# CMPT 213 Assignment 4

Due: Nov 21, 11:59:59pm.

## Question 1 – Bank Account:

Description:

Design an abstract class named `BankAccount` to hold the following data for a bank account:

- The Balance
- Number of deposits this month
- Number of withdrawals
- Annual interest rate
- Monthly service charges

The Class should have the following methods (Illegal Arguments should throw an `IllegalArgumentException`):

| | |
|---|---|
| `Constructor` | The constructor should accept arguments for the balance and annual interest rate; both arguments must be non-negative |
| `deposit` | A method that accepts an (positive) argument for the amount of the deposit. The method should add the argument to the account balance. It should also increment the variable holding the number of deposits. |
| `withdraw` | A method that accepts an (positive) argument for the amount of the withdrawal. The method should subtract the argument from the balance. It should also increment the variable holding the number of withdrawals. |
| `calcInterest` | A method that updates the balance by calculating the monthly interest earned by the account, and adding this interest to the balance. This is performed by the following formulas:<br>Monthly Interest Rate = [Annual Interest Rate / 12)<br>Monthly Interest = Balance * Monthly Interest Rate<br>Balance = Balance + Monthly Interest |
| `monthlyProcess` | A method that subtracts the monthly service charges from the balance, calls the `calcInterest` method, and then sets the variables that hold the number of withdrawals, number of deposits, and monthly service charges to zero. |

Next, design a `SavingsAccount` class that extends the `BankAccount` class. The `SavinqsAccount` class should have a status field to represent an active or inactive account. If the balance of a savings account falls below $25, it becomes inactive. (The `status` field could be a boolean variable.) No more withdrawals may be made until the balance is raised above $25, at which time the account becomes active again. The savings account class should have the following methods:

| | |
|---|---|
| `withdraw` | A method that determines whether the account is inactive before a withdrawal is made. (No withdrawal will be allowed if the account is not active.) A withdrawal is then made by calling the superclass version of the method. |
| `deposit` | A method that determines whether the account is inactive before a deposit is made. If the account is inactive and the deposit brings the balance above $25, the account becomes active again. A deposit is then made by calling the superclass version of the method. |
| `monthlyProcess` | Before the superclass method is called, this method checks the number of withdrawals. If the number of withdrawals for the month is more than 4, a service charge of $1 for each withdrawal |

above 4 is added to the superclass field that holds the monthly service charges. (Don't forget to check the account balance after the service charge is taken. If the balance falls below $25, the account becomes inactive.)

Finally, write a series of tests using the Junit5 framework. The tests should successfully test for all requirements mentioned above. Be sure to include tests for error cases as well (For example, passing in a negative number into the `deposit()` method).

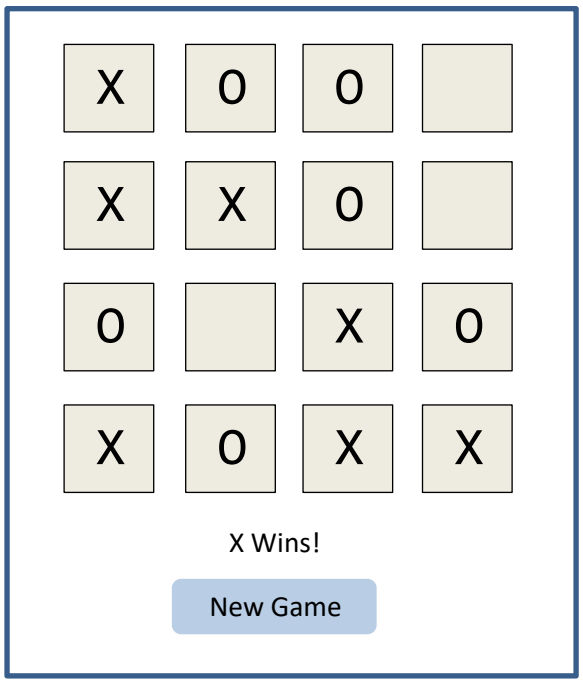
Technical Requirements:

- You do not need to write a `main()` method for this question. However, please note that we may write a `main()` method to call your class.
- Please put all your class files in a package called `ca.cmpt213.asn4.bank`
- Implementation must follow the online style guide. Specifically, important are:
  - Good class, method, field, and variable names.
  - Correct use of named constants.
  - JavaDoc - Good class-level comments (comment on the purpose of each class).
  - Clear logic.

## Question 2 – TicTac Toe:

Description:

In this question, you will create a *JavaFX* application that allows users to play 4x4 tic-tac-toe. The game takes place on a 4x4 board and involves two players, X and O. X starts and players take alternate turns. The object of the game is to place four of your pieces in a row, either vertically, horizontally, or diagonally. If a player wins, the game ends immediately. If the board is full, the game ends in a draw. Here is a rough sketch of the UI (you can be more creative than this :P): Do not use fxml to create elements in the UI.




- The game should use sixteen `ImageView` controls to display the X's and O's. The X's and O's should display image files (you may create them or find them online).
  - Do not create any of the UI elements using fxml.

- The application should use a two-dimensional `int` array to simulate the game board in memory.
- When the user clicks the *New Game* button, the application should clear the board and be ready to collect the first move.
- The current player will be able to click on the square that they want to place their move. This process will alternate until a winner is determined or there are no more empty spots on the board.

Technical Requirements:

- Main class should be in a file called `TicTacToe.java`
- Must exhibit good OOD principles:
  - Similar to asn3, you must have two packages: One package for the UI related class(es); another package for the model related classes (actual game logic). Please put all your class files in a packages called `ca.cmpt213.asn4.tictactoe.ui` and `ca.cmpt213.asn4.tictactoe.game`
  - Each class is responsible for one thing.
  - Reasonably detailed break-out of classes to handle responsibilities.
  - Good use of design principles discussed in class
- Implementation must follow the online style guide. Specifically, important are:
  - Good class, method, field, and variable names.
  - Correct use of named constants.
  - JavaDoc - Good class-level comments (comment on the purpose of each class).

**Note: any requirements not mentioned in this document is up to interpretation**. :)

**Marking Scheme:**

Question 1: Total [15] Marks

Question 2: Implementation - Total [25] Marks

Correctly follow coding style guide. - Total [0] Marks

- [-6] Up to 6 point max deductions

## Submission

Submit a zip file of your project (according to the directions outlined in the assignments link of the course website) to the coursys server. https://courses.cs.sfu.ca/. You do not need to generate JAR files for this assignment.

Please note: all submissions are automatically checked for similarities of all other submissions on the server.