Java Programming
Lab FP 3

Function as return values

1. Lab objectives

This initial lab is designed to help get started with the basics of using functions as return to other functions.

2. Lab Setup

For this lab, you should probably create a new project and new main class.

3. Lab Solutions

Since the demos are based on the labs, if you are having problems getting started, refer to the demo code in the repository to give you some hints.

3. Create the dispatch table

This is also a form of meta-programming but in this case we have to choose between several different implementations of a function depending on some condition in the run time environment. In it's simplest form, a dispatch table is a function factory that, when invoked, returns the appropriate function to be applied.

Note that this is the converse to the last lab. In lab 2, the meta-function was passed code to execute and return a result. In this lab, the dispatch table is passed some data and returns the function that the caller can execute. Dispatch tables are have been common for decades in system programming, especially operating system design, but used something called function pointers rather than Lambda functions.

The example in the lab is very, very simplistic. What it does is takes an integer representing and exponent and returns a function that can be used to raise a value to that exponent. Notice that only the powers 1,2,3 are supported and anything else is raised to the power 0.

A common use for this is to have a bunch of different transformation of the data as the alternative method bodies but we don't know until run time which transformation we need to apply. We can use a dispatch table to return the appropriate transformation when we know during execution which transformation we need.

```
public static Function<Integer,Integer> f(int power) {
    switch (power) {
    case 1:
        return x -> x;
    case 2:
        return x -> x * x;
    case 3:
        return x -> x * x * x ;
    default:
        return (x) -> 0;
    }
}
```

4. Test the function

In the code below, notice the second output line is a common notation, we just use the function we get back from f(2) without assigning it to an intermediate value. It is much more in the functional programming style as well.

```
public static void main(String[] args) {
    Function<Integer,Integer> newFunc = f(3);
    // whole bunch of code maybe.//
    System.out.println("Cube: " + newFunc.apply(3));
    System.out.println("Square: " + f(2).apply(3));
}
```

End of Lab