**CSC-14400: Computer Science I – Lab #4**
Optional Homework Due: Tuesday, November 17, 2015
In-Class Exercise: Thursday, November 19, 2015
Quiz On Related Material: Tuesday, November 24, 2015

**Objectives:**
This exercise will get you (further) experience with:

- editing, compiling, executing, and debugging a Java program. (again)
- working with Java classes. (again)
- using selective execution.
- using iterative execution.
- using arrays.

**Preliminaries**
Before attempting to complete this assignment, you should be comfortable with the material from Chapters 1 – 7 and 11 of the text book. As a result, you should be comfortable with all prior lab work and associated problems. You should also be able to complete the following problems before solving the actual lab problem.

1. Write a complete call to `printf` that would:
   (a) Print out the contents of an integer variable `a` in 6 spaces, right justified and followed by a newline.
   (b) Print out the contents of an integer variable `b1` in 7 spaces, left justified and followed by an integer variable `b2` in 8 left justified spaces followed by a newline. You should accomplish this with *one* call to `printf`.
   (c) Print out the contents of a double variable `c` in 11 spaces, right justified, with exactly 5 digits after the decimal point, and followed by a newline.
2. *Textbook, Chapter 4*: p. 169, problem E4.12.
3. *Textbook, Chapter 5*: p. 223-228, problems R5.1, R5.2, R5.26, R5.29 – R5.31, E5.7
4. What is the output resulting from the following code sequence?

```
int passNum=0;
if (passNum%2==1) // if passNum is odd ...
{
  while(passNum<10)
  {
    System.out.println(passNum);
    passNum++;
  }
}
```

5. What is the output resulting from the following code sequence?

```
int passNum=0;
while(passNum<10)
{
  if (passNum%2==1) // if passNum is odd ...
  {
    System.out.println(passNum);
  }
  passNum++;
}
```

6. What is the output resulting from the following code sequence?

```
int passNum=0;
while(passNum++<3)
  System.out.println(passNum);
```

7. What is the output resulting from the following code sequence?

```
int passNum=0;
while(++passNum<3)
  System.out.println(passNum);
```

8. chapter 6: p. 293-298, R6.1(a), R6.2(d), R6.15, R6.16, R6.18 (note that this problem has *3* parts), R6.27 (hint: how many *'s need to be printed in total?)

9. chapter 7: p. 361-365, R7.2, R7.3, R7.7, R7.13, R7.14(a)-(c), R7.31(a)-(e);

10. Consider each of the following code sequences and

    - if it would not compile, give "does not compile" as your answer.
    - if it would result in a run-time exception, give "exception" as your answer.
    - if it would result in a (non-null) memory reference, give "memory reference" as your answer.
    - in *any* other case, show the resulting output.

    (a) `int a[]; System.out.print(a);`

    (b) `int b[]; System.out.print(b[0]);`

    (c) `int c[] = new int[20]; System.out.print(c);`

    (d) `int d[] = new int[20]; System.out.print(d[0]);`

    (e) `String e[] = new String[20]; System.out.print(e);`

    (f) `String f[] = new String[20]; System.out.print(f[0]);`

    (g) `String g[] = new String[20]; System.out.print(g[0].length());`

    (h) `String h[] = new String[20]; System.out.print(h.length());`

    (i) `String i[] = new String[20]; System.out.print(i.length);`

11. Suppose you have a class called `Student` which has (among other methods) a method called `getAverage` that takes no inputs and returns a `double`. Give Java code for each of the following:

    (a) Declare a variable that could hold a single object of type `Student` and initialize it using a default constructor.

    (b) Using the variable from the last problem, call the `getAverage` method. Note that this part of the problem and the previous part are essentially review of prior material.

    (c) Declare a variable to hold an array of 10 `Student` objects and initialize the array appropriately (without calling any `Student` constructors yet).

    (d) Write a loop that would initialize each element of the array from the last part to a default constructed `Student`.

    (e) Give code that would print out the average (by calling the `getAverage` method) for the student found at index 7 of your array.

**The Exercise**

You will be writing a class called `Polynomial`, as specified below. You will also be making a small change to the `PolyTerm` class from Lab#3 (a solution to which is being provided), also described below:

**Change to the PolyTerm Class**

Make sure that when printing a `PolyTerm` using its `print` method, you follow some basic rules:

- coefficients should be printed with exactly two digits after the decimal place.
- when the coefficient of a `PolyTerm` is 1.0, the coefficient should not be printed (although the variable and the degree should be printed). For example, if the coefficient is 1.0 and the degree is 4, then `x^4` should be printed (not `1.00x^4`).
- when the degree of a `PolyTerm` is 1, the degree should not be printed (although the coefficient and the variable should be printed). For example, if the coefficient is 7.0 and the degree is 1, then `7.00x` should be printed (not `7.00x^1`).
- when the degree of a `PolyTerm` is 0, neither the degree or the variable should printed (although the coefficient should). For example, if the coefficient is -5.0 and the degree is 0, then `-5.00` should be printed (not `-5.00x^0`).
- There may be some "common sense" fixes to the above in the case of conflicting information. For example, if the coefficient is 1.0 and the degree is 0, you should print `1.00` (instead of nothing).

**The Polynomial Class**

A `Polynomial` is essentially a collection of `PolyTerm`s. You may assume that no `Polynomial` will have a degree larger than 99 or more than 100 `PolyTerm`s in it (although 100 terms is possible). Note that 0 terms is also possible.

In order to support this you should code a `Polynomial` class with the following public interface:

- a no-argument constructor that builds an empty Polynomial (that may later need to hold up to `100` terms).

- a method called `addTerm` that returns a Polynomial and takes a single `PolyTerm` as its only parameter. This method should create and return a *new* `Polynomial` that is the same as "this" `Polynomial`, but has the `PolyTerm` represented by the parameter added to the new `Polynomial`. Note that "this" `Polynomial` may already contain a `PolyTerm` with the same degree as the parameter; in this case, the corresponding coefficients should be added. *The* `addTerm` *method should NOT modify "this"* `Polynomial`*; it should create a new* `Polynomial` *and return it.*

- a method called `addPolynomial` that returns a Polynomial and takes a single `Polynomial` as its only parameter. This method should create and return a *new* `Polynomial` that is the same as "this" `Polynomial`, but has all of the `PolyTerm`s in the parameter added in. Note that "this" `Polynomial` and the parameter may contain several `PolyTerm`s of equal degree values; in these cases, the corresponding coefficients should be added. *The* `addPolynomial` *method should NOT modify "this"* `Polynomial`*; it should create a new* `Polynomial` *and return it.*

- a method called `evalAt` that takes a single double as its only parameter and returns a double. This should simply evaluate "this" `Polynomial` at the value of the parameter and return that value.

- a method called `print` that takes no arguments and prints each `PolyTerm` to the console, with a plus sign ("+") between two `PolyTerm`s when the second one has a positive coefficient. *The* `PolyTerm`*s should be printed from higest degree to lowest degree (left to right), and no two printed terms should have the same degree.*

- a method called `read` which should take a single parameter representing the full pathname of a file. This file will contain a sequence of `PolyTerm`s to be added to "this" `Polynomial`.

You are welcome to add more methods to the public interface if you wish. I will be providing you with some example `main` methods as well as some example input files via BlackBoard. I will also be providing a solution to Lab3's `PolyTerm` class, which you will have to amend as described above.

**Final Note**

- You are *required* to work on the exercise *before* the actual lab. If you do not, you will receive a zero for the lab and will be required to leave class for the day immediately.