Concordia University
Computer Science & Software Engineering
Faculty of Engineering and Computer Science

Concordia University
COMP 248 – Fall 2015
Assignment 3

Due Date:	By 11:55pm Wednesday November 11, 2015
Evaluation:	4% of final mark (see marking rubric at the end of handout)
Late Submission:	none accepted
Purpose:	The purpose of this assignment is to help you learn the Java flow of control statements (nested loops), 2 dimensional arrays, and an introduction to classes.
CEAB/CIPS Attributes:	Design/Use of Engineering tools/Communication Skills

General Guidelines When Writing Programs:
Please refer to the handout of Assignment #1.

Question 1 (15 Points) - Arrays/Nested loops – Game of TicTacToe
Write a Java program that will allow two users to play Tic-Tac-Toe. If you are not familiar with the game refer to the following website for a description: http://en.wikipedia.org/wiki/Tic-tac-toe.

Your program should loop as long as the players want to play. Your program keeps track of the total number of games played, as well as the number of games won by each player.

The program should:

1) At the beginning of a game display a clean board and prompt player X for the position (from 1 to 9) s/he wishes to mark. The program should make sure that the position entered is a valid position (between 1 and 9). If it is not, the program should prompt the user for a position number until a valid one has been entered.

```
1 2 3
4 5 6
7 8 9
Player X – Enter the position number you wish to mark:
```

2) After each move the program displays the changed board and whether there is a winner. "The player who succeeds in placing three respective marks in a horizontal, vertical or diagonal row wins the game" (http://en.wikipedia.org/wiki/Tic-tac-toe). If there is no winner yet, prompt the next player for his/her position. The program should make sure that the position entered is a valid position (between 1 and 9) and that it is unmarked. If it is not, the program should prompt the user for a position number until a valid one has been entered. This continues until there is a winner or the board is full.

```
O 2 X
4 5 6
7 8 9
There is no winner yet!
Player O – Enter the position number you wish to mark: 1
That position is not available
Player O – Enter the position number you wish to mark: 0
That is not a valid position – must be between 1 and 9 inclusive
Player O – Enter the position number you wish to mark:
```

3) If there is a winner, the program congratulates the winner and reports the total number of games played to date as well as the number of games each player has won:

```
O 2 X
4 X 6
X 8 O
--> We have a winner! Congratulations Player X <--
Games played to date 6;
Player X has won 1 game(s); Player O has won 2 game(s).
```

4) If the board is full with no winner, the program sends a message indicating that it is a tie game as well as the total number of games played to date and the number of games won by each player:

```
O O X
X O X
O X O
This game is a tie
Games played to date 7;
Player X has won 1 game; Player O has won 2 games.
```

5) When a game is over, the program asks if the players wish to play another game.
   a) if yes, the board is reset and a new game is played (back to Step 1)
   b) if no, your program should display the total number of games played, the number of games won by each player, the percentage of games won by each player and the grand champion.

```
Total number of games played 10
Player X has won 3 (30%) games; Player O has won 5 (50%) games.
The grand champion is player O!!!
```

Restrictions:

- You must use a 2D array for the tic-tac-toe board.
- You cannot use classes for this question.
- You can use static methods if you like.

<u>Question 2 (5 Points)</u> - Defining a class (based on Question 11 p. 256 of your textbook).

a) Create a class named `Pizza` that stores information about a single pizza. It should contain the following:
   - Private instance variables to store the size of the pizza (small, medium or large), the number of cheese toppings, the number of pepperoni toppings, and the number of mushroom toppings.
   - Constructors:
     - one that takes four arguments and sets all of the corresponding instance variables
     - a default constructor, that takes no arguments and initializes all instance variables to the 'zero' of their type.
   - Methods to get (accessor) and set (mutator) each instance variable individually.
   - A public method called `calcCost()` that returns a double that is the cost of the pizza.
     Pizza cost is determined by:
     Small: $10 + $2 per topping
     Medium: $12 + $2.25 per topping
     Large: $14 + $2.50 per topping.

     For example, a large pizza with 1 cheese, 2 pepperoni and 1 mushroom toppings should cost $24 (14 + 4 x $2.50) while a small pizza with the same toppings will cost $18 (10 + 4 x $2).

   - A `toString()` method which takes no arguments and returns a string the pizza size, quantity of each topping, and the pizza cost as calculated by .
   - An `equals()` method which is redefined to test for the equality of the content of 2 objects of type Pizza.

b) Write a test code to
   - Create 3 different pizzas: First one with the default constructor, and for the 2nd and third one prompt the user.
   - Output the descriptions of the 3 pizzas.
   - Compare the 3 pizzas and display a message saying whether they are identical in content or not. Make sure you code test for the following scenarios when comparing the pizzas: that all three are the same, only 2 are the same, or none are the same.
   - Change the first pizza's content to be the same as one of the other ones.
   - Compare the 3 pizzas again and display a message saying whether they are identical in content or not as well as their contents.

   <u>Note</u>: Assume the user enters valid input.

Here is a sample output screen:

```
-------------------------------------------
Welcome to Fancy Nancy's Pizza Shop ....
-------------------------------------------

Please place order for 1st pizza:
Size:small, medium or large?  small
Indicate the number of each of the following toppings: cheese, pepperoni and mushroom
1 2 3
Please place order for 2nd pizza:
Size:small, medium or large?  large
Indicate the number of each of the following toppings: cheese, pepperoni and mushroom
3 2 1

Here are the three pizzas:
A  pizza with 0 cheese topping(s), 0 pepperoni topping(s) and 0 mushroom topping(s) cost $0.0
A SMALL pizza with 1 cheese topping(s), 2 pepperoni topping(s) and 3 mushroom topping(s) cost $22.0
A LARGE pizza with 3 cheese topping(s), 2 pepperoni topping(s) and 1 mushroom topping(s) cost $29.0

Results of comparisons:
None of the pizzas are the same


Results of comparisons after changing one of the pizzas ....
First and second pizza are the same.
   A SMALL pizza with 1 cheese topping(s), 2 pepperoni topping(s) and 3 mushroom topping(s) cost $22.0
Pizza 3:
   A LARGE pizza with 3 cheese topping(s), 2 pepperoni topping(s) and 1 mushroom topping(s) cost $29.0


Enjoy the pizzas ... Don't forget your TUMS on the way out.
```

Submitting Assignment 3

- Zip the source code (the .java file only please) of this assignment.
- Naming convention for zip file: Create one zip file, containing the source files for your assignment using the following naming convention:
  - The zip file should be called *a#_studentID*, where # is the number of the assignment and *studentID* is your student ID number.
    For example, for the third assignment, student 123456 would submit a zip file named a3_123456.zip
    Submit your zip file at: https://fis.encs.concordia.ca/eas/
- Submit your assignment as "**Programming Assignment**" and select Submission **3** for assignment #3.
- **Assignments not submitted to the correct location will not be graded.**
- **Be sure to keep your submission confirmation email.**