Chris Piech
CS106A

Handout #extra
March 18, 2017

Additional Practice Problems

Some of these problems are courtesy of Julia Daniel

Short Problems

1. Read integers in the console from a user until the user enters a blank like. For each print "EVEN" if the integer is even and "ODD" if the integer is odd.

2. Write a double for loop to print out a rectangle of "0"s. The rectangle should have LINES number of strings and COLS number of "0"s per line (assume they are defined as constants). If LINES = 3 and COLS = 4 the following strings should be printed to the console.

```
0000
0000
0000
```

3. Populate an array of size 10 with random booleans.

4. Populate a grid with 3 rows and 2 colums where each value of the grid has a value equal to 2.

5. Populate a grid with 3 rows and 2 colums where each value of the grid has a value equal to its row index plus its col index.

6. Write a method

```
private void printMatrix()
```

that prints the contents of a grid to the screen (in any reasonable format).

7. Open a file named "doubles.txt" with 10 lines, one double per line and store each double in an array.

8. Write a `GraphicsProgram` such that when a mouse is clicked, you add a single 10 x 10-pixel rectangle with upper left corner in the place the mouse was clicked.

9. Make a hashmap that associates strings to integers and put in the following entries which associates universities and their endowment (in billions of dollars):

```
"Stanford" -> 22.4
"Berkeley" -> 4.0
"Harvard" -> 35.0
```

Then write a loop that can iterate over all of the key/value pairs and print them out. Any order / format is fine.

10. Put 10 random Integer pairs (both the key and the value are in the range 0, 100) in a hashmap. Then print out all the values whose keys are even.

11. Write a method:

```
   String addQuotation(String str)
```

Which takes in an input string and add quotation marks (") to the start and end of the string. Return the result.

11. Use a loop to create a string that contains all of the capital letters in the alphabet, eg: "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

12. Write a method:

```
   String makeCamelCase(String input)
```

That, returns the input in camelcase. The input is assumed to be all lowercase. Whenever the input has a space, remove the space and make the next letter uppercase. For example:
If the input was "**`this is a test`**", you would return "**`thisIsATest`**".

13. Write a ConsoleProgram that has two buttons one which says “yes” and one which says “no”. Each time a user clicks on a button you should printout whether or not the “yes” button has been pressed more, the “no” button has been pressed more, or whether they have been pressed an equal number of times.

13. Write a variable type **`IdCounter`** which has a single public method `getNextId`. `getNextId` should return an integer, and should never return the same integer twice. For example, a user of the **`IdCounter`** class might do something like this:

```
   IdCounter idCounter = new IdCounter ();
   int idForChris = idCounter.getNextId();  // can return any id for chris
   int idForNick = idCounter.getNextId();   // should return a different id
```

For more problems, look at the end of each chapter, work through the section problems, look at the problems on codingbat.com

Long Problems

Problem 1: Strings (15 Points)

An *isogram* is a word that contains no repeated letters. For example, the word “computer” is an isogram because each letter in the word appears exactly once, but the word “banana” is not because 'a' and 'n' appear three times each. “Isogram” is itself an isogram, but “isograms” is not because there are two copies of 's'.

There are many long isograms in English; for example, “uncopyrightable” and “computerizably.” Your job is to write a method that, given a list of all the words in the English language, finds out what the longest isogram actually is. Write a method

```
        private String longestIsogram(ArrayList<String> allWords)
```

that accepts as input a `ArrayList<String>` containing all words in English (stored in lower-case) and returns the longest isogram in the list. If multiple words are tied as the longest isogram, feel free to return any one of them.

```
private String longestIsogram(ArrayList<String> allWords) {
```

Problem 2: Random Numbers (25 Points)

Suppose you want to hold a never-ending birthday party, where every day of the year someone at the party has a birthday. How many people do you need to get together to have such a party?

Your task in this program is to write a program that simulates building a group of people one person at a time. Each person is presumed to have a birthday that is randomly chosen from all possible birthdays. Once it becomes the case that each day of the year, someone in your group has a birthday, your program should print out how many people are in the group.

In writing your solution, you should assume the following:

- There are 366 possible birthdays (this includes February 29).
- All birthdays are equally likely, including February 29.

You might find it useful to represent birthdays as integers between 0 and 365, inclusive.

```
public class NeverendingBirthdayParty extends ConsoleProgram {
```

Problem 3: Arrays (25 points)

A *magic square* is an $n \times n$ grid of numbers with the following properties:

1. Each of the numbers 1, 2, 3, …, n^2 appears exactly once, and
2. The sum of each row and column is the same.

For example, here is a 3×3 magic square, which uses the numbers between 1 and 9:

4	9	2
3	5	7
8	1	6

and here is a 5×5 magic square, which uses the numbers between 1 and 25:

11	18	25	2	9
10	12	19	21	3
4	6	13	20	22
23	5	7	14	16
17	24	1	8	15

Write a method

```
private boolean isMagicSquare(int[][] square, int n);
```

that accepts as input a two-dimensional array of integers (which you can assume is of size $n \times n$) and returns whether or not it is a magic square.