CS330: Programming Language Project (PLP)

Assignment 5: Functions and Parameter Passing

In programming, functions can be defined as "self-contained" sequences of code that are program to execute a specific task; it is a way to combine multiple actions into one single process (Germain). Functions in programming work in a similar way to functions in mathematics; as they both have inputs and outputs. However in programming, these inputs take in datatypes and process it order to return a result (Myers, 2010). I have created an example below to illustrate the similarities of how functions in mathematics are parallel with functions in programming.

$$f: X \rightarrow Y$$

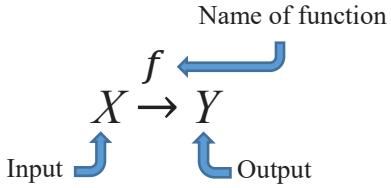
This is how a function is traditionally defined in mathematics:

A function from X to Y assigns each element of X to each element of Y.

However the notation of a function can also be written as:

$$X \overset{f}{\rightarrow} Y$$

Both notations, $f: X \rightarrow Y$ and $X \overset{f}{\rightarrow} Y$ are both equivalent to one another (Epp, 2019, p.384). I have written the second notation because it can be easier to visualize how a function also works within a program, which would then better illustrate the parallels of how functions work in both mathematics and in programming.



Think of X as your initial input, which is typically articulated through arguments or parameters which would then return the now transformed input of X, known as an output Y. The process of X changing to the value of Y in this case is a function called f.

"Functions" in Java:

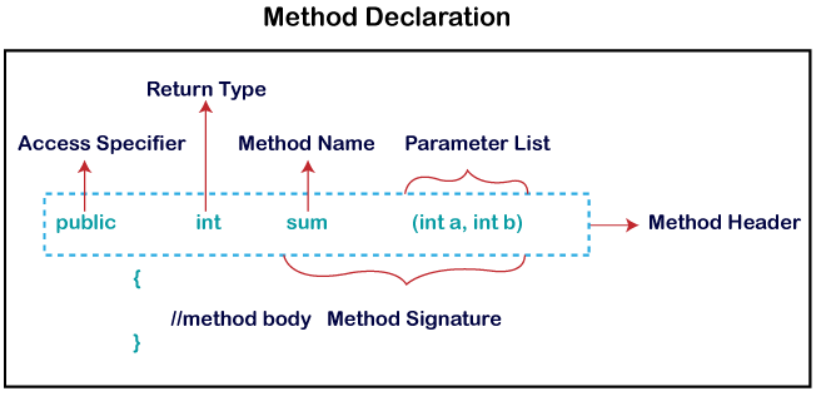
In the case of Java, what is typically referred to as a function in other programming languages is known as a method. What makes a method *a method* specifically is that it is a function that belongs to a class, and in Java "every function belongs to a class" (Myers, 2010). Information is inserted into methods in the form of parameters which act as a variable within the method (w3schools). I have inserted an example of a method in Java that is equivalent to the mathematical function $f(x) = y$.

```
public int f (int x) //method is called f
{                    //the parameter x is specified in the parentheses
   int y;  // int y declared in the body of the method
   y = x; //declares that y is equal to the parameter x
   return y; //the output that is returned
}
```

Discussion Questions:

1. What is the syntax for declaring a method in your language?

   The general declaration of a method in Java has six components: *access specifier, return type, method name, parameter list, method body, and method signature* (Javapoint). These six components in one collective are known as the *method header* (Javapoint).



   Access Specifier: The Access Specifier defines where your method can be accessed from, there are four types of access specifiers: *public, protected, private,* and *default*.

   Return Type: The return type would be the value that is either returned by the method via data type such as *int* or *double* for example. In the case that the method does not return a data type the return value is voided instead (Javapoint).

   Method Name: This would be the name of the method. When naming methods it is good practice to follow standard naming conventions and make your method name that is consistent and understandable (Bloch, 2018, p. 189).

   Parameter List: The parameter list is a list typically with data types and are separated by commas. However in the case that no data types are present then the parentheses are empty.

   Method Body: The method body contains all the directions the method will execute and it is surrounded by curly braces (Javapoint).

   Method Signature: The method signature are just the aspects of the method name and parameter list.

2. Are there any rules about where the method has to be placed in your code file so that it can run?

   A method must be declared inside of a Java class, this is because that is the one defining factor that makes a method a method rather than just a function. As previously stated a method can briefly be defined as a function within a class.

   It is important to note that in Java more than one method can be invoked within the same class, they can even have the same name. What would differentiate two methods with the same method name within the same class would be the unique sets of parameters within each method header, the execution of these methods strictly depend upon the parameters that are passed (TutorialsPoint).

3. Does your language support recursive methods?

Java supports recursive methods. This is useful within the Java language, or for any language, because the technique of recursion helps break-down one complex problem into simpler problems that are easier to solve (GeeksforGeeks, 2019). What happens throughout the process of recursion is that within the code there is the use of a function that calls upon itself. Down below is a classic example of recursion and I will explain how the computer views each process of the step.

```
static void recur(int evens) {
    if (evens < 1)
        return;
    else {
    System.out.printf("%d ", evens);
    recur(evens - 2);
    System.out.printf("%d ", evens);
        return;
    }
}

public static void main(String[] args) {
    int evens = 10;
    recur(evens);
}
```

View entire source code here

The output of this function will illustrate the recursive process on how the method moves down and works it way back up evenly to 10.

```
//Output: 10 8 6 4 2 2 4 6 8 10
```

4. Can methods in your language accept multiple parameters? Can they be of different data types?

For Java, practically speaking, you can have as many parameters as you like within your method, you can have 2 parameters, 12, or even 87! However, technically speaking the number of parameters is limited to only 255 within a method declaration (Sun Microsystem - class).

Despite having the ability to add a numerous amount of parameters to the header does not mean you should, it is actually good practice to only aim for four parameters or fewer within your parameter list. The main reason for this practice is for the ease of the programmer, it would be redundant work for the programmer to memorize a large list of parameters (Bloch, 2018, p. 189).

On another note, Java is able to accept different data types within the parameters list from the method header. In the example below, I have created one method that includes two different data types for addition within the parameter list, these two data types are an int and a double.

```
public class Main
{
   public static void main(String[] args)
   {
     System.out.println("We are going to add an int and a double!”);
      double z= add(10,1.1);
      System.out.println("Sum : " + z);
   }
   public static double add(int x, double y)
   {
      return (x+y);
   }
}
```

View entire source code here

What happened throughout this code is that with the use of a method, the double z was able to be referred to with the use of addition.

5. Can methods in your language return multiple values at the same time? How is that implemented? If not, are there ways around that problem? What are they?

   Java does not support returning multiple values at the same time, however there are ways around this structural barrier within the programming language (GeeksforGeeks, 2018).

   Utilizing an array:

   It is possible to return an array in Java in a feasible manner, with the use of arrays we can output more than one types of value.

```java
static int[] total(int a, int b)
{
   int[] sum = new int[2];
   sum[0] = a + b; //addition
   sum[1] = a - b; //subtraction

   return sum; //would return the elements from the array sum
}
public static void main(String[] args)
{
   int[] sum = total(200, 100);
   System.out.println("Sum = " + sum[0]);
   System.out.println("Sum = " + sum[1]);
}
```

   View entire source code here

   However there are other methods you can utilize in Java as an OOP loop hole to display multiple values from a method. Others include utilizing tuples, container classes, name values in a map, and even third person libraries (GeeksforGeeks, 2018)!

6. Is your language pass-by reference or value? Check your code against outside sources in case there is anything tricky going on (like in Perl).

   Before concluding whether Java is either pass-by reference or pass-by value, I will briefly describe what both terms mean in order to understand where Java lies. It is important to note that all the methods and variables created in Java are stored in the Java Virtual Machine (JVM) stack (Sun Microsystem -stucture).

   Pass-by value:

   The term pass-by value means that you are actually creating a copy of the parameter's value within the JVM stack (Zander). So you are not passing the actual variable into the contents of the stack just its value.

   Pass-by reference:

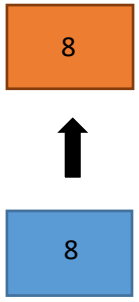
   The term pass-by reference means that a copy of the address of the actual parameter is stored within the JVM stack (Zander). So anything that is passed-by reference would operate on the actual variable itself since it is linked to the address of the original parameter.

   This is an illustration of what pass-by value and pass-by reference looks the one down below:

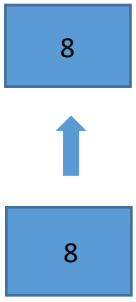
pass-by value and pass-by reference illustration

From this case an orange value has been created that has the same value by copy as the original blue value.

Any changes made to the first will NOT show in the second



pass-by value



pass-by reference

From this case a new blue value is created by the reference of the original blue value.

Any changes made to the first will show in the second

As a result, it is clear that Java is strictly pass by value because it creates a copy of the referred parameters and passes the value to the methods. To see an example view the source code from this link of an example!

Works Cited

Bloch, J. (2018). *Effective Java*. Boston, MA: Addison-Wesley.

Epp, S. S. (2019). *Discrete mathematics with applications*. Boston, MA: Cengage Learning.

GeeksforGeeks. (2019, June 18). Methods in Java. Retrieved October 12, 2020, from

https://www.geeksforgeeks.org/methods-in-java/

GeeksforGeeks. (2019, April 23). Recursion in Java. Retrieved October 16, 2020, from https://www.geeksforgeeks.org/recursion-in-java/

GeeksforGeeks. (2018, September 13). Returning Multiple values in Java. Retrieved October 16, 2020, from https://www.geeksforgeeks.org/returning-multiple-values-in-java/

Germain, H. (n.d.). Functions. Retrieved October 12, 2020, from

https://www.cs.utah.edu/~germain/PPS/Topics/functions.html

Javapoint. (n.d.). Method in Java. Retrieved October 16, 2020, from

https://www.javatpoint.com/method-in-java

Myers, B. (2010). Java Methods. Retrieved October 12, 2020, from

https://www.cs.fsu.edu/~myers/cgs3416/notes/methods.html

Reid, K. (n.d.) Java Method Arguments. Retrieved October 16, 2020, from

http://www.cs.toronto.edu/~reid/web/javaparams.html

Stanchfield, S. (2001, May 16). Java is Pass-by-Value, Dammit! Retrieved October 16, 2020, from

http://www.javadude.com/articles/passbyvalue.htm

Sun Microsystem (n.d.). The class File Format. Retrieved October 16, 2020, from

https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-4.html

Sun Microsystem. (n.d.).The Structure of the Java Virtual Machine. Retrieved October 16, 2020, from

https://docs.oracle.com/javase/specs/jvms/se6/html/Overview.doc.html

TutorialsPoint. (n.d.). Can we define multiple methods in a class with the same name in Java?. Retrieved October 16, 2020, from

https://www.tutorialspoint.com/can-we-define-multiple-methods-in-a-class-with-the-same-name-in-java

W3schools. (n.d.). Java Method Parameters. Retrieved October 12, 2020, from

https://www.w3schools.com/java/java_methods_param.asp

W3schools. (n.d.). Java Methods. Retrieved October 12, 2020, from

https://www.w3schools.com/java/java_methods.asp

Zander, C. (n.d.). Function pass by value vs. pass by reference. Retrieved October 16, 2020, from

https://courses.washington.edu/css342/zander/css332/passby.html