COMPUTER SCIENCE 11A (FALL TERM, 2016)
PROGRAMMING IN JAVA

PROGRAMMING ASSIGNMENT 6

Overview:

This assignment tests your understanding of everything covered so far, including arrays.

Modularity in your code is very important, YOU MUST USE STATIC METHODS.

Problem 1

Write a method called `isUnique` that accepts an array of integers as a parameter and returns a `boolean` value indicating whether or not the values in the array are unique (`true` for yes, `false` for no). The values in the list are considered unique if there is no pair of values that are equal. For example, if passed an array containing {3, 8, 12, 2, 9, 17, 43, -8, 46}, your method should return `true`, but if passed {4, 7, 3, 9, 12, -47, 3, 73}, your method should return `false` because the value 3 appear twice.

You should also write the `main` method, which interacts with the user, calls the static method `isUnique`, and prints the result.

Problem 2

Write a method called `longestSortedSequence` that accepts an array of integers as a parameter and returns the length of the longest sorted sequence of integers in the array. For example, in the array {3, 8, 10, 1, 9, 14, -3, 0, 14, 207, 56, 98, 12}, the longest sorted sequence in the array has four values in it (-3, 0, 14, 207), so your method would return 4 if passed this array. Your method should return 0 if passed an empty array.

You should also write the `main` method, which interacts with the user, calls the static method `longestSortedSequence`, and prints the result.

Problem 3

Write a method called `priceIsRight` that mimics the guessing rules from the game show *The Price is Right*. The method accepts as parameters an array of integers representing the contestants' bids and an integer representing a correct price. The method returns the element in the bids array that is closets in value to the correct price without being larger than that price. For example, is an array called bids stores the value {200, 300, 250, 1, 950, 40} the call of `priceIsRight(bids, 280)` should return 250, since 250 is the bid closets to 280 without going over 280. If all bids are larger than the correct price, your method should return -1.

You should also write the `main` method, which interacts with the user, calls the static method `priceIsRight`, and prints the result.

Problem 4

Write a program that allows a user to play tic-tac-toe against the computer. The user can choose who goes first, and can choose to play again when the current game is over. The computer's strategy should do its best to win: if it can complete a two-in-a-row if possible, and otherwise will block the user's 2-in-a-row if possible. The program should recognize the end of each game, and say whether the user won, the computer won, or that it ended in a row.

Problem 5

WARNING!!! DON'T PANIC. THERE IS A LOT TO READ BUT AT THE END OF THE DAY YOU NEED ONLY TO IMPLEMENT THREE STATIC METHODS

For this problem you will be working with images. A framework has been provided to you, which reflects the real-world scenario in which most programmers develop code; your code will build on top of existing work previously started by others. In the assignment's .zip file there are two java files: `CS11Editor.java` and `CS11AImage.java`.Your code goes in the file `CS11AImage.java`.
Also, in the .zip file you will find an image `cute.jpg` used to test your program.

Description

An image is stored as three-dimensional array; a two-dimensional array of pixels, where each pixel is stored as a three-element integer array.
Your task is to **write only three methods** that are already defined (NOTE: *the last method* `colorReplace()` *is extra credit*). Below is the list of the methods to implement with a brief description. One import note when writing these methods: the variables toward the top of the `CS11AImage.java` file i.e.,

```
public BufferedImage im = null;
public int[] packedData = null;
public int[][][] pixelData = null;
public int height = 0;
public int width = 0;
```

are effectively global variables: you can read and write their values from any method, even though they are not explicitly passed to that method. You are encouraged to use the values of `height` and `width`, but should not change their value. You should (in fact, must) both use and change the values in `pixelData`. **Changing `height`, `width`, `im`, or `packedData` could result in other parts of the program not working.**

1. Implement the method `flipHorizontal()`, which reflects an image across the vertical axis, as shown below.

2. Implement the method `flipVertical()`, which reflects an image across the horizontal axis, as shown below.

3. Implement the method `invert()`, which inverts the color of an image. Invert a pixel by replacing each of its channels (red, green, and blue), with the difference between that number and 255. The result of this operation is shown below.

4. EXTRA CREDIT Implement the method `colorReplace()`. This method takes as arguments two "colors", that is, three-element integer arrays, and an integer `range`. You should replace each pixel that is within `range` of `oldColor` with `newColor`. A pixel is in `range` if the value of each of its three channels, red, green, and blue, is within the corresponding channel in `oldColor` plus or minus range. As usual, sample output is shown. See the **Sample Interaction** at the end of this document for the command to produce this image.

Getting Started

Place the contents of the zip file in a directory. Run
```
javac CS11Editor.java CS11AImage.java
```
from that directory. Then run
```
java CS11Editor
```
to run the program.
At this point you can familiarize yourself with the program by reading and writing images, but all the program can do is make copies of images. In order to change them, you will have to implement the methods, specified above, in the `CS11AImage.java` file. If you try running any of the commands `flip-horiz`, `flip-vert`, `invert`, or `replace`, it will print a message saying that the method is not yet implemented. See the **Sample Interaction** section at the end of this document for an example use of the program.

Implementing Methods

There are four methods already defined for you:

```
public void flipHorizontal()
public void flipVertical()
public void invert()
public void replaceColor(int[] oldColor, int[] newColor, int range)
```

And right now all any of them does is print a message saying it is not yet implemented.

These methods may seem strange because none of them has the "`static`" keyword. This has to do with the way objects work in Java and gives all non-static methods access to the five variables defined at the top of the class.

Notice that these first three methods take no parameters and return no values. This is because they have access to the only input they need: `pixelData`. Also, instead of returning anything, they should just change the variable `pixelData` so that it contains the result. When you tell the program to "save" an image, it looks at the current contents of `pixelData` and writes that to disk.

Let's take a look at `pixelData`. As you can see from the definition, it is a three-dimensional array of integers.

```
public int[][][] pixelData = null;
```

It looks like it is `null`, and it is declared as such, but before your methods ever get called, the `CS11AImage` "constructor" will fill it with data from the image you load. The dimensions of this array are `[height][width][3]`. This means that the first dimension is an array with length equal to height of two-dimensional arrays, whose dimensions are `[width][3]`. So each row in the image is an array of 3-element arrays, with a length equal to the width of the image. Thus, each 3-element array represents a single pixel. If this is confusing, use the `Arrays.deepToString()` method to print a small image, such as `test.png` to see how the data is stored.

Each pixel is stored as a 3-element array of integers. There are many ways to represent images, and we're using RGB images. This means that the color information is broken up into three "channels," each representing one of the three colors red, green, and blue. Each of these channels has 8 bits of information, so its value can range from 0 to 255. We are using integers to represent these because Java does not have an unsigned-byte data type. The colors are stored in the order red, green, blue, so if you assign one of the 3-element arrays to the variable `pxl`, `pxl[0]` is the red channel, `pxl[1]` is the green channel, and `pxl[2]` is the blue channel.[1]

---

[1] Choosing how to represent color involves weighing many considerations. There is no absolute "right" way to do so and it often depends on the application. Using integers may seem extremely inefficient (and it is—we're wasting three bytes of memory for every channel in every pixel) but it has significant benefits in ease of programming. You might think that using a `char` would be better, but in Java chars are signed and 2 bytes, so they are not ideal either. Java does have a Color class, so each pixel could be an instance of a Color object, but that would incur its own overhead and reduce the amount of practice you get with arrays.

You are welcome to define additional methods in your solution to this assignment. Note, however, that if you want to access any of the "global" variables (`pixelData`, `height`, or `width`) without passing them as parameters, you should leave the "`static`" keyword out of your method definition.

Sample Interaction

This section demonstrates how the program works by showing the interaction used to generate the images in this assignment from cute.jpg. Note that in this example the load command has been run between each operation on the image. This is so that each image saved has had exactly one operation performed. You can, however, run multiple commands before saving to "stack" effects. When you load an image it replaces whatever pixels are in memory without warning, so you need to use save if you want to output the result of your operations.

```
Welcome to the CS11A command line image editor.
Type 'help' for usage information or, if you know how to use it, just type your command at
the prompt.

Enter command:
load
Enter the name of the file you wish to load:
cute.jpg
640
428
Getting pixel values from packed data...
Image successfully loaded.
Enter command:
flip-horiz
Flipped Horizontally.
Enter command:
save
Enter the file path where the image should be saved:
cute-flipped-horiz.png
putting pixel values in packed format...
Image successfully saved.
Enter command:
flip-vert
Flipped Vertically.
Getting pixel values from packed data...
Image successfully loaded.
Enter command:
flip-vert
Flipped Vertically.
Enter command:
save
Enter the file path where the image should be saved:
cute-flipped-vertically.png
putting pixel values in packed format...
Image successfully saved.
Enter command:
load
Enter the name of the file you wish to load:
cute.jpg
640
428
Getting pixel values from packed data...
Image successfully loaded.
Enter command:
invert
Inverted.
Enter command:
save
Enter the file path where the image should be saved:
cute-inverted.png
putting pixel values in packed format...
```

```
Image successfully saved.
Enter command:
load
Enter the name of the file you wish to load:
cute.jpg
640
428
Getting pixel values from packed data...
Image successfully loaded.
Enter command:
replace
Enter integers for the color to replace:
Red (0 - 255):
200
Green (0 - 255):
130
Blue (0 - 255):
30
Enter integers for the new color:
Red (0 - 255):
20
Green (0 - 255):
60
Blue (0 - 255):
100
Enter an integer specifying how large a range of colors to replace:
30
Replaced color
Enter command:
save
Enter the file path where the image should be saved:
cute-puppy-painted.png
putting pixel values in packed format...
Image successfully saved.
Enter command:
quit
Terminating.
```

Guidelines:

For this assignment you should limit yourself to the Java features covered in class so far (lecture 29). Although we will cover other topics while you are working on this assignment, do not use any of those features.

Grading:

You will be graded on

- **External Correctness:** The output of your program should match exactly what is expected. Programs that do not compile will not receive points for external correctness.
- **Internal Correctness:** Your source code should follow the stylistic guidelines shown in class. Also, remember to include the comment header at the beginning of your program.

Submission:

Create a folder named PA6 containing all your java files. Name your java files Problem1.java, Problem2.java, etc. Zip the folder, named PA6_USERNAME.zip (where USERNAME is your Brandeis username – your email address without the @brandeis.edu), and submit it via Latte the day it is due, **Thursday, Nov 17 at 11:00pm**.

For example, if your email is dilant@brandeis.edu, you will submit a file PA6_dilant.zip.

NOTE: Make sure that your submission is in the correct format, and that all your files work as you intend them to before submitting. Additionally, make sure you are submitting the .java files, not the .class files. Any submissions not in a zip file will receive a zero, and any assignments that don't compile or that don't have .java files will receive a zero. There will be no exceptions to this rule.