CMPSC 111
Introduction to Computer Science I
Fall 2014

Lab 8 for Sections 01 and 02
30 October 2014
Due Thursday, 6 November by 2:30pm

Objectives

To inhence your experience with creating your own classes, including instance variables, constructors, and methods. To practice using compound `if/else` statements as a solution to a relatively large program.

General Guidelines for Labs

- **Work on the Alden Hall computers.** If you want to work on a different machine, be sure to transfer your programs to the Alden machines and re-run them before submitting.
- **Update your repository often!** You should add, commit, and push your updated files each time you work on them. I will not grade your programs until the due date has passed.
- **Review the Honor Code policy.** You may discuss programs with others, but programs that are nearly identical to others will be taken as evidence of violating the Honor Code.

Reading Assignment

To learn more about `if` statement and boolean expressions, review Sections 5.1–5.3, and to review material on constructing classes and methods, read Sections 4.1–4.5.

Create a new directory and two Java programs

In your own `cs111F2014-<your user name>` repository inside `labs` directory, create a directory called `lab8`. Type "`cd lab8`" to move to the new directory. Then using `gvim` create a `Lab8.java` and `Lab8Main.java` files.

Sudoku Checker

Sudoku is a logic-based placement puzzle. The aim of this puzzle is to enter a numerical digit from 1 through 9 in each cell of a 99 grid made up of 33 subgrids (called "regions"), starting with various digits given in some cells (the "givens"). Each row, column, and region must contain only one instance of each numeral.

For this lab, you are going to write a Sudoku validator for 4x4 grids made up of 2x2 regions instead of 9x9 grids made up of 3x3 regions. Sudoku puzzles that use 4x4 grids only use the numerical digits 1 through 4 instead of 1 through 9.

Here are two correct 4x4 Sudoku grids:

3	2	4	1
4	1	3	2
1	4	2	3
2	3	1	4

1	2	3	4
3	4	1	2
2	1	4	3
4	3	2	1

Notice how each row, column and region has the individual numbers from 1 through 4 used only once. This means that the values used in each row, column and region must add up to 10 (1 + 2 + 3 + 4). This is how we will check our 4x4 Sudoku grids for this assignment.

Allow your user to enter their Sudoku grids row-by-row, separating each of the four values by a space and hitting ENTER at the end of the row.

When the user has entered all 16 values into the program, you should output validation checks for each region, row and column. An example run can be seen below.

```
Welcome to the Sudoku Checker v1.0!

This program checks simple, small, 4x4 Sudoku grids for correctness.
Each column, row and 2x2 region contains the numbers 1 through 4 only once.

To check your Sudoku, enter your board one row at a time,
with each digit separated by a space.  Hit ENTER at the end of a row.

Enter Row 1: 3 2 4 1
Enter Row 2: 4 1 3 2
Enter Row 3: 1 4 2 3
Enter Row 4: 2 3 1 4

REG-1:GOOD
REG-2:GOOD
REG-3:GOOD
REG-4:GOOD

ROW-1:GOOD
ROW-2:GOOD
ROW-3:GOOD
ROW-4:GOOD

COL-1:GOOD
COL-2:GOOD
COL-3:GOOD
COL-4:GOOD

SUDO:VALID
```

Your program does not have to check for the numbers 1 through 4 being used uniquely in each row, column and region. In other words, don't worry about validating their input to make sure they

only enter the digits 1, 2, 3 or 4 and only enter them once per row, column and region. You simply need to check that each row, column and region adds up to 10. This simplistic type of checking will, however, allow some bad Sudoku grids to be validated as good. This is ok.

Rows, columns and regions are identified as found below:

ROW1
ROW2
ROW3
ROW4

COL 1	COL 2	COL 3	COL 4

REGION 1	REGION 3
REGION 2	REGION 4

`Lab8Main` Class

To test your implementation of the `Lab8` class, use the following code for Lab8Main.java.

```
public class Lab8Main
{
     public static void main ( String args[] )
     {
          Lab8 foo = new Lab8();
          foo.getGrid();
          foo.checkGrid();
     }
}
```

`Lab 8` Class

The basic structure of `Lab8.java` is as follows:

```
import java.util.Scanner;

public class Lab8
{

     // put private data members here

     // put constructor here

     // put getGrid() here

     // put checkGrid() here
}
```

The UML diagram for `Lab8` is:

Lab8
- w1 : Integer - w2 : Integer - w3 : Integer - w4 : Integer - x1 : Integer - x2 : Integer - x3 : Integer - x4 : Integer - y1 : Integer - y2 : Integer - y3 : Integer - y4 : Integer - z1 : Integer - z2 : Integer - z3 : Integer - z4 : Integer
<< *constructor* >> Lab8 ( )
+ getGrid ( ) + checkGrid ( )

What follows is a short description of what each data member represents and what each method does:

`w1, w2, w3, w4`
`x1, x2, x3, x4`
`y1, y2, y3, y4`
`z1, z2, z3, z4`
	Private data members that will store the numbers the user inputs. `w1` through `w4` are for the first row, `x1` through `x4` are for the second row, etc. Do not have any other private data members in your class.

Lab8()	This is the constructor. It should display welcome greetings, explaining the rules of the 4x4 Sudoku grid we're processing.
getGrid()	This public method should read the Sudoku grid from the user row by row, using spaces between values in each row.
checkGrid()	This public method uses the private data members to test whether or not the given values are a valid Sudoku grid. Your method does not have to check for the numbers 1 through 4 being used uniquely in each row, column and region. In other words, don't worry about validating their input to make sure they only enter the digits 1, 2, 3 or 4 and only enter them once per row, column and region. When producing your output, you must first validate the regions, then the rows and then the columns. Each region, row and column validation should appear on a line by itself.

Points to Think About

Since we have not talked about programming concepts such as arrays yet (you can not use them), you will need to input your 16 values into 16 separate variables. Some suggestions would be a1, a2, a3 and a4 for the first row and then b1, b2, b3 and b4 for the second row, and so on. Once you validate that one row is good or not good, the rest of the program should essentially be a matter of copy and paste, replacing variables where appropriate. The "toughest" part of this program is determining whether or not you have a good Sudoku or not. One suggestion would be to keep track of a variable the counts the number of things that are invalid and if that variable's value is 0 at the end of the program, you have a valid Sudoku.

Additional Program Requirements

- Make sure your program prints your name, the lab number, and the date.
- Make sure your program contains the comment header with the honor pledge, your name, lab number, date, and the purpose of the program.
- Make sure your program is documented properly, with the comments throughout your program whenever appropriate.
- Make sure your output is neat (no missing spaces, no typos, etc.) and that your program is neat (indenting, etc.).
- **You may not use array structures or loops for this assignment.**

Required Deliverables

For this assignment you are invited to submit electronic versions of the following deliverables through the Bitbucket repository.

1. A completed, properly commented and formatted `Lab8.java` and `Lab8Main.java` programs.
2. An output document containing an output obtained after running `Lab8Main.java` program.

As you complete this step, you should make sure that you created a `lab8/` directory within the Git repository. Then, you can save all of the required deliverables in the `lab8/` directory—please see the course instructor or a teaching assistant if you are not able to create your directory properly.

Share your program and the output file with me through your Git repository by correctly using "`git add`", "`git commit`", and "`git push`" commands. When you are done, please ensure that the Bitbucket Web site has a `lab8/` directory in your repository with the two files called `Lab8.java`, `Lab8Main.java` and `output`. You should see the instructor if you have questions about assignment submission.