COMP 1020
Lab 4

MATERIAL COVERED

- Multidimensional arrays

Notes:

- Make sure your TA has recorded your mark before leaving.
- The three exercises are independent – you can do them in any order.
- Only one of the three exercises is required. The Gold exercise is not much more difficult than the Silver exercise this time (some students may actually find it easier).

A RandomArray class

1. Download the file **`RandomArrayTemplate.java`** and save it as **`RandomArray.java`** before using it. The supplied **`TestLab4Bronze.java`** program will expect it to be named this way.

2. This class creates rectangular two-dimensional arrays filled with random positive integer values. Some of the class is written for you. Complete it by adding code in the four places marked **`//***ADD YOUR CODE HERE***.`**

    a. Add a private instance variable that will hold a 2-dimensional array of integers.

    b. Complete the constructor, which will create the 2-dimensional array, with the specified number of rows and columns, containing values from **`0`** to **`range-1`**. The code to generate such random numbers is supplied in the comments.

    c. Complete the **`getRow`** method, which will return a copy (clone or deep copy) of the specified row of the array, as an array of integers of the correct length. In this method, use the built-in **`System.arraycopy`** method to copy the values from the original array into the result, as demonstrated in class.

    d. Complete the **`getCol`** method, which will return a copy (clone or deep copy) of the specified column of the array, as an array of integers of the correct length. In this method, you will have to copy the values yourself using a **`for`** loop since the **`System.arraycopy`** method cannot be used in this situation.

3. Run the supplied **`TestLab4Bronze.java`** program to test your class. Sample output from this program is shown below. User input is in blue. Your numbers will be different.

```
How many rows? 3
How many columns? 4
The rows contain:
Row 0: [15, 92, 48, 87]
Row 1: [74, 32, 24, 10]
Row 2: [55, 0, 50, 14]

The columns contain:
Column 0: [15, 74, 55]
Column 1: [92, 32, 0]
Column 2: [48, 24, 50]
Column 3: [87, 10, 14]
```

The MagicSquare class

---

1. Download the file **`MagicSquareTemplate.java`** and save it as **`MagicSquare.java`** before using it. The supplied **`TestLab4Silver.java`** program will expect it to be named this way.
2. This class creates a square two-dimensional array which is a "magic square". A magic square of "order n" is a square with n rows and n columns that contains each of the numbers from 1 to n^2. The sum of every row, every column, and the two diagonals, must be the same. For example, an order 5 magic square is shown below. Every row, column, and diagonal adds up to the same thing (65). Such squares are very old, with early examples dating back to 2200BC.

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

3. Some of the class is written for you. Complete it by adding code in the three places marked **`//***YOUR CODE HERE***.`**
    a. Add a private instance variable that will hold the magic square as a 2-dimensional array of integers.
    b. Complete the constructor, which will create a magic square with a given size (n), which must be an *odd number*. (You don't have to check.) To do this, use a very old technique called "de la Loubere's algorithm" (note that this will give a different square from the one shown above):
        i. Place the numbers in the square in order, from 1 to n^2, starting with the 1 in the centre of the bottom row.
        ii. If the number you just placed is divisible by n, then the next one goes immediately above it.
        iii. If the number you just placed is *not* divisible by n, then the next one goes diagonally down and to the right.
        iv. Both the rows and the columns are cyclic. If you go off the top of the square, re-enter at the bottom. If you go off the bottom of the square, re-enter at the top. Do the same for the columns. **Hint**: This can easily be accomplished using the mod operator (%).
        v. See the sample output below for examples of squares constructed this way.
    c. Complete the **`toString`** method, which will return the proper **`String`** to use to print the magic square. It should contain tabs (**`\t`**) between the columns and newlines (**`\n`**) between the rows.

4. Run the supplied **`TestLab4Silver.java`** program to test your class. Sample output from this program is shown below. User input is in blue.

```
Enter a small odd number, or 0 to quit:3

The magic square is:
4     9     2
3     5     7
8     1     6

Enter a small odd number, or 0 to quit:7

The magic square is:
22    31    40    49    2     11    20
21    23    32    41    43    3     12
13    15    24    33    42    44    4
5     14    16    25    34    36    45
46    6     8     17    26    35    37
38    47    7     9     18    27    29
30    39    48    1     10    19    28

Enter a small odd number, or 0 to quit:0
That was fun. Bye!
```

The CrosswordGrid class

---

1. Download the file **`CrosswordGridTemplate.java`** and save it as **`CrosswordGrid.java`** before using it. The supplied **`TestLab4Gold.java`** program will expect it to be named this way.
2. This class creates and displays a blank crossword puzzle grid. All that's missing are the clues. A crossword puzzle grid consists of a rectangular (usually square) array of squares. Some squares are black, some are white, and some of the white ones have numbers in the corner. An integer array will be used to represent this, as follows:
    a. Black squares will be represented by -1.
    b. White squares with no number in them will be represented by 0.
    c. White squares that contain a number will be represented by that number.

    A file will be provided that will specify the size of the puzzle, and the pattern of black and white squares. It will contain the number of rows (r), the number of columns (c), and then r*c values which will be only -1 (black) or 0 (white). For example, the file might contain

```
 4  4
-1  0  0 -1
 0  0  0  0
 0  0  0  0
-1  0  0 -1
```

    Your mission, should you choose to accept it, is to read in this file, create the array containing the -1s and 0s, and then add the clue numbers in the appropriate squares (as described below). This will require about 20 lines of Java code in total.

3. A large amount of code is supplied for you, including all of the file handling and the graphical output. Complete it by adding code in two places marked **`//***ADD YOUR CODE HERE***.`**

    a. Complete the constructor. A **`Scanner`** object has been set up for you that will allow you to read data from a text file chosen by the user using a standard dialog box. (Take a look at the supplied code to see how this is done if you like.) You can read from **`inFile`** in exactly the same way that you usually read from **`keyboard`**. Read in the values from the file and set up the pre-defined **`grid`** array.

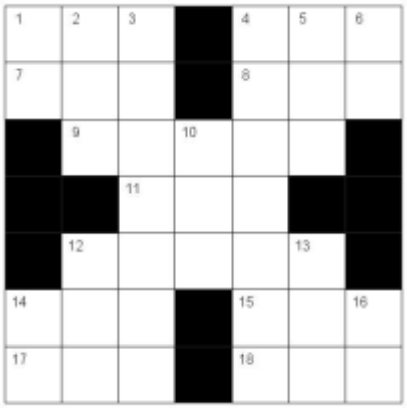
    b. Complete the **`putInNumbers`** method, which will change some of the 0's in the array to positive numbers instead. The numbers should start at 1 in the top left, and increase going across the rows, and down the columns, as shown in the example below. The rules for placing numbers are:

        i. An empty square should get a number *if* there is a black square immediately above it, *or* it is in the top row.

        ii. An empty square should get a number *if* there is a black square immediately to the left of it, *or* it is in the leftmost column.

        iii. Note that a square might get a number for both of these reasons.

4. Download the file **`CrosswordGridPattern.txt`**. Also download and compile the **`StdDraw.java`** file and the **`TestLab4Gold.java`** file. Make sure all the files are in the same folder. Run the **`TestLab4Gold`** program. A dialog box will appear asking you to choose a file. Choose the file **`CrosswordGridPattern.txt`**. A graphics window should appear containing the crossword puzzle grid shown below.



Just for fun: You can use the Save… command in the graphics window to save this file as a .jpg image. Print it out and solve the puzzle using these clues:

Across
1: And so on…
4: Abbreviation for 4 down
7: Prefix for "new"
8: Trig. function
9: Sphere of influence
11: A dessert
12: A type of pet
14: ___ serious!
15: Feline
17: Unity
18: Punk offshoot

Down
1: A small dash
2: A drink from Tim's
3: What computers do
4: A Faculty at the U of M
5: A small bed
6: But __ it art?
10: Base 2 (abbrev.)
12: Big ___
13: Sweet potato
14: Board game
16: Get back __ work!