**Name**

**Advanced Programming in Java**

**Lab Exercise 11/6/2019**

**Pass-by-value**

The actual parameter (or argument expression) is fully evaluated and the resulting value is ***copied*** into a location being used to hold the formal parameter's value during method/function execution. That location is typically a chunk of memory on the runtime stack for the application (which is how Java handles it), but other languages could choose parameter storage differently.

**Pass-by-reference**

The formal parameter merely acts as an *alias* for the actual parameter. Anytime the method/function uses the formal parameter (for reading or writing), it is actually using the actual parameter.

**formal parameter** — the identifier used in a method to stand for the value that is passed into the method by a caller.

For example, amount is a formal parameter of processDeposit method

**actual parameter** — the actual value that is passed into the method by a caller.

For example, the 200 used when processDeposit is called is an actual parameter. Actual parameters are often called **arguments**

**In general, primitives are passed by value and objects (including arrays) are passed by reference**

State what’s printed for each *println* in the code below:

public static void main(String args[ ])

{

MyClass theObj = new MyClass( );

theObj.gravy = 107.43;

String s = “hello”;

int xray[] = {1, 2, 3, 4, 5};

double stuff = 97.4;

myMethod(stuff, theObj, xray, s );

System.out.println(stuff); // Problem 1: \_\_\_\_\_\_\_\_\_\_\_\_\_

System.out.println(theObj.gravy); //Problem 2: \_\_\_\_\_\_\_\_\_\_\_\_\_

System.out.println(xray[2]); //Problem 3: \_\_\_\_\_\_\_\_\_\_\_\_\_

System.out.println(s); //Problem 4: \_\_\_\_\_\_\_\_\_\_\_\_\_

}

public static void myMethod(double stuff, MyClass anObj, int a[ ], String s)

{

stuff = 13.1;

anObj.gravy = 10.001;

a[2] = 100;

s = “good bye”;

}

**1. Project… Pass the Gravy, Please**

Create a new project called *PassingValues* and put the two above methods in a *Tester* class. Then create a class called *MyClass* having no constructor, no methods and only one *public static* *double* data member called *gravy*. The *gravy* instance field (data member) should only be declared, not initialized in the *MyClass* class.

Run the *main* method and confirm your answers in the exercise above.

**2. Project Multiple Returns from a Method.**

Create a new project called ArrayStats. Your main method should create an array of 100 random integers ranging from 1 to 1000. Your main method should call another method calcStats and is passed the array of random integers. The calcStats method should calculate the average, maximum value and minimum value of the array values and return those values back to main. The main method will then print them out the average of the array as well as the max and min values. In order for this to work, in addition to passing the array to the calcStats method, you will need to pass average, maxValue, and minValue as reference parameters. Remember, primitive data types such as int and double are passed by value. In order for something to be passed by reference, it must be either an array or an object. Consider creating a class to create objects from to store your value.