CS 251 Intermediate Programming using Java
Lab 4: Nonograms 1
Prof. Tom Hayes
T.A. Aayush Gupta

This assignment will give you some experience working with objects, and interacting with a GUI.

Problem Specification

We are going to implement a puzzle game called Nonograms (also known as Picross, Griddlers, and Paint by Numbers puzzles). `https://en.wikipedia.org/wiki/Nonogram`

In a nonogram, the player tries to reconstruct an image from information about the lengths of groups of consecutive pixels that are all black. Any white pixels are considered "empty," and are excluded from consideration. Adjacent groups of black pixels must be separated by one or more empty pixels.

What you're getting

I've created a `Nonogram.java` file with a code skeleton for you to fill in. You will also need to work with a helper class, `NonogramGUI`, which I am providing for you.

What you have to do

You will supply several methods that implement the functionality of the `Nonogram` class. Add your code to the provided `Nonogram.java` class.

The `Nonogram` constructor takes a 2D array of booleans representing a desired solution, and initializes the fields to make a Nonogram for which that solution would be correct.

The `findGroupLengths` method is a helper method that returns a list of the group lengths in a given array of colors repesenting a single row or column of the image.

The `isGuessCorrect` method will test the user's answer to the puzzle to see if it satisfies all of the nonogram constraints. If the solution is valid, this method returns `true`; otherwise, it returns `false`, after printing a message to standard out describing which of the constraints was violated.

The `toString` method should return a string representation of the nonogram, including all of the group lengths for every row and column, as well as the user's current guess at the answer. This will be useful for debugging. For this method, you may assume that the puzzle is black-and-white.

There are also four "callback" methods that you must provide for use by the GUI.

The `handleMouseClickAt` method flips the color of one cell of the user's guess. The arguments will be a row and column in the puzzle, not a screen pixel!

The `handleMousePressAt` method records the location of the most recent mouse press. The arguments will be a row and column in the puzzle, not a screen pixel!

The `handleMouseReleaseAt` method flips a line of pixels between the most recent mouse press and mouse release locations. The arguments will be a row and column in the puzzle, not a screen pixel!

The `handleResetButtonClick` method resets the guess to all cells being empty.

Compiling and Running

1. Download the class files `Nonogram.java`, `NonogramGUI.class` and `NonogramPanel.class` from the course web site and place them in your working directory.

2. Add code to `Nonogram.java` to supply the missing functionality. There are comments to help you keep track of what you need to do.

3. Start small. Compile frequently. It is a good idea to start out with a minimal implementation, and compile it to make sure things are set up properly before you do the more complicated parts.

4. Compile using `javac`. The command `javac Nonogram.java` should compile your program.

5. Run using `java Nonogram`. If you have kept the `main` method in its original form, it should bring up a GUI window with a puzzle in it. (Not much will happen until you flesh out your implementation, of course.)