Project 6.2: An image was encoded with one of the four methods below.

Your task is to re-create the original image from its run-length encode text file with respect to the encoding methods used.

Method 1) Encode without zero and no wrap-around.

Method 2) Encode without zero and wrap-around.

Method 3) Encode with zero and no wrap-around. //algorithm given below.

Method 4) Encode with zero and wrap-around.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Language: Java

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Due date: Soft copies: 4/11/2018, Wednesday before Midnight

Early submission deadline: 4/8/2018, before midnight

Due date: Hard copies: 4/12/2018, Thursday in class

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Input (argv): An encoded text file of an image, in the following format:

The first text line is the input image header;

The second text line is the method used;

follows by one run-length per text line

Example:

20 15 0 9 // header information

2 // method 2 was used to encode

1 4 8 10 // startRow is 1, startCol is 4, color is 8, and 10 pixels long

2 4 7 5 // startRow is 2, startCol is 4, color is 7, and 5 pixels long

:

:

:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

II. Output (argv): the original image (including header information.)

Run this program with the results of four encoded run-length of your run-length encoding project

// Check (eyeballing) to see if your program works correctly to produce the original image.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

III. Data structure: On your own!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Use object-oriented data structures similar to your encoding project 6.1 or those given in the previous project specs.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IV. Algorithm in main

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 0: inFile 🡨 open input file

outFile 🡨 open a text file to write

initializing those needed in the data structures

step 1: Read the image header from inFile

Write the header to outFile

step 2: methodUsed <-- read from inFile

if methodUsed is not within 1 – 4

exit with error message

step 3: case of methodUsed

1: call deCodeMethod1 (inFile) // on your own

2: call deCodeMethod2 (inFile) // on your own

3: call deCodeMethod3 (inFile) // on your own

4: call deCodeMethod4 (inFile)) // on your own

step 4: closed all files