Concordia University
Comp 249 - Winter 2015
Object Oriented programming II
Assignment 4

Deadline:	By 11:59pm, Friday April 10, 2015
Evaluation:	5% of your final grade
Late Submission:	No late submission.
Teams:	The assignment can be done individually or in teams of 2 (from the same lecture section). Submit only one assignment per team.
Purpose:	The purpose of this assignment is to have you experience dealing with GUIs, linked lists, and inner classes

---

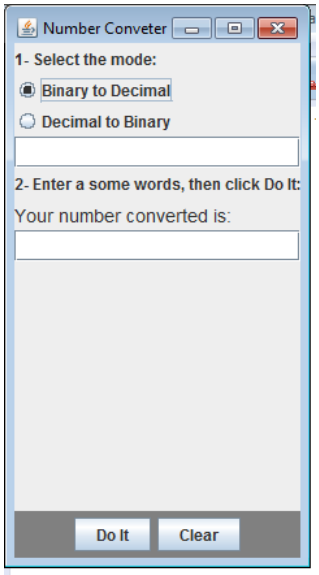
General Guidelines When Writing Programs:

See handout for assignment 1 for details.

Question 1 – Swing

A number converter converts a binary number to its decimal notation or vice versa. The Swing GUI is like the graph on the right side of the page. It works in this way:

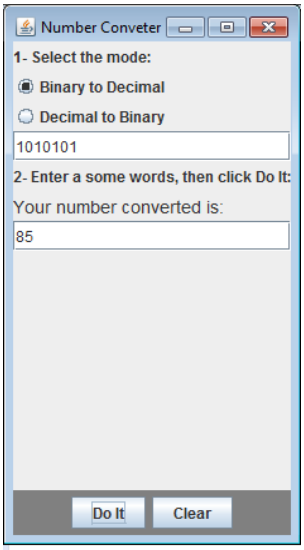
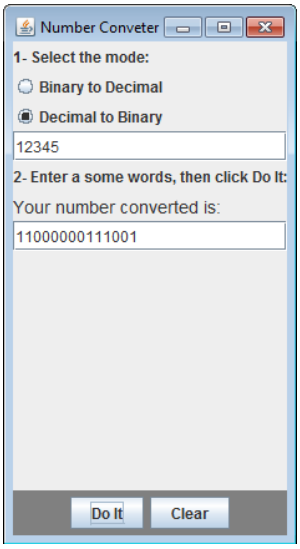
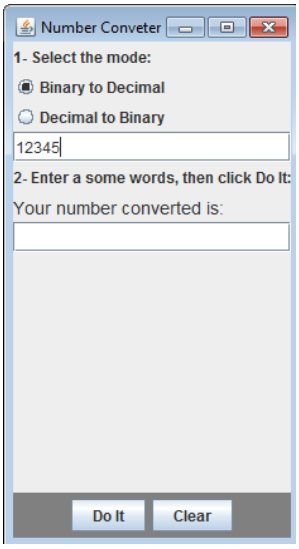
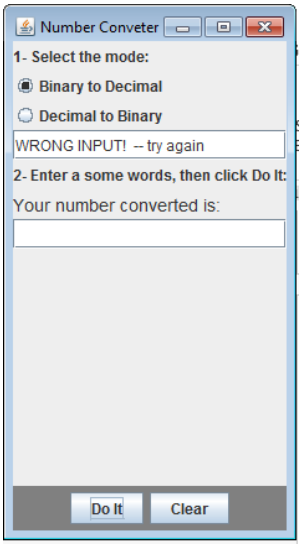
1. Select the convert mode by clicking one of the two radio buttons.
2. Enter the number in the first text field.
3. Click the "convert" button, and the result is in the bottom text field.
4. Click the "clear" button, the two text fields are cleared.
5. If the number is not a desired notation, e.g. select "Binary to Decimal", but input is a decimal, a wrong message "WRONG INPUT! -- try again" displayed in the input textbox.



Requirements:

- Build a swing application to implement this number converter.
- To the right is a sample of what your GUI can look like. You are free to choose the layout and colors.
-

Following are some sample screen captures to illustrate the expected behavior of your Swing GUI.

Binary to Decimal	Decimal to Binary	Binary to decimal but input is not a binary number	
			

Question 2 – Single Linked Lists

Part 1. Consider a class *Term*, which represents a single term of a polynomial of a single variable and is set up so that aterm can be part of a linked list.

Term
- coefficient: double - degree:int - link:Term
+ Term() + Term(coef:double, deg:int, linkValue:Term) + setData(coef: double, deg:int):boolean + setLink(newLink:term):boolean + getCoefficient():double + getDegree():int + getLink():Term + toString():String + equals():boolean

The `toString()` method should return the term in the format
- *cx^d* if degrees are greater than 0 or
- *c* if degrees is equal to 0

where *c* is the coefficient and *d* is the degrees of the term.

Consider the class *Polynomial* which stores a single variable polynomial equation. It is represented as a linked list of *Term* objects with a pointer to the *head* (first term) of the polynomial list and a pointer to the *tail* (last term of the polynomial)

Polynomial
- head: Term - tail:Term
+ Polynomial() + copyOfPolynomial() + addToStart(coef:double, deg:int): void + addToEnd(coef:double, deg:int): void + numberOfTerms() : int + add(poly: Polynomial): Polynomial + evaluate(x: double): double + derivative():Polynomial + integral(): Polynomial + readPolynomial(): void + toString():String + equals(poly: Polynomial):boolean

Explanation of the methods:

- Constructor with no arguments which set all attributes to the zero of the type.
- **copyOfPolynomial()** returns a deep copy of the calling object
- **addToStart()** adds a *Term* to the beginning of the list. Make sure the references of head and tail are correctly updated.
- **addToEnd()** adds a *Term* to the end of the list. Make sure the references of head and tail are correctly updated.
- **numberOfTerms()** returns the number of terms of the calling polynomial.
- **add()** returns a Polynomial which contains the sum of the calling polynomial and the passed polynomial
- **evaluate()** computes the value of the polynomial at the passed double value.
- **derivative()** returns the derivative of the calling polynomial.
- **integral()** returns the integral of the calling polynomial.
- **readPolynomial()** which prompts the user for the polynomial and stores it into the calling polynomial.
- **toString()** such that the statement *System.out.println(a Polynomial object)* outputs the polynomial in the format 3x^2 – 2x^2 + 9 or -3x^2 – 2x^2 + 9.
  The first term should start with a digit if the first coefficient is positive and a – if it is negative.
- **equals()** which will return true if the calling polynomial and the passed polynomial contain the same number of terms with the same coefficients and degrees, false otherwise.
- The class *Term* should be a **private** inner class of the class Polynomial. The methods in Polynomial should use the methods of Term whenever possible.

Part 2. Write a driver program that will (see sample screen on next page) :

1. Prompt the user for a polynomial poly1 (make sure some of the coefficiecent are negative).
2. Create a polynomial poly2 which will contain the derivative of the polynomial poly1.
3. Create a polynomial poly3 which will contain the integral of the polynomial poly2.
4. Create a polynomial poly4 which will contain the sum of poly1 and poly2.
5. Create a polynomial poly5 which will contain a deep copy of poly1.
6. Prompt the user for a double value x.
7. Will check whether poly1 is equal to poly1, poly1 is equal to poly2, poly1 is equal to poly3 and poly1 is equal to poly5 and print appropriate messages.
8. Print the original polynomial along with the number of terms it contaims and the value of the polynomial at the value entered by the user in step 6

A few restrictions:

1. An object of type *Polynomial* should contain only non-zero terms.
2. You can add other methods to *Polynomial*, but you cannot change the header or functionality of the methods described above.
3. The class *Term* should be a **private** inner class of the class *Polynomial*. The methods in *Polynomial* should use the methods of *Term* whenever possible.
4. You cannot add any other methods to the inner class *Term* or change the the header or functionality of the methods described above.

Sample output screen:

```
==================================
 ======= Polynomial Driver =======
 ==================================

Enter a polynomial - Start with term with highest degree. Enter coefficient then degree:
3 3
Another Term? (1 for yes): 1
-2 1
Another Term? (1 for yes): 1
7 0
Another Term? (1 for yes): 0
The polynomial you just entered is 3.0x^3 -2.0x^1 + 7.0

The derivative of this polynomial is 9.0x^2 -2.0

The integral of this last polynomial is 3.0x^3 -2.0x^1

The sum of the original polynomial and its derivative is 3.0x^3 + 9.0x^2 -2.0x^1 + 5.0

Are the equations the same?
3.0x^3 -2.0x^1 + 7.0 and 3.0x^3 -2.0x^1 + 7.0 are equal is true
3.0x^3 -2.0x^1 + 7.0 and 9.0x^2 -2.0 are equal is false
3.0x^3 -2.0x^1 + 7.0 and 3.0x^3 -2.0x^1 are equal is false
3.0x^3 -2.0x^1 + 7.0 and 3.0x^3 -2.0x^1 + 7.0 (copy of first one) are equal is true

Enter a value of x to evaluate the first polynomial at:
5
Polynomial 3.0x^3 -2.0x^1 + 7.0 has 3 terms and evaluates to 372.0 for x = 5.0

All done .... Bye Bye!
```

Submitting Assignment 4

- Naming convention for zip file: Create one zip file containing all source files and Javadoc files for your assignment using the following naming convention:
  - If the assignment is done by 1 student:
    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 first assignment, student 123456 would submit a zip file named a4_123456.zip.
  - If the assignment is done by 2 students:
    The zip file should be called *a#_studentID1_studentID2*, where # is the number of the assignment, and *studentID1* and *studentID2* are the student ID numbers of each student.
    For example, for the first assignment, student 123456 and 9876543 would submit a zip file named a4_123456_9876543.zip
- Submit your zip file at: https://fis.encs.concordia.ca/eas/ as **Programming Assignment** and submission **4** for **COMP249.**
- Be warned that assignments not submitted in the requested location, be it in a different folder than Programming Assignment, a different submission number than 4 or in the folder of a different course will not be graded. If the wrong files are submitted you will not have the chance of resubmitting. We were lenient in COMP 248, but now it is your responsibility to make sure your assignment is submitted in the proper location and that the requested files are present.
- Submit only ONE version of an assignment. If more than one version is submitted the first one will be graded and all others will be disregarded.

Evaluation Criteria for Assignment 4 (20 points)

Question 1 (9 points)	
GUI	9 pts
Question 2 (8 points)	
Declaration of the class Polynomial with inner classs Term	6 pts
Driver program	2 pts
For entire assignment (3pts)	
Use of significant variable names and comments, indentation and readilbility	3 pts
Total	**20 pts**