Colin Kincaid Section #5

CS 106A July 27, 2018

Solutions to Section #5

**1. Warmup: Trace**

**Array 1: [10, 9, 9, 6, 6]**

**Array 2: [12, 12, 11, 11, 9, 8]**

2. Index Of

private int indexOf(int[] list, int target) {

for (int i = 0; i < list.length; i++) {

if (list[i] == target) {

return i;

}

}

return -1;

}

3. Unique Numbers

private int numUnique(int[] list) {

if (list.length == 0) {

return 0;

}

int count = 1;

for (int i = 1; i < list.length; i++) {

if (list[i] != list[i - 1]) {

count++;

}

}

return count;

}

4. Banish

private void banish(String[] a1, String[] a2) {

for (int i = 0; i < a1.length; i++) {

// see whether a1[i] is contained in a2

boolean found = false;

for (int j = 0; j < a2.length && !found; j++) {

if (a1[i].equals(a2[j])) {

found = true;

}

}

if (found) { // shift all elements of a1 left by 1

for (int j = i + 1; j < a1.length; j++) {

a1[j - 1] = a1[j];

}

a1[a1.length - 1] = “”;

i--; // so that we won't skip an index

}

}

}

}

5. Collapse

private int[] collapse(int[] list) {

int[] result = new int[list.length / 2 + list.length % 2];

for (int i = 0; i < result.length - list.length % 2; i++) {

result[i] = list[2 \* i] + list[2 \* i + 1];

}

if (list.length % 2 == 1) {

result[result.length - 1] = list[list.length - 1];

}

return result;

}

**6. Histogram**

/\*

\* File: Histogram.java

\* --------------------

\* This program reads a list of exam scores, with one score per line.

\* It then displays a histogram of those scores, divided into the

\* ranges 0-9, 10-19, 20-29, and so forth, up to the range containing

\* only the value 100.

\*/

import acm.program.\*;

import acm.util.\*;

import java.io.\*;

import java.util.\*;

public class Histogram extends ConsoleProgram {

public void run() {

initHistogram();

readScoresIntoHistogram();

printHistogram();

}

/\* Initializes the histogram array \*/

private void initHistogram() {

histogramArray = new int[11];

for (int i = 0; i < histogramArray.length; i++) {

histogramArray[i] = 0;

}

}

/\* Reads the exam scores, updating the histogram \*/

private void readScoresIntoHistogram() {

try {

Scanner fileScanner =

new Scanner(new File(DATA\_FILE));

while (fileScanner.hasNextLine()) {

String line = fileScanner.nextLine();

int score = Integer.parseInt(line);

if (score < 0 || score > 100) {

fileScanner.close();

throw new ErrorException(

"That score is out of range");

} else {

int range = score / 10;

histogramArray[range]++;

}

}

fileScanner.close();

} catch (IOException ex) {

throw new ErrorException(ex);

}

}

/\* Displays the histogram \*/

private void printHistogram() {

for (int range = 0; range <= 10; range++) {

String label;

if (range == 0) {

label = "00-09";

} else if (range == 10) {

label = " 100";

} else {

label = (10 \* range) + "-" + (10 \* range + 9);

}

String stars = createStars(histogramArray[range]);

println(label + ": " + stars);

}

}

/\* Creates a string consisting of n stars \*/

private String createStars(int n) {

String stars = "";

for (int i = 0; i < n; i++) {

stars += "\*";

}

return stars;

}

/\* Private instance variables \*/

private int[] histogramArray;

/\* Name of the data file \*/

private static final String DATA\_FILE = "res/MidtermScores.txt";

}

**7. How Prime!**

**/\* File: Sieve.java**

**\* ------------------------**

**\* This program prints out prime numbers in the range**

**\* up to and including UPPER\_LIMIT.**

**\*/**

**import acm.program.\*;**

**public class Sieve extends ConsoleProgram {**

**private static final int UPPER\_LIMIT = 1000;**

**public void run() {**

**// crossedOff[i] represents the number i + 2;**

**boolean[] crossedOff = new boolean[UPPER\_LIMIT - 1];**

**for (int i = 0; i < crossedOff.length; i++) {**

**crossedOff[i] = false;**

**}**

**for (int n = 0; n < crossedOff.length; n++) {**

**if (!crossedOff[n]) {**

**int number = n + 2;**

**println(number);**

**// Cross off all the multiples of n**

**for (int k = n; k < crossedOff.length; k += number) {**

**crossedOff[k] = true;**

**}**

**}**

**}**

**}**

**}**

**8. Flip Vertical**

**private void flipVertical(GImage image) {**

**int[][] pixels = image.getPixelArray();**

**int width = pixels[0].length;**

**int height = pixels.length;**

**for (int col = 0; col < width; col++) {**

**for (int p1 = 0; p1 < height / 2; p1++) {**

**int p2 = height - p1 - 1;**

**int temp = pixels[p1][col];**

**pixels[p1][col] = pixels[p2][col];**

**pixels[p2][col] = temp;**

**}**

**}**

**image.setPixelArray(pixels);**

**}**

**9. Stretch**

**private void stretch(GImage image, int factor) {**

**int[][] pixels = image.getPixelArray();**

**int[][] result = new int[pixels.length][pixels[0].length \* factor];**

**for (int row = 0; row < result.length; row++) {**

**for (int col = 0; col < result[0].length; col++) {**

**result[row][col] = pixels[row][col / factor];**

**}**

**}**

**image.setPixelArray(result);;**

**}**

**10. 2D Arrays Trace**

**4, 5, 6, 6**

**5, 6, 7, 7**

**6, 7, 8, 8**