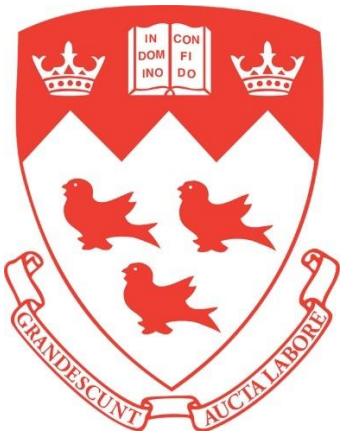
McGill University



ECSE 321 – Project #5

Static Quality Assurance and Release Engineering

Prepared by:

Brent Coutts (260617550)

December 2nd, 2016

1 Code Quality

1.1 Brief Description of the Method's purpose

The method 'doStuff' serves the purpose of determining the Fibonacci number in a specified index of the Fibonacci sequence. The first few terms of the Fibonacci sequence are provided in Table 1 below to show the pattern; however, the sequence continues infinitely. The Fibonacci sequence is a pattern of numbers where each number is equal to the sum of the previous two numbers. This relationship is seen in Equation 1, where F_n is the Fibonacci number 'F' in index 'n' for example.

It is an important consideration that the code labels the indices incorrectly for the sequences. Clearly this is not its intended purpose, and the code's purpose will be discussed as if it functioned correctly. This rationale, along with other assumptions, are further discussed in the assumptions heading of the report.

The index that the program will solve for is the input value 'number1thatdoesstuff'. If this index is less than or equal to two, the Fibonacci number is equal to one. Otherwise, the program uses three other variables, 'number2thatdoesstuff', 'number3thatdoesstuff', and 'number4thatdoesstuff', in a 'for' loop as continually updating Fibonacci number placeholders. These variables are used calculate proceeding Fibonacci number terms in the sequence. Specifically, the next number in the sequence ('number2thatdoesstuff') is determined using the previous two numbers in the sequence ('number3thatdoesstuff' and 'number4thatdoesstuff'). This leads to the 'number2thatdoesstuff' value being returned once the desired index of 'number1thatdoesstuff' is reached.

Table 1: Fibonacci Sequence with Index Values

Index	1	2	3	4	5	6	…
Fibonacci Number	1	1	2	3	5	8	…

Equation 1: Fibonacci Formula

$$F_n = F_{n-1} + F_{n-2}$$

1.2 Comments on Code Quality

The code quality is certainly capable of improvement. Various factors of the code make it difficult for other programmers to understand. Firstly, the code lacks meaningful variable names. When initializing variables, it is good practice to name the variable something meaningful to its use. The current variable names of 'numberXthatdoesstuff', are long and do not assist in understanding the code. Furthermore, it is slightly difficult to differentiate between each variable legibility wise as they are all named very similarly. This similarity is furtherly impractical as the some of the variables serve more different purposes than their names would imply.

Another room for improvement in the code, is adding succinct comments to the method. These comments can be placed before important sections in the code to describe the logic that is occurring. A few simple comments would greatly assist other programmers in understanding the function of the code.

The code is also not indented in a neat and intuitive manner. This makes it difficult for the flow of the program to be understood. Some lines have multiple, non-directly related, operations being performed. As multiple operations are on the same line it is easy to overlook the additional operations of the line, as a programmer would commonly assume that one operation is occurring per line. This jumble of indentation causes 'for' loops and 'if' statements to be laid out in a manner where it is tough to understand their beginning, contained operations, and ending points. It would also be prudent to remove any excess spaces between variables and punctuation or mathematical operators. These suggestions would improve the organization and flow of the code.

This comment about the code relates to the perceived purpose that it serves. The way the code is currently laid out, the 'for' loop will not perform and iterations as a return statement is contained within the loop. This will result in Fibonacci numbers smaller than the correct values to be returned. Furthermore, the indexing applied to the numbers is slightly incorrect. The first Fibonacci number is one and the second Fibonacci number is also one. The program currently returns an Fibonacci value from an index that is one too large due to this oversight. A last small consideration is returning zero when the value input is a negative integer or equal to zero as the sequence begins from the first number.

Finally, as the code is written, the method will only return the Fibonacci number in the index of interest and not the entire sequence. However, as the numbers get much larger, calculating one value in this sequential way is much slower. Either the code should be altered to return the entire sequence up to and including the index of interest, or a more efficient method using the golden ratio would be advisable (Equation 2). This equation allows the n^{th} Fibonacci number (X_n) to be calculated using a simple formula instead of a computationally intensive summation sequence. This comment will not be implemented in the rewritten code as that would drastically alter the code past the point of the assignment.

Equation 2: Binet's Formula

$$X_n = \frac{\emptyset^n - (1-\emptyset)^n}{\sqrt{5}} \quad , \quad \emptyset = 1.618034$$

1.3 Rewritten Code Based Upon Suggested Comments

```java
public int calculateIndexedFibonacciNumber (int fibIndex) {
    // This method will calculate the Fibonacci number from a given index.
    // Note that indices begin from 1 as it is a number rather than a java index.

    // Variables are initialized in one location for easy alteration.
    int fibNumber = 1;
    int fibNumberLess1;
    int fibNumberLess2;

    // Returns 0 if invalid input (index less than 1) is provided.
    if (fibIndex <= 0) {
            return 0;
    }

    // By convention first two indices have a fibNumber of 1.
    if (fibIndex <= 2) {
                return fibNumber;
    }

    // A loop is performed using the Fibonacci algorithm to determine the fibNumber of the desired index.
    // The next Fibonacci number is calculated by the sum of the previous two Fibonacci numbers.
    fibNumberLess2 = 1;
    for (int i = 3; i < fibIndex; i++) {
            fibNumberLess1 = fibNumber;
            fibNumber = fibNumber + fibNumberLess2;
                fibNumberLess2 = fibNumberLess1;
    }

    // The fibNumber in the fibIndex of interest is returned.
    return fibNumber;
}
```

Figure 1: Improved Assignment Code

2 Build Specification

The build specification was performed for the Computation.zip source code using Ant. These xml files are present in the project submission zip folder.

3 Assumptions

- The code in the assignment description is missing a '}' bracket and it is unsure if the return statement is inside or outside of the loop. If the return value was inside the 'for' loop, the loop would be exited immediately. Thus, it is reasonable to assume that the return statement appears after the loop in the code. With this assumption, the code clearly serves the purpose of only providing the Fibonacci number in a sequence index of interest and not the whole sequence up to that index.
- The code solves for the Fibonacci number in one index greater than the index of interest input into the program. This is presumed to be a mistake and the code is discussed and altered with this change in mind.
- The first value of the Fibonacci sequence is either zero or one. The common convention was chosen that the sequence begins at 1.