CS 1331: Introduction to Object Oriented Programming

Homework 5 **Due:** February 21, 2014 @ 8:00 PM

Family Planning

How many children must a couple have in order to make sure they have at least one boy and one girl? In this program, you will develop a simulation that will help you estimate the answer to that question. Your objective is to model a couple that continues to have children and only stops once they have at least one boy and one girl. Assume that for any particular childbirth, the chances of having a boy or a girl are even (50% chance of a boy, 50% chance of a girl).

More specifically, write a program that prompts the user for the number of couples in the simulation. The program then should simulate each couple's child birth pattern, printing out the order of boys and girls born. Once the requisite number of families has been simulated, print out both the average and maximum number of children necessary to achieve the problem constraints. Also print the total number of boys and girls born.

After that, prompt the user whether they want to run another simulation and continue in the same fashion until the user decides to quit.

```
Ready to run a family simulation. Enter the number of families:
4
Simulating Families
1 - BBG
2 - GB
3 - GGGB
4 - BG
The average number of children was 2.75 and maximum was 4.
A total of 5 boys and 6 girls were born.

Would you like to run another simulation? (y/n)
y
Ready to run a family simulation. Enter the number of families:
25
1 - GGB
2 - BG
...
```

Notes and Hints

- You can assume that the user will only enter an integer for the number of families. You should check if the integer is invalid (I.E. negative) and re-prompt the user for a valid number.
- Do NOT code this entire program in main(). There is a Code Structure section below, but you'll notice looser constraints this time around.
- Try a big number like 100 and see what your answers are!

Code Structure

We are going to give some more basic guidelines for this problem, keep in mind we are giving you less structure this time. You should be able to start designing these programs in a smart way on your own now (smart Object-Oriented design will be part of your grade for these assignments)!

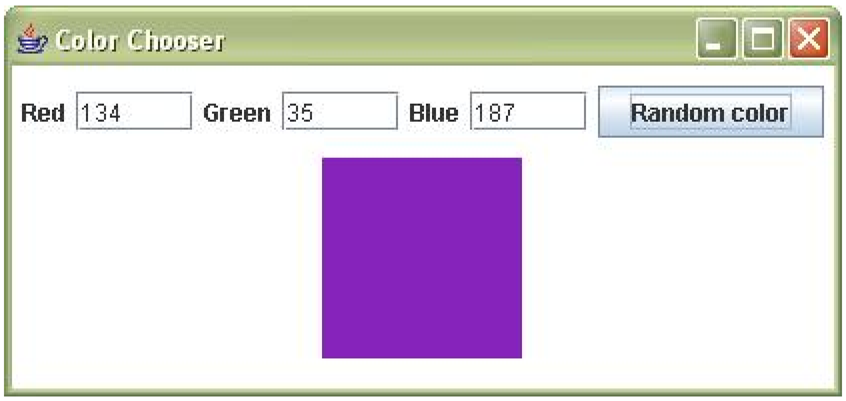
- A class called `Family`
  - This class should have some way to have a child (it is up to you as to how you want to do this!)
  - It should also have a way to keep track of how many boys, girls, and total children have been born
- A class called `FamilyPlanning` that will contain the main method to run the program
  - This class should make `Family` objects and have them have kids until a boy and a girl have been born

Color Chooser

In this assignment you will build a simple graphical application that helps a person choose a color. Your program will have three text entry boxes, one each for the red, green, and blue components of a color. It will also have a rectangular region that is used to display the currently specified color.

A valid value in each box is from 0 to 255. When the user types a return in any of the boxes, you should update the color shown in the box. Additionally, your program should have a button that allows the user to make a random choice. When this button is clicked, you should generate random values from 0 to 255 for each color component, change the text boxes, and then display the new color.

An example view of what your program might look like is shown below, but you should feel free to design your own user interface.



We aren't going to give you any code structure for this problem, but again, you should be able to make smart programming design decisions at this point in the class!

Javadocs

We are going to have you do Javadocs again for this assignment (and for all assignments here on out). The samples from the last HW writeup are below for your reference.

```
import java.util.Scanner;

/**
* This class represents a Dog object.
* @author Ethan Shernan
* @version 1.0
*/

public class Dog {
....
}

/**
* This method takes in two ints and returns their sum
* @param a, b
* @return their sum
*/

public int add(int a, int b)){
...
}
```

Checkstyle

Just in case you forget how to do Checkstyle, here are the instructions again!

You have been provided with a Checkstyle Jar file and the CS 1331 Checkstyle configuration file (`cs1331-checkstyle.xml`). We do this to make sure you are following the proper style guidelines as posted in the Style Guide under T-Square resources. Please refer to these guidelines for any help as you are writing your programs.

We will run checkstyle on all the Java source files you submit. You can run Checkstyle with the command below (in the directory containing all your Java source files):

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. You can easily count Checkstyle errors by using the command below and subtracting 2 from the number printed (which is how we will deduct points). For example:

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | wc -l
2
```

Also means no errors.

If you have any questions about a Checkstyle error, please let a TA know! They should be pretty self-explanatory and you should be able to fix them easily. We will be taking points off for checkstyle on this assignment, so please make sure you run it!

Turn-in Procedure

Submit all of the Java source files you created to T-Square. Do not submit any compiled bytecode (`.class` files). When you're ready, double-check that you have submitted and not just saved a draft.

Please remember to run your code with Checkstyle!

Verify the Success of Your Submission to T-Square
Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.

2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.

3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.

4. Recompile and test those exact files.

5. This helps guard against a few things.

   (a) It helps insure that you turn in the correct files.

   (b) It helps you realize if you omit a file or files.[1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)

   (c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is 8PM Friday. Do not wait until the last minute!