**CS381-37: Project 7.1 (JAVA)**

**Yida Tao**

**Due date: Apr. 12, 2018**

Algorithm Steps:

step 0: - open all files

* initializing those needed in the data structures
* Read the image header and output the image header to outFile

step 1: Display to the console those four method and ask the user to input choice

step 2: method <-- get from the user from the console

if method is not within 1 – 4

exit with error message

step 3: output method to outFile

case of method

1: call method1 (inFile) // given in class

2: call method2 (inFile) // on your own

3: call method3 (inFile) //

4: call method4 (inFile) // on your own

step 4: closed all files

**Source Code**

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**public** **class** project7\_1 {

**public** **static** **void** main(String[] args) {

**int** method = 0;

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));

**try**{

System.***out***.println("argument 1:" + args[0]);

System.***out***.println("argument 2:" + args[1]);

}**catch**(ArrayIndexOutOfBoundsException e){

System.***out***.println("no arguments");

System.*exit*(0);

}

System.***out***.println("Which method do you want to encode? ");

System.***out***.println("Method 1) Encode without zero and no wrap-around");

System.***out***.println("Method 2) Encode without zero and wrap-around");

System.***out***.println("Method 3) Encode with zero and no wrap-around");

System.***out***.println("Method 4) Encode with zero and wrap-around");

System.***out***.println("Give me a number (1,2,3,4)");

**try**{

method = Integer.*parseInt*(br.readLine());

} **catch** (IOException e) {

e.printStackTrace();

} **catch**(NumberFormatException e){

System.***out***.println("input is not a number!");

System.*exit*(0);

}

**if**(method < 1 || method > 4){

System.***out***.println("invaild number!");

System.*exit*(0);

}

runLengthEncoder rle = **new** runLengthEncoder(method, args[0], args[1]);

rle.encode();

}

}

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** runLengthEncoder {

**int** method;

String infile;

String outfile;

**int** numRows = 0;

**int** numCols = 0;

**int** minVal = 0;

**int** maxVal = 0;

**int** greyScale = 0;

**public** runLengthEncoder(**int** m, String in, String out){

method = m;

infile = in;

outfile = out;

}

**public** **void** encode(){

Scanner sc = **null**;

FileOutputStream fos = **null**;

**try** {

fos = **new** FileOutputStream(outfile);

System.*setOut*(**new** PrintStream(fos));

} **catch** (FileNotFoundException e) {

e.printStackTrace();

}

**try** {

sc = **new** Scanner(**new** File(infile));

} **catch** (FileNotFoundException e) {

e.printStackTrace();

System.***out***.println("Cant find the file: " + infile);

}

numRows = sc.nextInt();

numCols = sc.nextInt();

minVal = sc.nextInt();

maxVal = sc.nextInt();

System.***out***.println(numRows + " " + numCols + " " + minVal + " " + maxVal);

System.***out***.println(method);

**int** count = 0;

**int** currVal = 0;

**int** totalCount = 0;

**int** srow = 0;

**int** scol = 0;

**if**(method == 1){

currVal = sc.nextInt();

totalCount++;

**while**(sc.hasNextInt()){

greyScale = currVal;

srow = (totalCount-1)/numCols;

scol = (totalCount-1)%numCols;

count = 1;

**if**(sc.hasNextInt()){

currVal = sc.nextInt();

totalCount++;

}

**while**(currVal == greyScale){

**if**((totalCount-1)%numCols == 0){

**break**;

}

count++;

**if**(sc.hasNextInt()){

currVal = sc.nextInt();

totalCount++;

}

**else** **break**;

}

**if**(greyScale != 0){

System.***out***.println(srow +" "+ scol +" "+ greyScale +" "+ count);

}

}

}

**else** **if**(method == 2){

currVal = sc.nextInt();

totalCount++;

**while**(sc.hasNextInt()){

greyScale = currVal;

srow = (totalCount-1)/numCols;

scol = (totalCount-1)%numCols;

count = 1;

**if**(sc.hasNextInt()){

currVal = sc.nextInt();

totalCount++;

}

**while**(currVal == greyScale){

count++;

**if**(sc.hasNextInt()){

currVal = sc.nextInt();

totalCount++;

}

**else** **break**;

}

**if**(greyScale != 0){

System.***out***.println(srow +" "+ scol +" "+ greyScale +" "+ count);

}

}

}

**else** **if**(method == 3){

currVal = sc.nextInt();

totalCount++;

**while**(sc.hasNextInt()){

greyScale = currVal;

srow = (totalCount-1)/numCols;

scol = (totalCount-1)%numCols;

count = 1;

**if**(sc.hasNextInt()){

currVal = sc.nextInt();

totalCount++;

}

**while**(currVal == greyScale){

**if**((totalCount-1)%numCols == 0){

**break**;

}

count++;

**if**(sc.hasNextInt()){

currVal = sc.nextInt();

totalCount++;

}

**else**{

**break**;

}

}

System.***out***.println(srow +" "+ scol +" "+ greyScale +" "+ count);

}

}

**else** **if**(method == 4){

currVal = sc.nextInt();

totalCount++;

**while**(sc.hasNextInt()){

greyScale = currVal;

srow = (totalCount-1)/numCols;

scol = (totalCount-1)%numCols;

count = 1;

**if**(sc.hasNextInt()){

currVal = sc.nextInt();

totalCount++;

}

**while**(currVal == greyScale){

count++;

**if**(sc.hasNextInt()){

currVal = sc.nextInt();

totalCount++;

}

**else**{

**break**;

}

}

System.***out***.println(srow +" "+ scol +" "+ greyScale +" "+ count);

}

}

sc.close();

**try** {

fos.close();

System.*setOut*(**new** PrintStream(**new** FileOutputStream(FileDescriptor.***out***)));

} **catch** (IOException e) {

e.printStackTrace();

}

System.***out***.println("Done");

}

}

**Input**

15 20 0 9

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4

0 0 4 4 4 4 4 4 4 4 4 0 0 0 0 0 0 0 0 0

0 0 0 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

3 3 3 0 0 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7

7 7 7 7 7 7 7 7 7 7 0 0 0 0 0 0 0 0 0 0

0 0 1 1 1 1 1 1 1 1 0 0 0 8 8 8 8 9 9 9

9 9 9 9 9 0 0 0 9 9 9 9 9 0 0 0 0 8 8 8

8 8 8 9 9 0 0 0 9 9 9 9 9 0 0 0 0 0 0 0

0 0 0 0 0 0 0 2 2 2 2 2 3 3 4 4 4 4 4 4

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 6 6 6

6 6 6 6 6 6 6 6 6 6 0 0 0 0 0 0 0 0 0 0

**Output 1**

15 20 0 9

1

0 15 4 5

1 2 4 9

2 5 3 15

3 0 3 3

3 5 3 6

3 11 7 9

4 0 7 10

5 2 1 8

5 13 8 4

5 17 9 3

6 0 9 5

6 8 9 5

6 17 8 3

7 0 8 3

7 3 9 2

7 8 9 5

8 7 2 5

8 12 3 2

8 14 4 6

12 2 1 18

13 0 1 10

13 10 6 10

14 0 6 10

**Output 2**

15 20 0 9

2

0 15 4 5

1 2 4 9

2 5 3 18

3 5 3 6

3 11 7 19

5 2 1 8

5 13 8 4

5 17 9 8

6 8 9 5

6 17 8 6

7 3 9 2

7 8 9 5

8 7 2 5

8 12 3 2

8 14 4 6

12 2 1 28

13 10 6 20

**Output 3**

15 20 0 9

3

0 0 0 15

0 15 4 5

1 0 0 2

1 2 4 9

1 11 0 9

2 0 0 5

2 5 3 15

3 0 3 3

3 3 0 2

3 5 3 6

3 11 7 9

4 0 7 10

4 10 0 10

5 0 0 2

5 2 1 8

5 10 0 3

5 13 8 4

5 17 9 3

6 0 9 5

6 5 0 3

6 8 9 5

6 13 0 4

6 17 8 3

7 0 8 3

7 3 9 2

7 5 0 3

7 8 9 5

7 13 0 7

8 0 0 7

8 7 2 5

8 12 3 2

8 14 4 6

9 0 0 20

10 0 0 20

11 0 0 20

12 0 0 2

12 2 1 18

13 0 1 10

13 10 6 10

14 0 6 10

14 10 0 10

**Output 4**

15 20 0 9

4

0 0 0 15

0 15 4 5

1 0 0 2

1 2 4 9

1 11 0 14

2 5 3 18

3 3 0 2

3 5 3 6

3 11 7 19

4 10 0 12

5 2 1 8

5 10 0 3

5 13 8 4

5 17 9 8

6 5 0 3

6 8 9 5

6 13 0 4

6 17 8 6

7 3 9 2

7 5 0 3

7 8 9 5

7 13 0 14

8 7 2 5

8 12 3 2

8 14 4 6

9 0 0 62

12 2 1 28

13 10 6 20

14 10 0 10