Fibonacci Sequence

Assignment 2

By:

Alexander Rivera

Submitted to:

Prof. Ian O’Toole

CEN 4802C – Software Integration, Configuration, and Testing

Department of Engineering, Computer Programming & Technology

Valencia College – West Campus

5/21/2023

**Introduction**

In this assignment, students will implement the Fibonacci sequence along with implementing Git into their workflow. Students will document their workflow using screenshots in a separate document.

**Requirements**

* A functioning computer
* A working Integrated Development Environment
* Word processing software
* GitHub account
* Internet connection

**Discussion**

In this assignment, the student implemented two methods within one class. The first method called “main” simply prompts the user to enter a number used to calculate the corresponding value in the Fibonacci sequence. The second method, called “sequence”, is used to calculate the Fibonacci values based on the input received from the main method. Throughout development, the student documented finding via ten screenshots and used version control through GitHub. The main branch of this Fibonacci project can be found using the link below.

Link: <https://github.com/arivera247/CEN4802/tree/main/Fibonacci>

**Validation of Data**

A screenshot of a computer program

Description automatically generated with medium confidence

Figure 1 - Screenshot of successful calculation of the Fibonacci sequence.

A screenshot of a computer

Description automatically generated

Figure 2 - Screenshot showing Main class added to GitHub repo.

A screenshot of a computer

Description automatically generated with low confidence

Figure 3 - Newly created issue logged in GitHub.

A screenshot of a computer

Description automatically generated with low confidence

Figure 4 - Screenshot showing new branch created to address missing documentation issue.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 5 - Screenshot showing Main.java class updated in the MissingDocs branch. Note the status message above stating the branch is 2 commits ahead of the main branch.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 6 - Screenshot demonstrating a newly created pull request for the MissingDocs branch merge into the main branch.

A screenshot of a computer

Description automatically generated

Figure 7 - Screenshot containing side by side comparison of main and MissingDocs branches. Newly added code is highlighted in green within the IDE. For emphasis, newly added code in screenshot was boxed in red using a markup tool.

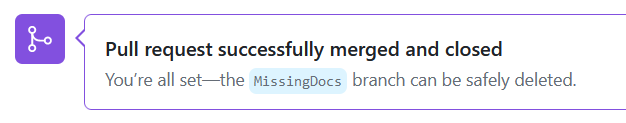


Figure 8 - Screenshot showing both main and MissingDocs branches were merged successfully.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 9 - Screenshot showing remaining branches. The MissingDocs branch is missing since it was deleted after successfully merging the branch to main.

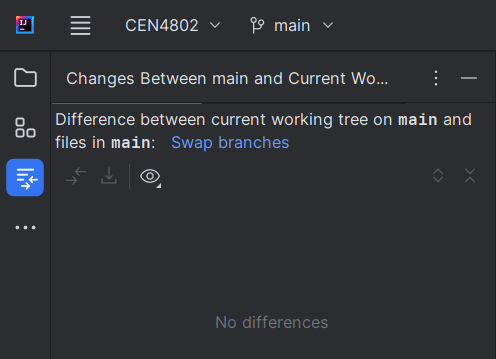


Figure 10 - Screenshot showing difference between local and remote repositories. No differences were found.

**Conclusion**

In this assignment the student was able to implement a Java class with separate methods for input and calculation. While straightforward, the assignment was not without challenges. Throughout the project, the student maintained the Word file containing screenshots within the repository to track changes. Once the separate branch was complete and pulled into the main branch, the Word file continued to be updated within the separate branch logging successful push and merge messages. Upon merging, changes applied to the Word file were rejected via git. The student had to “shelf” the document within the IDE and commit changes afterward. Once the previous document was shelved and replaced with the newer version, changes appeared to sync normally. Moving forward, Word documents must be updated within the main branch or ignored via git. Document changes should not be done in a branch that will ultimately be deleted.

Something else to note, briefly mentioned in the JavaDoc text, was the upper limit of 46 for the n value input into the Fibonacci sequence. This value is the last value before integer sequence(n) value exceeds 2147483647, the upper limit of an integer variable. In future development, proper JUnit tests should be implemented to determine maximum values. In this case, testing was done manually until the student noticed an error in the Fibonacci sequence output, where a negative number was provided. Additionally, the integer data type should be replaced with a long data type variable, depending on the upper values expected for input into the sequence method. Using a long data type variable would allow the Fibonacci sequence to have a maximum value of 9223372036854775807.

Source: <https://www.w3schools.com/java/java_data_types.asp>