Project 5 (Java): Implement the thinning algorithm as taught in class. Thinning is the 2nd methods to obtain the skeletons of objects in a given binary image. The thinning of an object is like peeling off one layer of object from 4 sides (north, south, west, and east, with three conditions) in iterations, until the object becomes a skeleton (i.e., no more pixel can be peel off).

*** This is an easy project which can be done in few hours.

What you need to do:
1) You will have for (4) date files: data1, data2, data3 and data4 to test your program.
2) Run your program four times using each of test data
3) Include in your hard copies:
    - cover page
    - source code
    - outFile1 for data1
    - outFile2 for data1
    - outFile1 for data2
    - outFile2 for data2
    - outFile1 for data3
    - outFile2 for data3
    - outFile1 for data4
    - outFile2 for data4

****

Language: Java
Project points: 10pts
Due Date: <u>Soft copy (*.zip) and hard copies (*.pdf)</u>:

+1 (11/10 pts): early submission, 3/27/2022 Tuesday before midnight.
-0 (10/10 pts): on time, 3/29/2022 Wednesday before midnight.
-1 (9/10 pts): 1 day late, 3/30/2022 Wednesday before midnight.
-2 (8/10 pts): 2 days late, 3/31/2022 Thursday before midnight.
(-10/10 pts): none submission, 3/12/2022 Saturday after midnight

*** Name your soft copy and hard copy files using the naming convention as given in the project submission requirement.
*** All on-line submission MUST include Soft copy (*.zip) and hard copy (*.pdf) in **<u>the same email attachments</u>** with correct email subject as stated in the email requirement; otherwise, your submission will be rejected.

****

I. Input: inFile (args [0]): a binary image

****

II. Outputs: There are two outfiles:
    a) outFile1 (args [1]): to store the final thinning result with the image header.
    b) outFile2 (args [2]):
        - prettyPrint input image with proper caption.
        - prettyPrint after completing each cycle (after thinning all sides) with proper caption, i.e.,
          ("result of thinning: cycle – 1")
          ("result of thinning: cycle – 2")

****

III. Data structure:

****

- A Thinning class
    - (int) numRows
    - (int) numCols
    - (int) minVal
    - (int) maxVal
    - (int) changeflag
    - (int) cycleCount
    - (int) aryOne [][] // a 2D array, need to dynamically allocate at run time of size numRows + 2 by numCols + 2.
                // aryOne is for checking those three conditions for thinning.
    - (int) aryTwo [][] // a 2D array, need to dynamically allocate at run time of size numRows + 2 by numCols + 2
                // aryTwo is for storing the intermediate result each side of thinning.

```
- methods:
- constructor(...) // need to dynamically allocate aryOne and aryTwo, etc.
- zeroFrame (…)// zero framing the two extra rows and two extra columns.
- loadImage (inFile, aryOne) // Read from the input file onto inside frame of aryOne
- copyArys (…) // always copy aryTwo to aryOne
- thinning (aryOne, aryTwo) // call the four thinning methods to thin one layer in each iteration.
- NorthThinning (aryOne, aryTwo) // As taught in class. On your own.
- SouthThinning (aryOne, aryTwo) // As taught in class. On your own.
- WestThinning (aryOne, aryTwo) // As taught in class. On your own.
- EastThinning (aryOne, aryTwo) // As taught in class. On your own.
- prettyPrint (aryTwo, outFile2, cycleCount) // Since the image is binary, there is no need to do reformatting.
                // First, output the caption with cycleCount, then output all pixels in aryTwo,including pixels
                //outside of the frame, to outFile2, use "." for zeros
- printAry (ary, outfile1) // print all pixels inside the frame of ary to outfile1 without reformatting nor prettyPrint
                // but with spaces between two pixels.
```

****
IV. main (...)
****

```
Step 0: inFile ← open input file from args [0]
    numRows, numCols, minVal, maxVal ← read from inFile
    outFile1, outFile2 ← open from args []
    outFile1 ← write numRows, numCols, minVal, maxVal to outFile header
    dynamically allocate all arrays and initialize via constructor.
    zeroFrame(aryOne) // For Java, you may not need to do this
    zeroFrame(aryTwo) // For Java, you may not need to do this
Step 1: loadImage (inFile, aryOne)
Step 2: cycleCount ← 0
Step 3: prettyPrint (aryTwo, outFile2, cycleCount) // This print is before thinning
Step 4: changeFlag ← 0
Step 5: thinning (aryOne, aryTwo)
Step 6: cycleCount ++
Step 7: prettyPrint (aryTwo, outFile2, cycleCount)
Step 8: repeat step 4 to step 7 while changeFlag > 0
Step 9: printAry (aryOne, outFile1) ← output inside frame of firstAry from [1][1] with space between 0's 1's
Step 10: close all files
```

****
V. thinning (aryOne, aryTwo)
****

```
Step 1: NorthThinning (aryOne, aryTwo)
    copyArys (aryTwo, aryOne)
Step 2: SouthThinning (aryOne, aryTwo)
    copyArys (aryTwo, aryOne)
Step 3: WestThinning (aryOne, aryTwo)
    copyArys (aryTwo, aryOne)
Step 4: EastThinning (aryOne, aryTwo)
    copyArys (aryTwo, aryOne)
```