Deadline: Sunday November 9, at 11:59pm.
Evaluation: 6 pts of your final grade
Late Submission: 20% off per day
Teams: The assignment can be done individually or in teams of 2 (from the same section) (submit only one assignment per team)

Purpose: The purpose of this assignment is to help you practice arrays.

Game of Ognib

Write a Java program to simulate a 2-player game called Ognib. In Ognib, each player is given a 5x5 grid where they place numbered tokens face up. The tokens are numbered with an integer ranging from 1 to 100 inclusively, are all distinct and are sorted in increasing order. Players cannot see their opponent's grid – only their own. One at a time, each player "calls" an integer and if any player has this token on their grid, they turn the token face down on their grid. The goal of the game is for a player to have all tokens in a row, or in a column or in any of the 2 diagonals facing down before the opponent does. The first player who achieves such a configuration wins the game, and is allowed to brag about it on his/her Facebook page.

To better understand the rules of the game, here is an illustration of a game of Ognib between Fred & Wilma.

1. Initially, each player is given a grid of 25 random integers between 1 and 100 (all distinct and sorted is ascending order).

Fred's grid:

4	10	12	20	35
37	40	43	47	50
51	57	64	67	73
80	86	87	88	90
92	95	97	99	100

Wilma's grid:

2	6	9	11	15
22	23	27	35	40
42	47	51	54	59
66	68	69	74	76
81	84	87	91	93

2. Fred calls token 4. If they have it on their grid, both Fred and Wilma turn token 4 face down (we can represent this by replacing it with a zero.)

Fred's grid:

0	10	12	20	35
37	40	43	47	50
51	57	64	67	73
80	86	87	88	90
92	95	97	99	100

Wilma's grid:

2	6	9	11	15
22	23	27	35	40
42	47	51	54	59
66	68	69	74	76
81	84	87	91	93

3. Wilma calls token 51. Each player turn token 51 face down. (What Wilma does not know is that she just helped Fred in his game because he had token 51 on his grid conveniently located in the same column where he already had a token facing down...)

Fred's grid:

0	10	12	20	35
37	40	43	47	50
0	57	64	67	73
80	86	87	88	90
92	95	97	99	100

Wilma's grid:

2	6	9	11	15
22	23	27	35	40
42	47	**0**	54	59
66	68	69	74	76
81	84	87	91	93

And so forth... until either Fred or Wilma (or both) has turned all tokens in a row, column or any of the two main diagonals face down. For example, if we arrive at the following configuration, then Fred wins the game:

Fred's grid:

0	10	12	20	35
0	40	43	47	50
0	57	64	**0**	73
0	86	87	88	90
0	95	**0**	99	100

Wilma's grid:

2	6	9	11	15
0	23	27	35	40
42	47	**0**	**0**	59
66	68	**0**	74	76
81	84	87	91	93

If both players arrive at a winning configuration at the same turn, then there is a tie.

Write a Java program to play Ognib against the computer. The 2 grids (yours and the computer's) must be represented by a one-dimensional array of 25 integers. For example, Fred's initial grid would be represented as:

4	10	12	20	35	37	40	43	47	50	51	57	65	67	73	80	86	87	88	90	92	95	97	99	100

Your program must first fill in the two arrays (representing the two grids) with random integers between 1 and 100 inclusively. Each grid cannot contain duplicate values and must be sorted in ascending order.

Once the grids are initialised, we can start playing:

1. The computer will be the first to call a token. To choose a token, the computer will simply select an integer at random within the tokens on its own grid that have not been called yet.
2. Your program will then turn this token face down on both grids.
3. Then your program will evaluate both grids to see if there is a winner (or a tie). If this is the case, then your program must announce the winner(s) on the screen, and stop the game.
4. If there is no winner yet, then a new token will be called (by you or by the computer), and your program will turn the tokens face down and check again for a potential winner, ... until the game ends.

Because the computer's strategy to pick tokens will probably not be as sophisticated as yours[1], to give it a fair chance at winning, the computer will pick a token 3 times in a row, then you will pick a token once, then the computer will pick three times again, then you will pick only once...

After each the computer has called its 3 tokens and you have called yours, your program must display on the screen:

- the name of who called the token (you or the computer) and the value called
- your grid only (in 5 x 5 format) where all tokens that are face down are shown by a zero

At the end of the game, your program must display:

- the name of the winner (or "tie")
- the final grid of the computer (with the tokens called replaced by zero)

Notes:

1. To better understand the rules of the game, you can consult the files `game1.txt`, `game2.txt` and `game3.txt` available on Moodle.
2. On Moodle you will also find an example of code to initialise an array with distinct random integers between 1 and 100. Feel free to re-use this code in your program.
3. You must use a one-dimensional array to represent the grids – not a two-dimensional array.

Have fun!

[1] Remember that the computer picks a number at random within all tokens that are still face up its grid – without trying to optimise a specific row, column or diagonal.

Evaluation Criteria (on 100):

Comments: 5 pts
- Description of the program (authors, date, purpose) . . 2 pts
- Description of variables and constants . . . . . 1 pt
- Description of the algorithm . . . . . 2 pts

Programming style: 25 pts
- Use of constants where appropriate . . . . . 2 pts
- Use of significant names for identifiers . . . . . 2 pts
- Indentation and readability . . . . . 4 pts
- Simplicity of the algorithm . . . . . 12 pts

Functionality: 75pts
- Creation of the grids . . . . . 12 pts
- Calling a token . . . . . 12 pts
- Replacing tokens by zero . . . . . 12 pts
- Checking/displaying winner . . . . . 12 pts
- Displaying output . . . . . 12 pts
- Miscellaneous . . . . . 15 pts

Submission

When you are finished the program, you must submit it online.

1) Create **one** zip file, containing the source file (the .java file).

Please name your file following this convention:

- If the work is done by 1 student, your file should be called *a#_studentID*, where # is the number of the assignment *studentID* is your student ID number. For example, for the first assignment, student 123456 should submit:
  `assignment1_123456.zip`

- If the work is done by 2 students: The zip file should be called *a#_studentID1_studentID2*, where # is the number of the assignment *studentID1* and *studentID2* are the student ID numbers of each student, for example,
  `assignment1_123456_987654.zip`
  Only one member of the team needs to upload the file.

2) Upload your zip file at the URL: https://fis.encs.concordia.ca/eas/ as ***Programming Assignment 3***