Name: Toshiro Mendoza ID: 834872958 SOFTENG282 – Assignment 1

--------------------------------------------------------------------------------------------------------------------------------------

**Program Documentation**

1) ----------------------------------------------------------------------------------------------------------------------------------

The program description is outlined as: Given two positive integers A and B, write a program to calculate the greatest common divisor (GCD) of the sum of the first A Fibonacci numbers and the sum of the first B Fibonacci numbers. The program can be in either C or Java.

Input: Two positive integers A and B.

Output: The GCD of the sum of the first A Fibonacci numbers and the sum of the first B Fibonacci numbers.

**FibonacciGCD.java**

The programming language I chose to do this task is Java. I simply created the actual program that contains the algorithm under ‘FibonacciGCD.java’. I call the method that does the algorithm “computeGCD()”. For now, this is represented simply, before the logic is added:

public class FibonacciGCD {

    public FibonacciGCD() {}

public int computeGCD(int a, int b) {

return -1;

    }

}

It is easier to first find the sum of the first N Fibonacci numbers, thus I first created a different method which first finds this very thing. I call this “sumOfFirstNFibonacciNumbers()”.

private static int sumOfFirstNFibonacciNumbers(int n) {

        // Initialize tracking of Fibonacci numbers

        int sum = 0;

        int a = 0;

        int b = 1;

        // Loop to find first n Fibonacci numbers and add them to sum

        for (int i = 0; i < n; i++) {

            sum += a;

            int temp = a;

            a = b;

            b = temp + b;

        }

        // Return sum

        return sum;

    }

Next, it would also be very helpful if there is a method which finds the greatest common divisor between two numbers. Following the Euclidean Algorithm, we can create a recursive method. I called this method “getGCD()”.

A screen shot of a computer program

Description automatically generated

With these two methods in our grasps, it becomes very easy to find the GCD of the sum of the first A Fibonacci numbers and the sum of the first B Fibonacci numbers.

    public static int computeGCD(int a, int b) {

        // If a or b is 0, return 0

        if (a == 0 || b == 0) {

            return 0;

        }

        // Find sum of first a and b fibonacci numbers

        int sumA = sumOfFirstNFibonacciNumbers(a);

        int sumB = sumOfFirstNFibonacciNumbers(b);

        // Return gcd of sum of first a fibonacci numbers and sum of first b fibonacci numbers

        int result = getGCD(sumA, sumB);

        return result;

    }

**DisplayFibonacci.java**

I also made DisplayFiboacci.java class. This class will contain the bulk of what will be shown on the program. That is, apart from the printed texts from FibonacciGCD.java. Thus, the main.java will have to simply call this class’s method “run()” to run the entire program.

A screenshot of a computer program

Description automatically generated

The run() method simply calls on the other methods to run them in the correct order, by repeating the menu() method until the user uses the EXIT command.

A screen shot of a computer program

Description automatically generated

The printHeader() and printFooter() methods simply prints out texts that indicates the start and end of the program. They are only made to make the program cleaner to look at; thus the ascii art for the header.

A screen shot of a computer

Description automatically generated

A black screen with orange and white text

Description automatically generated

The menu() method asks the user which command they want to use. The repeating nature of this method in the run() method makes it possible for the user to use this program again and again until satisfied of calculating all they need to calculate. The two available commands are:

CALCULATE: Calculate the GCD of the sum of the first A and B Fibonacci numbers.

EXIT: Exit the program.

A computer screen shot of a program code

Description automatically generated

The getNumberInput() method is the method that obtains the desired A and B needed to calculate the GCD of the sum of the first A and B Fibonacci numbers. This method specifically needs a string input which specifies whether the number being obtained is for number “A” or “B”.

A screen shot of a computer program

Description automatically generated

Then lastly, the calculate() method uses the getNumberInput() function to obtain A and B, then uses it to calculate the desired output by calling the FibonacciGCD class which contains the desired algorithm.

A screen shot of a computer program

Description automatically generated

The Main.java class thus then simply looks like:

A screen shot of a computer program

Description automatically generated

2) ----------------------------------------------------------------------------------------------------------------------------------

Next task to do is the following: Add suitable output commands to the GCD algorithm that show the progress of the algorithm.

We can print into the command line the Fibonacci numbers as they are being added into the sums. This tells us the progress of the sumOfFirstNFibonacciNumbers(). Thus, I rewrote this method as:

A computer screen shot of numbers

Description automatically generated

I also modified the computeGCD() method to track what the sums are and display what the resulting GCDs are.

A screen shot of a computer code

Description automatically generated

3) ----------------------------------------------------------------------------------------------------------------------------------

Next task to do is the following: Add suitable comments to your code that would help a code reviewer.

4) ----------------------------------------------------------------------------------------------------------------------------------

Next task to do is the following: Provide a readme file that explains your solution to a reader with sample output similar to the output shown in the question above and contains other instructions i.e. how to compile and run the program.

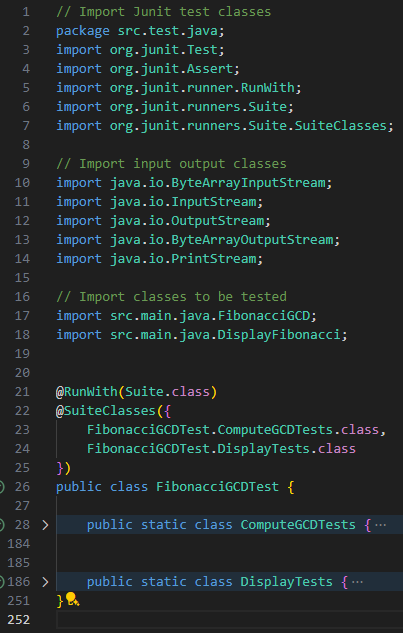
Proper and adequate comments on how the program works are added during the programming process already. The README.md file should be submitted to the same folder as this document.

5) ----------------------------------------------------------------------------------------------------------------------------------

Next task to do is the following: Test your code and document this. Introduce, if useful, bounds for the input.

**FibonacciGCDTest.java**

For the test, I created FibonacciGCDTest.java class. This test is divided into two subclasses. The first tests the algorithm itself. While the second tests what is displayed when commands are used.



**ComputeGCDTests Class**

The first test subclass is called ComputeGCDTests and is composed of 11 test cases.

A screenshot of a computer program

Description automatically generated

The test methods test\_example(), test\_example\_2(), and test\_example\_3() are all test cases where the inputs are valid and checks whether the output GCD of sum of first A and B Fibonacci numbers are correct. The first one being the example given in the task specification.

A computer screen with text and images

Description automatically generatedA computer screen shot of a program code

Description automatically generated

A computer screen shot of a program code

Description automatically generated

The test methods test\_cardinal\_A\_is\_zero(), test\_cardinal\_B\_is\_zero(), and test\_cardinal\_A\_and\_B\_is\_zero(), are all tests which looks on what happens if at least on of the input numbers to the algorithm is 0. This should cause the algorithm to stop as there is no “0th” Fibonacci.

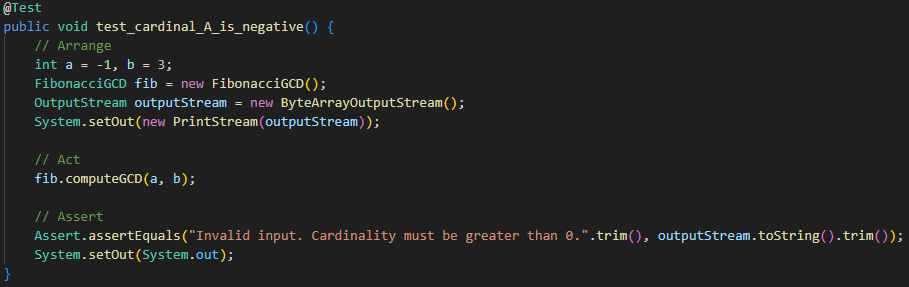
A computer screen shot of a program code

Description automatically generated A computer screen shot of a program code

Description automatically generated A screen shot of a computer program

Description automatically generated

This is followed by the test\_cardinal\_A\_is\_negative(), test\_cardinal\_B\_is\_negative(), and test\_cardinal\_A\_and\_B\_is\_negative(), which are methods that tests what happens to the algorithm if he input is negative. In such cases, the result should be the same when inputs are zero as there is no “Negative Nth” Fibonacci number.

 A computer screen shot of text

Description automatically generated A screen shot of a computer program

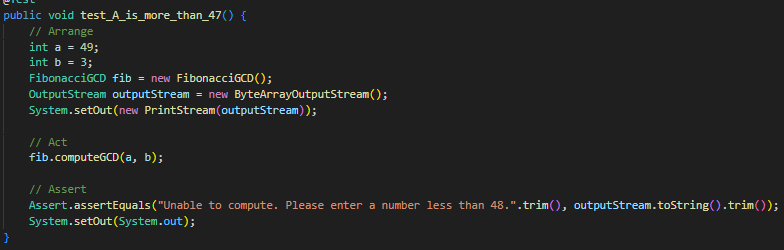
Description automatically generated

Then, there is the test\_B\_is\_greater\_than\_A() method which just checks what happens if numbers A and B are flipped instead of A>B from other tests.

A computer screen with text and images

Description automatically generated

Then lastly there is test\_A\_is\_more\_than\_47() method which just checks that the output is correct when the input is more than 47 as this will cause an overflow if not managed.



**DisplayTests Class**

The second test subclass is called DisplayTests and is composed of 3 test cases.

A screen shot of a computer program

Description automatically generated

The first test method called test\_menu\_exit() tests if the program properly if the EXIT command is selected by the user.

A computer screen shot of a program code

Description automatically generated

The second test method is called test\_menu\_invalid\_command() which tests if the program would show “Invalid command. Please try again…”.

A screen shot of a computer program

Description automatically generated

The last test method is the test\_run\_calculate\_multiple\_times(). This fully uses the run() method instead of just one pass through menu(). This checks if the program would let the user calculate different wanted GCD and be able to exit.

A screen shot of a computer

Description automatically generated

After making sure that all the test cases are passed we can say that the task is complete!

A screenshot of a computer program

Description automatically generated