CoSc 20803 – Fall, 2015
Lab #1

Due Date: **GUI** code – **Thursday, Sept 24, 2015** (all files, including a **runnable `.JAR`** file, related to your Lab1 project "zipped" together and submitted with **TURNIN**).

Finished Project – Thursday, Oct 8, 2015 (all files, including a **runnable `.JAR`** file, related to your Lab1 project "zipped" together and submitted with **TURNIN**), hardcopy printout of User's Manual turned in at my office by noon on Friday morning (Oct 9, 2015) - (**not accepted late!!!**).

On the Friday morning following the due date for the final project, you will be required to provide a short User's Manual detailing the proper procedures to follow in order to use your program. Be sure to include any information that will be needed in order to test your code (include statements about any features that you failed to implement).

This MUST be turned in at my office (TUC 208) or the secretary's office (TUC 207) by 12:00 noon. Failure to do so will result in a substantial reduction in your lab grade for the assignment.

Language: Java

Lab Value: This assignment is worth 250 points total (it constitutes roughly 38% of the lab grade). You should use good OOP techniques in developing your lab solution. You were taught the MVC programming architectural pattern in COSC10403 (and probably in COSC 20203). *You are welcome to copy that design pattern.*

Purpose: To exercise your knowledge of circular linked lists.

Documentation: You are expected to thoroughly document your program to the extent discussed in class. Your program should contain "preamble" documentation like that taught in CoSc 10403 and CoSc 20203 **AND** internal documentation consistent with that expected of experienced programmers. ***If you don't know what that means – ASK!!!*** See the CoSc 10403 Documentation Standard for a very simple example (you will need to do better!).

Feel free to use JavaDoc.

Problem: The history of mathematics, science, and technology has a long tradition of discoveries. These discoveries in recent years have been enhanced by the creation of new computational technologies that support integrated symbolic computation for applications ranging from mathematics to chemistry to geometry to finance. Over the past several years, various computer algebra systems have evolved. Powerful tools such as Wolfram Research's Mathematica and Maplesoft's Maple13 and MapleSim and MatLab are examples.

One of the "classical" problems in computer science is to perform symbolic manipulation of polynomials with integral coefficients. **Note**: *a polynomial is a mathematical expression involving a sum of powers multiplied by coefficients*. A polynomial expressed in one variable is known as a univariate polynomial while one expressed in more than one variable is known as a multivariate polynomial. *Note also that constants are considered to be multiplied by a variable raised to the zero power*. For this assignment, you will be developing a Java, stand-alone application (not an applet) with associated GUI to manipulate polynomials based on the basic list processing methods outlined below.

Your program will need to support operations on multivariate polynomials in three variables (**x**, **y**, and **z**) with constant non-negative whole number exponents and coefficients. Three variable polynomials are of the following form:

$$p(x,y,z) = c_n x^{A1} y^{B1} z^{C1} + \ldots + c_1 x^{An} y^{Bn} z^{Cn}$$

where the C_n are non-zero coefficients and the $A_i B_i C_i$, represent each variable's exponent.

An important issue when working with symbolic representations of polynomials involves maintaining the order of the variables. For example it is necessary that the term x^2yz be recognized as identical to yx^2z . The easiest way to achieve this is to always keep the variable names in a consistent order. The most obvious way to do this is alphabetically. Thus no matter what order the user enters the term x^2yz, whether it be as x^2yz or as yx^2z or as x^2zy or any other permutation it will always be stored as x^2yz .

Another idea would involve storing the individual polynomial terms such that the exponents $A_i B_i C_i$, are arranged as:

$$A_1B_1C_1 > A_2B_2C_2 > \ldots > A_nB_nC_n >= 0.$$

For Example: For the current assignment:

$p(x,y,z) = 6x^9y^6z^3 + 5x^4z^5 - 3$ is a valid multivariate polynomial, while:

$p(x,y,z) = 6x^2y - 4/x + 7x^{3/2}$ is **not** because its second term involves division by the variable x (thus making it a negative exponent) and also because its third term contains an exponent that is not a whole number.

Operations: Given the following multivariate polynomials –

$$A = 3x^2 + 5xy - 8y^2 + 4x - 3y + 12$$

$$B = x^3 + 6y^2 - 15x + 11$$

$$C = 2x + 3y - 5$$

$$D = x - 4y + 1$$

- - - - - - - -- - - - - - - - - Valid Math operations -

$$(A+B) = x^3 + 3x^2 + 5xy - 2y^2 - 11x - 3y + 23$$

$$(A-B) = -x^3 + 3x^2 + 5xy - 14y^2 + 19x - 3y + 1$$

$$(C*D) = 2x^2 - 5xy - 12y^2 - 3x + 23y - 5$$

Adding or subtracting polynomials is just a matter of searching for terms with the same exponent and adding or subtracting their coefficients. Multiplication and division are just a matter of properly modifying exponents. Notice however, that dividing a single PolyTerm by another single PolyTerm might cause negative exponents to appear which stretches our mathematical definition of polynomial.

Further, division of two polynomials, each of multiple terms, only succeeds if the division results in no remainder. Thus, Division is the most complicated arithmetic operation on multivariate polynomials and the result is in general not a polynomial.

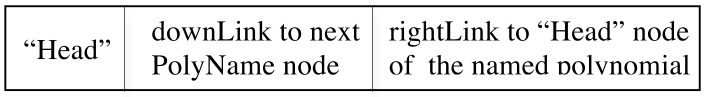
For this assignment you will be asked to develop a system that will support addition and subtraction of multivariate polynomials and multiplication for "Bonus" credit. *We will skip division*.

Data Structure: For this program you are required to maintain a circular linked list of multivariate ***PolyName*** nodes each of which has a *rightLink* pointing to the head node for its respective polynomial and a *downlink* pointing to the next PolyName node.

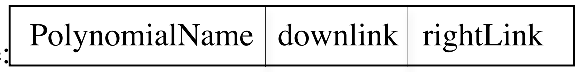
NOTE: You may NOT use arrays, vectors, nor any of the builtin Java classes other than those swing classes necessary for the GUI.

The node formats for terms in this linked list are:

PolyNameList "Header":

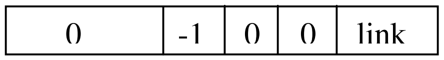

"Head"
downLink to next PolyName node
rightLink to "Head" node of the named polynomial

PolylName node:

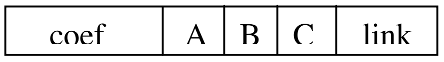


For your program, each term of the multivariate polynomial will be represented by a separate node in a circular linked list. Each node will be of fixed size having fields which represent the coefficient and **x**, **y**, and **z** exponents of a given term plus a pointer to the next lower term. You may assume that all coefficients and exponents are integer and that exponents are non-negative. Your ***PolyNode*** structure should correspond to the following format:

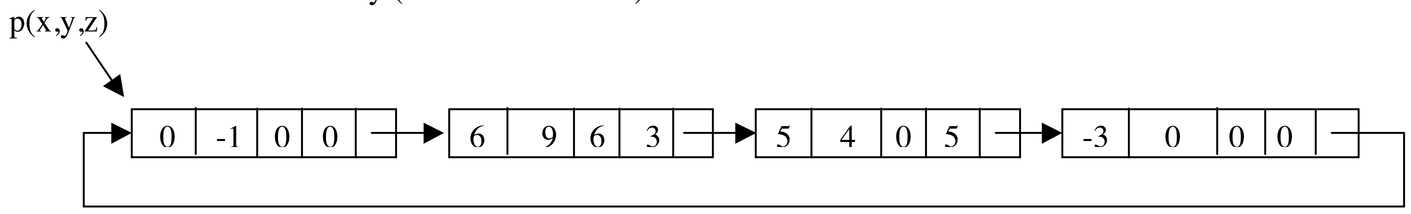
PolyNode "Header":



PolyNode "standard term" node:



Thus, the polynomial ($6x^9y^6z^3 + 5x^4z^5 - 3$), shown earlier, will be represented internally (with "Head" node) as:



Note that the nodes are organized within the list such that: 963 (the ABC exponents for node #1) precede the 2nd node (which has ABC exponents = 405). Your algorithms will be greatly simplified by so doing. A multivariate polynomial linked list is most easily constructed if the incoming exponents, $A_iB_iC_i$, are read in decreasing order. Feel free to develop your program with this understanding – **of course, you are not required to make this restriction.**

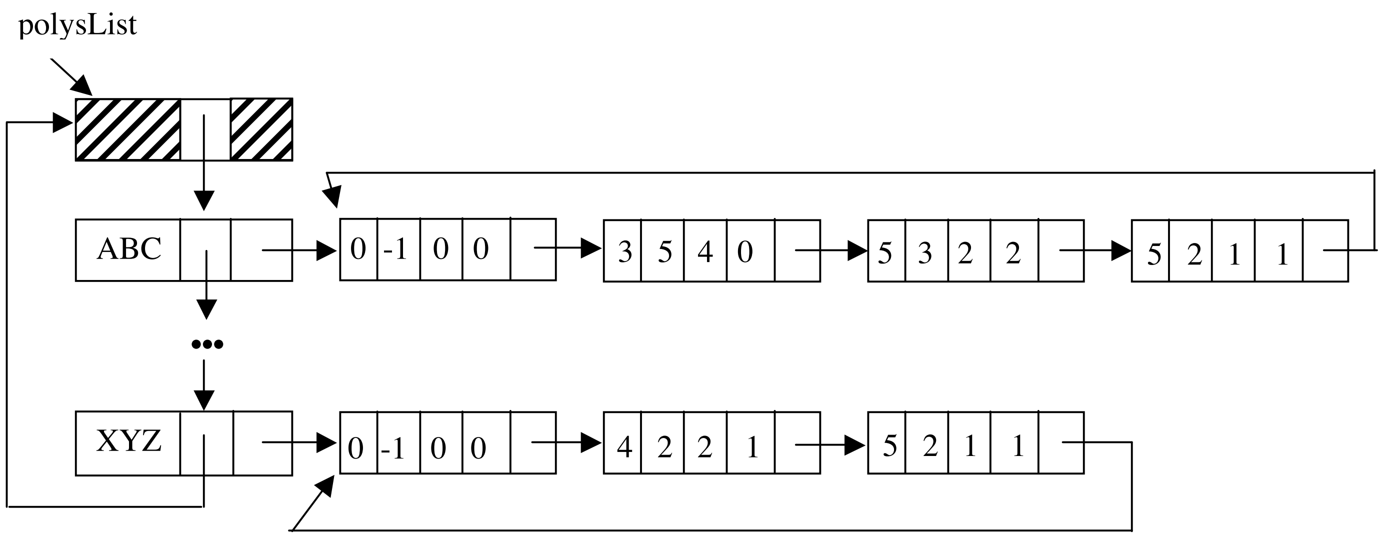
The data structure representation shown below is an example of what is REQUIRED of all software submitted for this assignment.

For Example: Assume the following two multivariate polynomials –

$$\mathtt{ABC} = 3x^5y^4 + 5x^3y^2z^2 - 5x^2y^1z^1$$

$$\mathtt{XYZ} = 4x^2y^2z^1 + 5x^2y^1z^1$$

Given the above two polynomials as input, your program should construct the following data structure.



Input: Your GUI should be intuitive and must support a variety of operations (e.g., enter new polynomial, add two polynomials, subtract two polynomials, display a polynomial, evaluate a polynomial (given values for x, y, and z), delete a polynomial, etc.). In this regard, you are free to design an interface that will allow the user to input any information necessary to perform these operations (**`JLists`**, **`JCheckboxes`**, **`JComboBoxes`**, etc).

Obviously, your grade will depend substantially on the esthetics of your GUI and its functionality as well as on whether the program produces correct output for the various operations.

With regard to reading an equation in polynomial form from the user – you are free to design your program to accept that input in any form. (i.e., you may allow a user to type the full equation into a **`JTextArea`** or **`JTextField`** in polynomial form similar to the following:

Alpha = 6*x^9*y^6*z^3 + 5*x^4*y^0*z^5 - 3*x^0*y^0*z^0

where "**^**" means exponentiation). Your program can then parse the input string into (coefficient and exponent pairs) in order to build up the circular linked list for the named polynomial. (*The Java* **`StringTokenizer`** *class will be helpful*).

Alternatively, you might prompt the user for one term of the polynomial, add that term to the linked list, and prompt for the next term – terminating whenever the last term is entered. This is entirely up to you.

Data: You should develop your own test suite to determine the correctness of your program. You should be careful to do your testing completely. I will use my own test suite to determine the accuracy of your program.

Your program will be enhanced by providing a "user feedback" area to supply operational feedback to the user informing him/her of what operation was just performed (e.g., a "new polynomial was CREATED", two polynomials were added or subtracted (or multiplied if attempting the "**Bonus**" credit) and the resulting polynomials is:, etc).

This same area may be used to produce a nicely formatted polynomial for output when asked to do so by the user. **Note**: polynomials should be displayed in an appropriate form. Example: $5x^4$ might be displayed as:

```
5 * x ^ 4 * y ^ 0 * z ^ 0
```

Interface: For this assignment, you must develop a Java stand-alone application (not an applet) that employees an easy to use, intuitive GUI that will allow the user to input any information necessary to perform the operations outlined below (you should use the Java swing classes as they provide a "richer" appearing interface).

Active "widgets" (**`JButtons`**, **`JTextFields`**, etc.) should be employed whenever possible to make your program easier to use (you will want to use the methods: **`setVisible()`**, **`setEnabled()`**, **`requestFocus()`**, etc…).

Requirements: Your software should provide, at a minimum, the basic functionality listed below. You are free to implement other methods besides these.

(1) <u>**`createPoly`**</u> — reads in a multivariate polynomial from the user, converts the polynomial into a circular linked list according to the representation specified earlier, and <u>returns</u> the address of the "Head" node.

The following methods should probably employ either a `JComboBox` or `JList` widget in order for the user to select polynomials that need to be operated on.

(2) <u>**`displayPoly`**</u> — should allow the user to click and select a polynomial to be displayed. The chosen trinomial should be displayed in an appropriate form. Example., $5x^4$ might be displayed as:

5 * x ^ 4 * y ^ 0 * z ^ 0).

(3) <u>**`deletePoly`**</u> — this method should delete the chosen polynomial.

(4) <u>**`addPoly`**</u> — this method should compute and display the sum (i.e., $\alpha + \beta$) of any two polynomials selected by the user. The original α and β polynomials are to be left unaltered. <u>After calculation, the user should be given the option to name and save the resultant polynomial or to discard it.</u>

(5) <u>**`subtractPoly`**</u> — this method should compute and display the difference (i.e., $\alpha - \beta$) of any two polynomials selected by the user. The original α and β polynomials are to be left unaltered. <u>After calculation, the user should be given the option to name and save the resultant polynomial or to discard it.</u>

(6) <u>**`evalPoly`**</u> — this method should evaluate the selected polynomial for the given real values "x", "y", and "z" supplied by the user. (Is Horner's rule useful?)

(7) <u>**`exitPoly`**</u> — a function used to close the system down at the end of the day. <u>**No backup of the polynonial data structure is required**</u>.

Note: For "***BONUS***" credit (15 points each), you may choose to implement any (or all) of the following additional functionality:

(i) <u>**`storePoly`**</u> — used to store the contents of the polynomial data structure to a disk file for later use by the loadPoly function (described below).

(ii) <u>**`loadPoly`**</u> — used to restore the polynomial data structure at the beginning of the day.

(iii) <u>**`multiplyPoly`**</u> — this method should compute and display the product (i.e., $\alpha * \beta$) of any two polynomials selected by the user. The original α and β

polynomials are to be left unaltered. After calculation, the user should be given the option to name and save the resultant polynomial or to discard it.

Requirements: Two Java interfaces are provided below. Your Lab1 solution should **implement** both of these interfaces.

The following two Java interfaces MUST be included and compiled along with your program:

Interface #1: `PolyNameNodeInterface`

```
// PolyNameNode interface (** should be included as a part of your Lab1 project)
// *****************PUBLIC OPERATIONS*****************************************
// public String getPolyName();
//          -->  returns the name of the polynomial being pointed to
//
// public void setPolyName(String s);
//          -->  sets the name of this polynomial to s
//
// public PolyNameNode getDownPtr();
//          -->  returns the value of the “down ptr” for the specified PolyListNode
//
// public void setDownPtr(PolyNameNode p);
//          -->  sets the “down ptr” of the specified node to p
//
// public PolyNode getRightPtr();
//          -->  returns the value of the “right ptr” for the specified PolyNameNode
//
// public void setRightPtr(PolyNode p);
//          -->  sets the “right ptr” of the specified node to p
//
// ***************************************************************************

/**
* Protocol for PolyNameNodeInterface
* @author James Comer
*/
public interface PolyNameNodeInterface
{
   //Method signatures
   public String getPolyName();
   public void setPolyName(String s);
   public PolyNameNode getDownPtr();
   public void setDownPtr(PolyNameNode p);
   public PolyNode getRightPtr();
   public void setRightPtr(PolyNode p);
}
```

Interface #2: `PolyNodeInterface`

```
// PolyNode interface (** should be included as a part of your Lab1 project)
// *****************PUBLIC OPERATIONS*****************************************
// public int getCoeff();
//          -->  returns the coeff of a specified polynomial term
//
// public void setCoeff(int c);
//          -->  sets the coeff of the specified polynomial's term to c
//
// public int getXPower();
//          -->  returns the specified polynomial's x power
//
// public void setXPower(int ix);
//          -->  sets the specified polynomial's x power to ix
//
// public int getYPower();
//          -->  returns the specified polynomial's y power
//
// public void setYPower(int iy);
//          -->  sets the specified polynomial's y power to iy
//
// public int getZPower();
//          -->  returns the specified polynomial's z power
//
// public void setZPower(int iz);
//          -->  sets the specified polynomial's z power to iz
//
// public PolyNode getPtr();
//          -->  returns the value of the specified polynomial's link field
//
// public void setPtr(PolyNode p);
//          -->  sets the specified polynomial's link field to p
//
// *****************************************************************************

/**
* Protocol for PolyNodeInterface
* @author James Comer
*/
public interface PolyNodeInterface
{
   //Method signatures
   public int getCoeff();
   public void setCoeff(int c);
   public int getXPower();
   public void setXPower(int ix);
   public int getYPower();
   public void setYPower(int iy);
   public int getZPower();
   public void setZPower(int iz);
   public PolyNode getPtr();
   public void setPtr(PolyNode p);
}
```