Chris Piech
CS 106A

Section #6
February 18, 2019

Section Handout #6: Arrays and HashMaps

Portions of this handout by Eric Roberts and Marty Stepp

1D Arrays

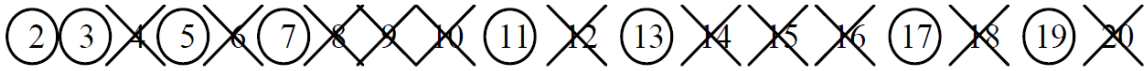
1. How Prime!

In the third century B.C., the Greek astronomer Eratosthenes developed an algorithm for finding all the prime numbers up to some upper limit *N*. To apply the algorithm, you start by writing down a list of the integers between 2 and *N*. For example, if *N* were 20, you would begin by writing down the following list:

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

You then begin by circling the first number in the list, indicating that you have found a prime. You then go through the rest of the list and cross off every multiple of the value you have just circled, since none of those multiples can be prime. Thus, after executing the first step of the algorithm, you will have circled the number 2 and crossed off every multiple of two, as follows:

From here, you simply repeat the process by circling the first number in the list that is neither crossed off nor circled, and then crossing off its multiples. Eventually, every number in the list will either be circled or crossed out, as shown in this diagram:



The circled numbers are the primes; the crossed-out numbers are composites. This algorithm for generating a list of primes is called the sieve of Eratosthenes. Write a program that uses the sieve of Eratosthenes to generate a list of all prime numbers between 2 and 1000.

2. Trace

Write the final array contents when the following method is passed each array below:

```
private void mystery(int[] nums) {
    for (int i = 0; i < nums.length - 1; i++) {
        if (nums[i] > nums[i + 1]) {
            nums[i + 1]++;
        }
    }
}
```

```
Array 1: {10, 8, 9, 5, 5} Array 2: {12, 11, 10, 10, 8, 7}
```

3. Switch Pairs

Write a method **`switchPairs`** that accepts as a parameter an array of strings and returns a new array with the order of values switched in a pairwise fashion. In other words, your method should return an array with the same elements as the original, but with the first two values switched, the second two values switched, and so on. For example, if the following array named `arr` was passed to your method:

```
["four", "score", "and", "seven", "years", "ago"]
```

Your method should return a new array with the first pair, "four" and "score", switched, the second pair, "and" and "seven", switched, and the third pair, "years", "ago", switched. So the call of **`switchPairs(arr);`** would return this array:

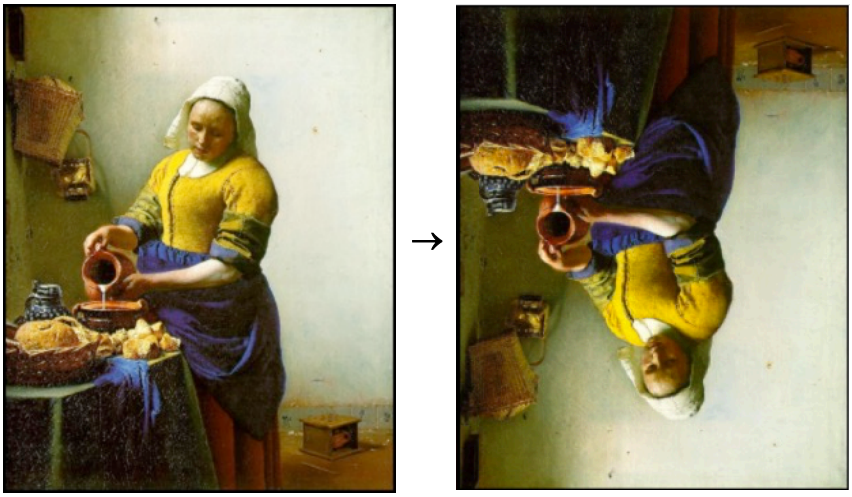
```
["score", "four", "seven", "and", "ago", "years"]
```

If there are an odd number of values, the final element should not be changed.

2D Arrays

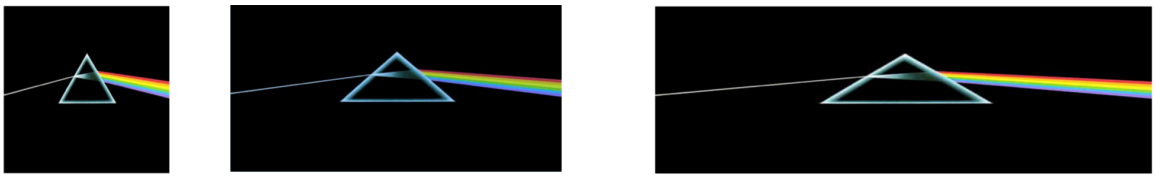
4. Flip Vertical

Write a method **`flipVertical`** that reverses a picture in the vertical dimension. Thus, if you had a **`GImage`** containing the image on the left (of Jan Vermeer's *The Milkmaid,* c. 1659), calling **`flipVertical`** on it would return a new **`GImage`** as shown on the right:



5. Stretch

Write a method stretch that takes a GImage and an int representing a scale factor as parameters, and modifies the image to horizontally stretch it by the given factor. For example, if the factor is 2, stretch the image to be twice as wide, or if the factor is 3, stretch it to be 3x as wide. As an example, here's a before-and-after comparison of the cover of Pink Floyd's "The Dark Side of the Moon" stretched by 2x and 3x:



6. Trace

Consider the following method.

```
private void mystery2D(int[][] numbers) {
    for (int r = 0; r < numbers.length; r++) {
        for (int c = 0; c < numbers[0].length - 1; c++) {
            if (numbers[r][c + 1] > numbers[r][c]) {
                numbers[r][c] = numbers[r][c + 1];
            }
        }
    }
}
```

If a two-dimensional array `numbers` is initialized to:

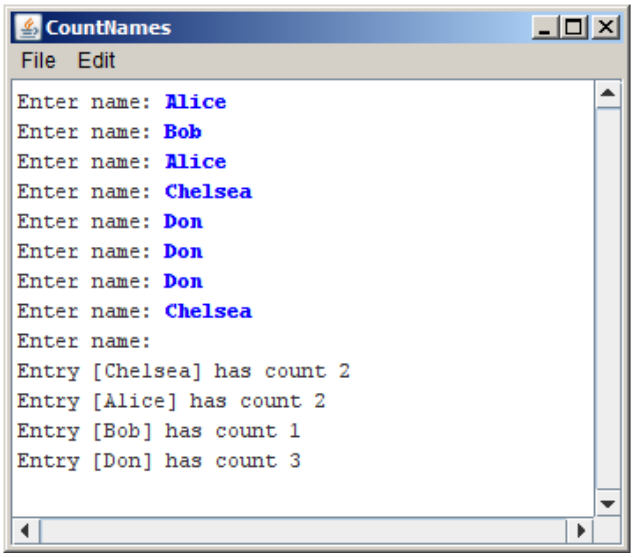
```
3 4 5 6
4 5 6 7
5 6 7 8
```

...what are its contents after the call of `mystery(numbers);` ?

HashMaps

7. Name Counts

Write a program that asks the user for a list of names (one per line) until the user enters a blank line (i.e., just hits return when asked for a name). At that point the program should print out ***how many times*** each name in the list was entered. A sample run of this program is shown below.



8. Mutual Friends

In the days before Facebook, one of the easier ways to find mutual friends was to compare address books and find common entries. Write a method named **`mutualFriends`** that accepts as parameters two **`HashMap`**s from strings to integers representing two phonebooks, and returns a new map containing only the key/value pairs that exist in both of the phonebooks. Remember that for an entry to be included in your result map, not only do both maps need to contain that key, but they need to map that key to the **same value** (phone number). For example, consider the following two maps:

```
{Jenny=8675309,  Julia=2124320,  Chris=4602121,  Annie=4444444,
Brahm=8080543}

{Logan=6202121,  Jeff=8888888,  Brahm=8080543,  Chris=4602121,
Annie=4444444, Jenny=2128765}
```

Calling your method on the preceding maps would return the following new map (the order of the key/value pairs does not matter):

```
{Brahm=8080543, Chris=4602121, Annie=4444444}
```

Notice how even though the key `Jenny` is present in both maps, it is not included in our final result map because it maps to a different phone number in each phonebook.