follows:
1st text line is the header; 2nd text line is the position (w.r.t. index from 0) of the origin of the structuring element then follows by the rows and column of the structuring element.

For example:

5 5 0 1 // 5 rows, 5 columns, min is 0, max is 1.
2 2 // origin is at row index 2 and column index 2 ( the center pixel, in bald and underscore.)
0 0 1 0 0
0 1 1 1 0
1 1 **1** 1 1
0 1 1 1 0
0 0 1 0 0

** Note: when a structure element has zeros, only those 1's uses in dilation and the matching in the erosion!

Another example:

3 3 1 1 // 3 rows, 3 columns, min is 1, max is 1: 2-D structuring element
1 1 // origin is at row index 1 and column index 1.
1 1 1
1 **1** 1
1 1 1

c) choice (args [2]): 1 (dilation), or 2 (erosion), or 3 (opening), or 4 (closing), or 5 (all 4 operations).

**********
II. Outputs:

- prettyPrintFile (args [3]): as specs dictates.
- dilationOutFile (not from args): the result of dilation image with image header as in inFile.
- erosionOutFile (not from args): the result of erosion image with image header, as in inFile.
- closingOutFile (not from args): the result of closing image with image header, as in inFile.
- openingOutFile (not from args): the result of opening image with image header, as in inFile.

**********
III. Data structure:
**********

- Morphology class
  - (int) numImgRows
  - (int) numImgCols
  - (int) imgMin
  - (int) imgMax
  - (int) numStructRows
  - (int) numStructCols
  - (int) structMin
  - (int) structMax
  - (int) rowOrigin
  - (int) colOrigin
  - (int) rowFrameSize // set to (numStructRows / 2), integer division, i.e., 3/2 is 1; 4/2 is 2; 5/2 is 2.
  - (int) colFrameSize // set to (numStructCols / 2).
  - (int) extraRows // set to (rowFrameSize * 2)
  - (int) extraCols // set to (colFrameSize * 2)
  - (int) rowSize // set to (numImgRows + extraRows)
  - (int) colSize // set to (numImgCols + extraCols)
  - (int[][]) zeroFramedAry // a dynamically allocate 2D array, size of rowSize by colSize.
  - (int[][]) morphAry // Same size as zeroFramedAry.
  - (int [][]) tempAry // Same size as zeroFramedAry.
    // tempAry is to be used as the intermediate result within opening and closing operations.
  - (int [][]) structAry //a dynamically allocate 2D array of size numStructRows by numStructCols.

  Methods:
  - constructor (..) // may performs all allocations and initializations.
  - zero2DAry (Ary, nRows, nCols) // Set the entire Ary (nRows by nCols) to zero.
  - loadImg (…) // load imgFile to zeroFramedAry inside of frame, begins at (rowOrigin, colOrigin). On your own!
  - loadstruct (…) // load structFile to structAry. On your own!
  - ComputeDilation (inAry, outAry) // process every pixel in inAry, store result in outAry // see algorithm below.
  - ComputeErosion (inAry, outAry) // process every pixel in inAry, store result in outAry // see algorithm below.
  - ComputeOpening (inAry, outAry, tmp) // see algorithm below.
  - ComputeClosing (inAry, outAry, tmp) // see algorithm below.
  - onePixelDilation (i, j, inAry, outAry) // Perform dilation on pixel (i, j) with structAry. // See algorithm below.
  - onePixelErosion (i, j, inAry, outAry) // Perform erosion on pixel (i, j) with structAry. // See algorithm below.
  - AryToFile (inAry, fileOut) // **output the image header (same as input image header)**
    // output pixels inside of frame of inAry to fileOut
  - binaryPrettyPrint (inAry, fileOut) // output all pixels in inAry, including pixels in the frame.
    // if inAry [i, j] == 0 output '. ' // a period follows by a blank
    // else output '1 ' // 1 follows by a blank
    // Note: make sure all displays fit within a page, no wrapped around; use "Courier new" font
    //with font size 5 or 6 to fit in one page. -2 for poor displays.

**********
IV. Main(...)
**********
Step 0: inFile, structFile ← open via args [] for reading.

Step 1: numImgRows, numImgCols, imgMin, imgMax ← read from inFile.
numStructRows, numStructCols, structMin, structMax ← read from structFile.
rowOrigin, colOrigin ← read from structFile.

Step 2: zeroFramedAry, structAry, morphAry, tempAry ← dynamically allocate // see description in the above.
initialized all members of the class. // see description in the above.

Step 3: zero2DAry (zeroFramedAry, rowSize, colSize) // see description in the above.

Step 4: loadImg (inFile, zeroFramedAry) // see description in the above.
binaryPrettyPrint (zeroFramedAry, prettyPrintFile) // with caption.

Step 5: zero2DAry (structAry, numStructRows, numStructCols)
loadstruct (structFile, structAry)
binaryPrettyPrint (structAry, prettyPrintFile) // with captions.

Step 6: choice ← from args [2] // Use Integer.parseInt () method.
Step 7: if choice is 1
process1 (prettyPrintFile)
if choice is 2
process2 (prettyPrintFile)
if choice is 3
process3 (prettyPrintFile)
if choice is 4
process4 (prettyPrintFile)
if choice is 5
process5 (prettyPrintFile)

Step 8: close all files

**********
V. process1 (prettyPrintFile)
**********
Step 1: fileName ← "dilationOutFile.txt"
outFile ← open (fileName)
Step 2: zero2DAry (morphAry, rowSize, colSize)
ComputeDilation (zeroFramedAry, morphAry)
AryToFile (morphAry, outFile)
binaryPrettyPrint (morphAry, prettyPrintFile)
Step 3: close outFile

**********
VI. process2 (prettyPrintFile)
**********
Step 1: fileName ← "erosionOutFile.txt"
outFile ← open (fileName)

Step 2: zero2DAry (morphAry, rowSize, colSize)
ComputeErosion (zeroFramedAry, morphAry)
AryToFile (morphAry, outFile)
binaryPrettyPrint (morphAry, prettyPrintFile) //with captions.

Step 3: close outFile

```
******************************
VII. process3 (prettyPrintFile)
******************************
Step 1: fileName ← "openingOutFile.txt"
        outFile ← open (fileName)

Step 2:  zero2DAry (morphAry, rowSize, colSize)
         ComputeOpening (zeroFramedAry, morphAry, tempAry)
         AryToFile (morphAry, outFile)
         binaryPrettyPrint (morphAry, prettyPrintFile) //with captions.

Step 3: close outFile

******************************
VIII. process4 (prettyPrintFile)
******************************
Step 1: fileName ← "closingOutFile.txt"
        outFile ← open (fileName)

Step 2:  zero2DAry (morphAry, rowSize, colSize)
         ComputeClosing (zeroFramedAry, morphAry, tempAry)
         AryToFile (morphAry, outFile)
         binaryPrettyPrint (morphAry, prettyPrintFile) //with captions.

Step 3: close outFile

******************************
IX.   process5 (prettyPrintFile)
******************************
 Step 1: fileName ← "dilationOutFile.txt"
         outFile ← open (fileName)
         zero2DAry (morphAry, rowSize, colSize)
         ComputeDilation (zeroFramedAry, morphAry)
         AryToFile (morphAry, outFile)
         binaryPrettyPrint (morphAry, prettyPrintFile)
         close (outFile)

Step 2:  fileName ← "erosionOutFile.txt"
         outFile ← open (fileName)
         zero2DAry (morphAry, rowSize, colSize)
         ComputeErosion (zeroFramedAry, morphAry)
         AryToFile (morphAry, outFile)
         binaryPrettyPrint (morphAry, prettyPrintFile) //with captions.
         close (outFile)

Step 3: fileName ← "openingOutFile.txt"
         outFile ← open (fileName)
         zero2DAry (morphAry, rowSize, colSize)
         ComputeOpening (zeroFramedAry, morphAry, tempAry)
         AryToFile (morphAry, outFile)
         binaryPrettyPrint (morphAry, prettyPrintFile) //with captions.
         close (outFile)

Step 4:  fileName ← "closingOutFile.txt"
         outFile ← open (fileName)
         zero2DAry (morphAry, rowSize, colSize)
         ComputeClosing (zeroFramedAry, morphAry, tempAry)
         AryToFile (morphAry, outFile)
         binaryPrettyPrint (morphAry, prettyPrintFile) //with captions.
         close (outFile)
```

```
*******************************
X. ComputeDilation (inAry, outAry) // process dilation on each pixel inside of zeroFramedAry.
*******************************
step 1: i ← rowFrameSize
step 2: j ← colFrameSize
step 3: if inAry [i, j] > 0
          onePixelDilation (i, j, inAry, outAry) // only processing one pixel inAry[i,j]
step 4: j++
step 5: repeat step 3 to step 4 while j < (colSize)
step 6: i++
step 7: repeat step 2 to step 6 while i < (rowSize)

*******************************
XI. ComputeErosion (inAry, outAry) // process erosion on each pixel inside of zeroFramedAry
*******************************
step 1: i ← rowFrameSize
step 2: j ← colFrameSize
step 3: if inAry[i, j] > 0
          onePixelErosion (i, j, inAry, outAry) // only processing one pixel inAry[i,j]
step 4: j++
step 5: repeat step 3 to step 4 while j < (colSize)
step 6: i++
step 7: repeat step 2 to step 6 while i < (rowSize)

*******************************
XII. onePixelDilation (i, j, inAry, outAry)
*******************************
step 0 : iOffset ← i - rowOrigin
         jOffset ← j - colOrigin
           // translation of image's coordinate (i, j) with respected to the origin of the structuring element
step 1: rIndex ← 0
step 2: cIndex ← 0
step 3: if (structAry[rIndex][cIndex] > 0)
          outAry[iOffset + rIndex][jOffset + cIndex] ← 1
step 4: cIndex ++
step 5: repeat step 3 to step 4 while cIndex < numStructCols
step 6: rIndex ++
step 7: repeat step 2 to step 6 while rIndex < numStructRows

*******************************
XIII. onePixelErosion (i, j, inAry, outAry)
*******************************
step 0 : iOffset ← i - rowOrigin
         jOffset ← j - colOrigin
           // translation of image's coordinate (i, j) with respected of the origin of the structuring element
         matchFlag ← true
step 1: rIndex ← 0
step 2: cIndex ← 0
step 3: if (structAry[rIndex][cIndex] > 0) and (inAry[iOffset + rIndex][jOffset + cIndex] ) <= 0)
            matchFlag ← false
step 4: cIndex ++
step 5: repeat step 3 to step 4 while (matchFlag == true) and (cIndex < numStructCols )
step 6: rIndex ++
```

step 7: repeat step 2 to step 6 while (matchFlag == true) and (rIndex < numStructRows)
step 8: if matchFlag == true
outAry[i][j] ← 1
else
outAry[i][j] ← 0

**********
XIV. ComputeClosing (zeroFramedAry, morphAry, tempAry)
**********
step 1: ComputeDilation (zeroFramedAry, tempAry)
step 2: ComputeErosion (tempAry, morphAry)

**********
XV. ComputeOpening (zeroFramedAry, morphAry, tempAry)
**********
step 1: Compute Erosion (zeroFramedAry, tempAry)
step 2: ComputeDilation (tempAry, morphAry)