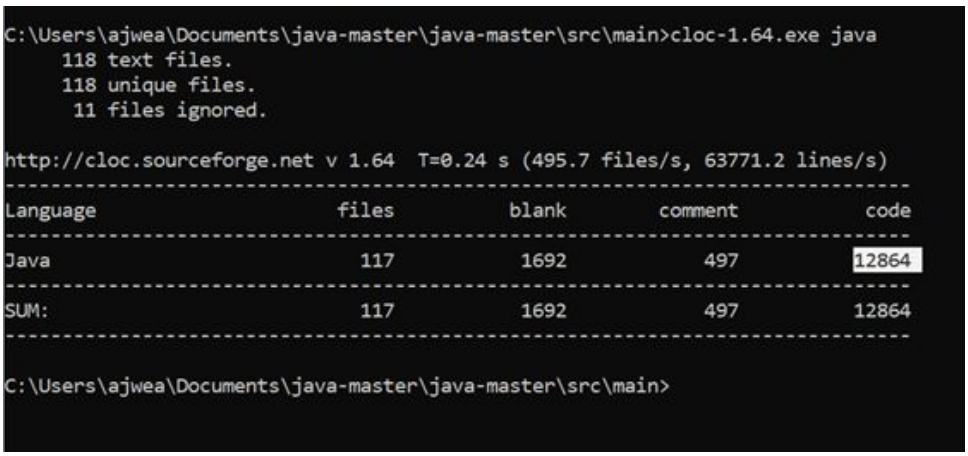
Axel Ward
Cooper Strahan

ESOF 322 Homework 4

QUESTION 1
Part 1

1. The system downloaded is JSON Iterator.
2. It is a parser for JSON files that is an incredibly efficient reader. The goal of the software is to provide a flexible and fast parser for Java and Go, allowing people to cut out a lot of the work of reading a JSON file.
3. The program has around 12,864 lines of code. This was calculated by running the CLOC tool against the codebase, the output of which is shown below.



Part 2

1. OUTPUT:

   Found 6 files that possibly contain design patterns.

   C:\Users\ajwea\Documents\java-master\java-master\src\main\java\com\jsoniter\JsonIterator.java
   Possible patterns: Iterator

   C:\Users\ajwea\Documents\java-master\java-master\src\main\java\com\jsoniter\JsonIteratorPool.java
   Possible patterns: Iterator

   C:\Users\ajwea\Documents\java-master\java-master\src\main\java\com\jsoniter\ReflectionDecoderFactory.java
   Possible patterns: Factory

C:\Users\ajwea\Documents\java-master\java-master\src\main\java\com\jsoniter\extra\NamingStrategySupport.java
Possible patterns: Strategy

C:\Users\ajwea\Documents\java-master\java-master\src\main\java\com\jsoniter\output\ReflectionEncoderFactory.java
Possible patterns: Factory

C:\Users\ajwea\Documents\java-master\java-master\src\test\java\com\jsoniter\extra\TestNamingStrategy.java
Possible patterns: Strategy

2. This tool looks for patterns by comparing them against XML files that have the pattern structure built within them.

3. I think that this tool does this in an extremely smart way. This way has a really low overhead by just parsing through code, as opposed to running it. If I were to implement this kind of tool I would try to write software that runs the target code, and then looks at the stack trace to determine where things are being called. This would show the dependencies between the different pieces of code that could be compared against a logic structure, perhaps an XML template, that would interpret what kind of design pattern the code is closest to.

QUESTION 2:

1. Github
    a. In order to upload the files to GitHub we started by downloading all of the assignments from Google Drive. We put those assignments in a folder on Cooper's computer named ESOF 322. We then created a new repository on Cooper's GitHub. Next we used the terminal to initialize a local git repository, added all of the files to the local repository, committed those files (saved the files as they were in that instance), set the remote origin to the be the repository on Cooper's GitHub and then pushed the committed files to the remote repository.
    b. We added the file SequenceDiagram.png to the remote repository. The commands that we used were:
        i. git add . //To add the file not yet included to the local repository
        ii. git commit -m "Adding Dummy file Sequence Diagram.png" //to commit the file
        iii. git push -u origin master //to push to the remote repository

(base) Coopers-MacBook-Pro:ESOF_322 cooperstrahan$ git add .
(base) Coopers-MacBook-Pro:ESOF_322 cooperstrahan$ git commit -m "Adding Dummy file Sequence Diagram.png"

```
[master bd75131] Adding Dummy file Sequence Diagram.png
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 SequenceDiagram.png
(base) Coopers-MacBook-Pro:ESOF_322 cooperstrahan$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 297 bytes | 297.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/cooperstrahan/ESOF_322.git
   467bc67..bd75131  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
(base) Coopers-MacBook-Pro:ESOF_322 cooperstrahan$
```

Screenshot: