**Polynomials**

If you happen to speak multiple languages, you'll be familiar with the concept of "use it or lose it" – if you don't practice your multi-lingual skills, they quickly get rusty and eventually fade away. Programming languages are no different. Hopefully you spent at a *little* time programming this summer – regardless, let's review some of the basics to get us back in the swing of things.

For those that didn't take PAP/CS 1 with Mr. Hanna, there is a comprehensive set of powerpoints on our website that exactly follow the material in this class - the "Java powerpoints" link is on the landing (home) page. **You'll need to sign in with your school email (e.g.** jane.doe.123@k12.friscoisd.org**).**

As the material in this lab is intended as a review, it might be useful to utilize both the PAP/CS 1 material and the AP material **(note that AP has its own folder of powerpoints).**

1. Open BlueJ\*, make a new project (saved anywhere for now), and make a new class called Tester.

/\* At this point you should be using BlueJ, a user-friendly Java IDE. If you've never used BlueJ before, the powerpoints on the website will help. Soon, I will give some instruction on using Eclipse (a professional-level IDE with many built-in shortcuts and features). \*/

1. Create a public static void main(String[] args) method inside Tester.
2. Write a print statement that will print "Hello again, world!" to the console (screen).
3. Create a comment below the print statement that says: "the above prints "Hello again, world!"".
4. Declare and initialize (to a value of your choosing) an integer variable called numApples.
   1. Per Java convention, variable names should be descriptive, **should start with a lower-case letter**, and should always be in camelCase (when applicable).
5. Declare and initialize an integer variable *constant* called PRICE\_OF\_APPLES representing the price of an apple (in cents). 
   1. Per Java convention, constants should be in all caps with underscores separating words and should be preceded by the final keyword.
6. Using ***one*** println statement, print the total cost for all apples to the console as follows (where the angle brackets < and > indicate you should use the variables' values rather than literal text):

The total for <numApples> apples:

<numApples> \* <PRICE\_OF\_APPLES> cents

* 1. **Hint**: Use the concatenation operator (+) to join two or more Strings. Use the escape code for a newline character to print a "return".

1. Write an if statement that will print "Thank you valued customer!" if the total price of all apples was at least 2000 cents. Maybe you don't even need curly brackets for this; check the "More on curly brackets" powerpoint for more info.
2. Write a for loop that will produce the following output: 1 2 3 4 5 6 7 8 9 10 11 12
3. Write a for loop that will produce the following output: 1, 4, 9, 16, 25, 36, 49, 64, 81
4. Write a for loop that will produce the following output: 1 4 16 64 256
5. (Riddle) What disappears the moment you say its name?

/\* I am a big fan of riddles and critical thinking in general. In most of my labs, you will find at least one riddle to solve. Riddles are for fun - they are not "required" as part of the lab. Solve them if you have time! \*/

1. Declare and initialize a new Scanner object called console that you will use to get keyboard (user) input. Don't forget to add the import statement above the class header!
2. Declare and initialize a new String called name whose value is taken from the keyboard. Don't forget to prompt the user (with a print statement) to enter something prior to reading in the value!
3. Print the length (number of characters) of the name variable to the console; this should work no matter what the user enters. Next, print the first three characters (only) of what the user entered.
   1. The substring(int start, int end) method is (inclusive, exclusive).
4. Write a while loop that will sum all the integer values entered by the user, until the user enters 0. Print the total sum and the average of the numbers entered (the average should be a double).
5. Declare (only!) an array of doubles called areas. On separate line, initialize the array such that it will have 20 elements.
6. Set the value of the first element in areas equal to 4.56. Note that Java, like most programming languages, uses zero as the first index!
7. Declare a new integer variable called length and set it equal to the number of elements in areas **without using an integer literal to the right of the equals sign.**
8. Set the value of the last element in areasequal to 8.08 **using the** length **variable declared previously.**
9. Declare and initialize (**in one line**) a new boolean array that starts with the values true, true, false, false, true.
10. (Riddle) I have a name, but it isn't *my* name. My face shows signs of age. I always mean the same thing, no matter what I say. I'm born in mourning, and I last 'til the end of days. People plant me, but I never grow. They run from me, but I never move. They look at me and see their future. What am I?

/\* If you solve a riddle, don't say it out loud and spoil it for everyone else! Write it down somewhere and let me know, I'll check it \*/

1. **Outside the** main **method,** write a static method with a void return type that will print "This is a method!" when called.
   1. By convention, method names must be descriptive, **should start with a lower-case letter**, and should always be in camelCase (when applicable).
      1. Method names can be distinguished from variable names in that they will always be followed by round brackets (parentheses).
2. Write a method public static int sum(int a, int b) that will return the sum of the integer parameters. Make sure you test that it works.
3. Write a method public static String longer(String a, String b) that will return the longer of two String parameters. Make sure you test that it works.
4. Write a static method altCaps that accepts a String as a parameter and returns a String with the capitalization altered such that the even letters are all in lowercase and odd letters are all in uppercase. Non-alphabetic characters don't count as letters (use Character.isLetter(char c) which will return true if the parameter is a letter). String's charAt method will help with this.

altCaps("Hey!! THERE!") >>> hEy!! ThErE!

altCaps("aaaaa") >>> AaAaA

1. **Inside Tester's** main **method,** print a call to each of the methods you just wrote (to test them), rather than calling them directly through BlueJ's workbench (if you didn't already).
2. Make a new class called Player that represents an individual player on a team, that has the following:
   1. String name and int number instance variables.
   2. A default (no-parameter) constructor that initializes name to "Default" and number to -1.
   3. A two-parameter constructor that initializes name and number to the value of the parameters.
   4. An instance (non-static) method String playerInfo() that returns a printable String containing the info for a Player object, in the form of:

"Player: <name>, #<number>"

where the angle brackets < and > indicate that you should use the value of the instance variable (rather than literal text).

1. In the Tester class' main method, construct two Player objects, one using a call to the default constructor and one using a call to the two-parameter constructor (supply any values you want), and print a call to playerInfo for each object (to test that your class works).

**Polynomials**

Objects are one the most difficult to understand topics in a first-year Java programming course, and their usefulness can take some time to sink in – we'd better start now. Object oriented programming is a big part of AP computer science! A (very) brief refresher:

***A class is a custom type*** *that you define – a bundle of variables and methods. In addition, a class is a blueprint for objects. Objects have all the fields and methods defined in the class, but with actual values.*

*An object is a specific* ***instance*** *of the class – one where all the declared variables have values. Much more info can be found in the powerpoints!*

Polynomials are expressions that appear in *many* areas of math and science; a polynomial is a series of coefficients, variables, and non-negative exponents (e.g. x2 – 4x + 8). In this lab, we will only be considering single-variable quadratic polynomials.

1. Make a new class called [Monomial](https://en.wikipedia.org/wiki/Monomial). The Monomial class represents a *single term* in a Polynomial (e.g. x2 could be a Monomial object). This class has the following:
   1. Instance variables:
      * double coefficient – the red portion of this term: **4.97**x2
      * int exponent – the red portion of this term: 4.97x**2**
   2. Method(s)
      * public Monomial(double coeff, int exp) – constructor, initializes instance variables to given values
2. Make another class called [Polynomial](https://en.wikipedia.org/wiki/Polynomial). The Polynomial class will represent a quadratic polynomial (e.g. x2 + 2x – 7); in other words, a Polynomial is comprised of three Monomials.
   1. Instance variables
      * Monomial firstTerm – first term in the Polynomial, e.g. x2
      * Monomial secondTerm – second term in the Polynomial, e.g. 3.17x
      * Monomial thirdTerm – third term in the Polynomial, e.g. 20.12
   2. Methods
      * public Polynomial() – "default" constructor, initializes all the Monomials objects to null.

/\* When a Polynomial object is created with this constructor, the Monomial objects *firstTerm*, *secondTerm*, and *thirdTerm* are **declared only.** Each term would need to be initialized *after* the Polynomial is initialized (instantiated)\*/

* + - public Polynomial(Monomial first, Monomial second, Monomial third) – parameterized constructor, initializes the Polynomial's instance variables to reference the values of the parameters.

/\* This constructor requires three already constructed Monomial objects. Just like with primitive types, the constructor will use the supplied values to initialize this object's instance variables \*/

* + - public double evaluate(double x) – evaluates the polynomial for the given value of x (plug in the parameter value x for all occurrences of x in the polynomial).

/\* Example: for a polynomial f(x) = x2 + 2x – 7, evaluating f(5) would give the following result: 52 + 2\*5 – 7, or 28 \*/

* + - public String toString() – returns a String representation of this Polynomial the format shown below. You may assume each term will be non-null; also, don't worry about the signs, e.g. "+ -3" is fine.

3.5x^2 + 0.6x + 13.6

To test your classes, copy the following into a new class called Runner with a main method. Fix any mistakes you may have. If you get an error you can't decipher, check the **FAQ** first!

Monomial first = new Monomial(1.0, 2);

Monomial second = new Monomial(2.0, 1);

Monomial third = new Monomial(3.0, 0);

Polynomial poly = new Polynomial(first, second, third);

System.out.println(poly.evaluate(4.5)); //should print: 32.25

System.out.println(poly.toString()); //should print: 1.0x^2 + 2.0x + 3.0

Using objects of the Polynomial and Monomial classes you wrote, solve the following problems (in the Runner class):

1. Evaluate the polynomial f(x) = 4.32x2 – 2.12x + 3.67 for f(2).
2. Suppose the concentration (in parts per million) of a new medicine in the bloodstream after t hours is given by the polynomial -0.05t2 + 2t + 2. Write the code to print a "table" that shows the concentrations each hour for 8 hours, to ensure proper dosing.
3. If a projectile is fired vertically into the air, its position at any time is given by the following ballistic trajectory formula (i.e. polynomial): y = y0 + v0\*t – 1/2\*G\*t2 where y0 is the initial launch height, v0 is the initial launch velocity, and G is the gravitational constant (9.81 m/s2).

If a projectile is launched at 50 m/s from an initial height of 3m, find the time (in increments of 0.1 seconds) when the projectile passes a height of 100m.

**Setting up your H: drive for the year**

Create a folder titled "AP CS" on your H: drive. You will use this folder to store all the labs you'll work on throughout the year. At this point, you should have one folder in your AP CS folder, called *Lab01-Polynomials*. In BlueJ, go to **Project > Save as** to save a copy of the project in your AP CS folder.

When you're done, you will submit your work to the ***StudentWork*** folder, a special folder on the Student shared drive (the S: drive) that students can write to but not read from. Save a *copy* of your Lab01 project, renamed with YOUR NAME, in the following location:

S:\StudentWork\Hanna\<Your class>\Lab01-Polynomials.

**Click** [**here**](https://www.youtube.com/watch?v=sRIqxiikfjI) **for a video that will walk you through the submission process.**