

# SE443- SOFTWARE TESTING & QUALITY ASSURANCE HOMEWORK 2

By Nihar Muniraju (2072857)

## **Part 1: Code smells and refactoring**

### **A. Search for an open-source project (Java) – a total of 200,838 LOC.**

I have chosen Apache-Ant-version 1.10.10, which is an open-source project with a minimum of 200,838 LOC which I found in Designitejava and in Understand Software the LOC is 285,222. It has 91 number of packages, the number of classes are 987, the number of methods is 9239, the files are 1,322 and there are 2 errors with 5% parse Accuracy.

### **B. Compare between DesigniteJava and Understand based on the CK metric results.**

I have compared between DesigniteJava and Understand based on the CK metrics – WMC, LCOM, and DIT.

	UNDERSTAND	DESIGNITE.JAVA
WMC (Weighted Methods per Class)	13.39	12
LCOM (Lack of Cohesion of Methods)	29.30	33
DIT (Depth of Inheritance Tree)	1.92	1
RFC (Response for a Class)	22.72	17
CBO (Coupling Between Objects)	7.82	5

### **Weighted Methods per Class (WMC)**

The DesigniteJava shows better analysing methods than the Understand even though the difference is quite small. But it's got the upper hand when compared to.

### **Lack of Cohesion of Methods (LCOM)**

DesigniteJava provides the value of 33 which is a feature concentration better than understand even though it has more than classes and LOC compared to Designitejava.

### **Depth of Inheritance Tree (DIT)**

Understand has a value of 1.92 and DesigniteJava has a value of 1 which is smaller DIT values are preferred for the betterment of code analysis.

### **RFC (Response for a Class)**

Understand has a value of 22.72 and DesigniteJava has a value of 17 which is smaller RFC because of the number of lines not been able to detect in DesigniteJava the Understand gets more Response for a class in the open-source project.

### **CBO (Coupling Between Objects)**

DesigniteJava has a value of 5 and Understand possess a value of 7.82 which is higher CBO values due to coupling happening between the functions and Objects as the Objects Specified in understand are of a greater number.

### **Conclusion:**

Based on the above comparison of CK metrics between Understand and Designite.java. I have observed that each software provides similar values but understand has more accurate values than Designite.java. Understand could also give more accurate points and better code analysis as compared with the other software and the LOC was better identified by Understand. So henceforth I would say to analyse CK metrics Understand is much better.

## **C. Recommend a set of useful refactoring operations to fix two different types of code smells (antipatterns) identified by DesigniteJava.**

- **Large class (Insufficient Modularization):**

The DesigniteJava shows the number of large classes found are a minimum of 1814 to a maximum found in Understand are 1894. This shows that the large classes are extensively used just to create more functions and objects. But there are so many classes that be combined and brought together like ftp client and ftp host into one ftp and can be modularized which helps save more space and memory and be used for the code betterment of understanding.

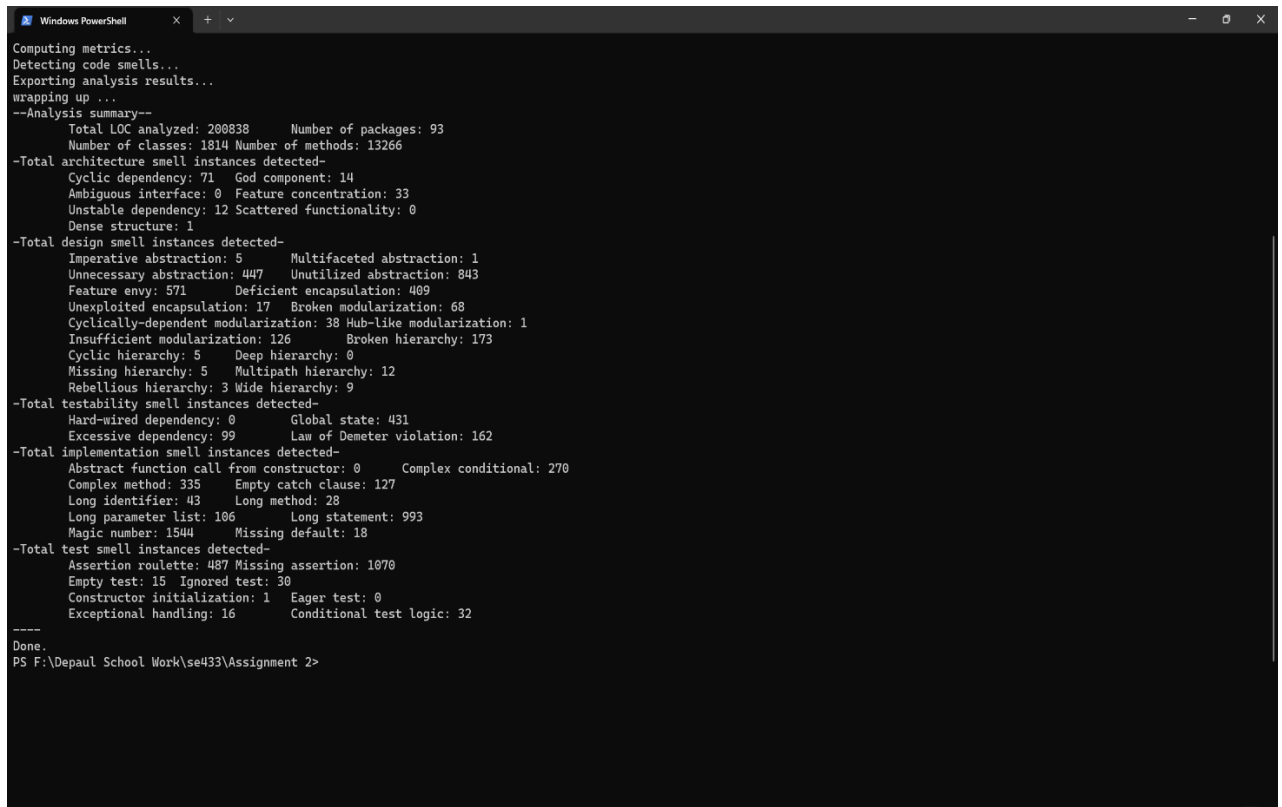
- **Feature Envy:**

There are so many methods here which can be detailed made into one method rather than using different methods to do the same or different amount of work in the FTP CLIENT and FTP HOST. The feature Envy can be used to understand the features of the methods and can be arranged and analysed depending on the usage of the feature required.

- **Comments:**

Comments play an important role as they increase the space and memory of the code with unnecessary readability required. According to Understand there are an approximately of 110,853. We should reduce the number of lines to make the code analysis better for the user.

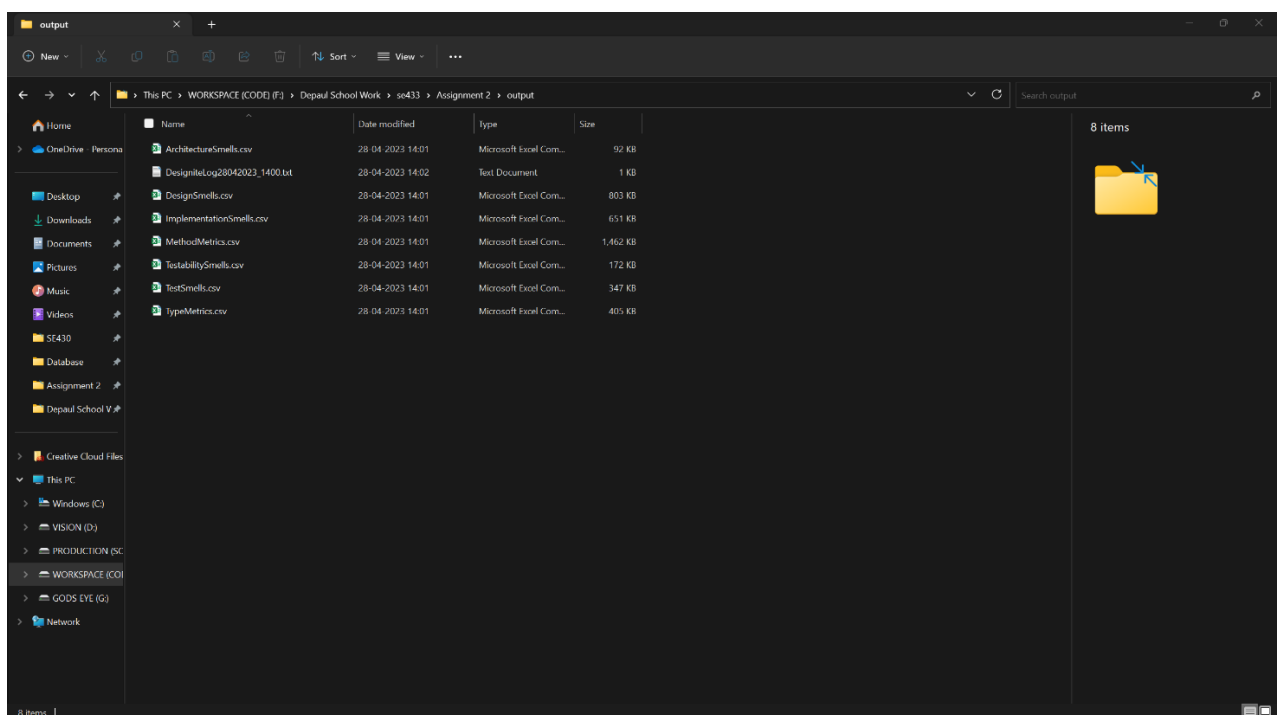
**D. Add a screenshot from the command prompt (batch) and also a screenshot from the output directory in the report.**

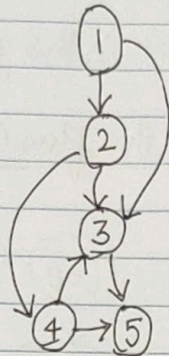


```

Computing metrics...
Detecting code smells...
Exporting analysis results...
wrapping up ...
--Analysis summary--
  Total LOC analyzed: 200838      Number of packages: 93
  Number of classes: 1814 Number of methods: 13266
-Total architecture smell instances detected-
  Cyclic dependency: 71  God component: 14
  Ambiguous interface: 0  Feature concentration: 33
  Unstable dependency: 12  Scattered functionality: 0
  Dense structure: 1
-Total design smell instances detected-
  Imperative abstraction: 5      Multifaceted abstraction: 1
  Unnecessary abstraction: 447  Unutilized abstraction: 843
  Feature envy: 571      Deficient encapsulation: 409
  Unexploited encapsulation: 17  Broken modularization: 68
  Cyclically-dependent modularization: 38  Hub-like modularization: 1
  Insufficient modularization: 126  Broken hierarchy: 173
  Cyclic hierarchy: 5      Deep hierarchy: 0
  Missing hierarchy: 5      Multipath hierarchy: 12
  Rebellious hierarchy: 3  Wide hierarchy: 9
-Total testability smell instances detected-
  Hard-wired dependency: 0      Global state: 431
  Excessive dependency: 99      Law of Demeter violation: 162
-Total implementation smell instances detected-
  Abstract function call from constructor: 0      Complex conditional: 270
  Complex method: 335      Empty catch clause: 127
  Long identifier: 43      Long method: 28
  Long parameter list: 106  Long statement: 993
  Magic number: 1544      Missing default: 18
-Total test smell instances detected-
  Assertion roulette: 487  Missing assertion: 1070
  Empty test: 15  Ignored test: 30
  Constructor initialization: 1  Eager test: 0
  Exceptional handling: 16      Conditional test logic: 32
-----
Done.
PS F:\Depaul School Work\se433\Assignment 2>

```



**Part 2: Cyclomatic Complexity**PART-2 Cyclomatic Complexity.

A) Based on the following flow Graph. Calculate the Cyclomatic Complexity using two different methods.

1)  $V(G)$  The number of regions of the flow graph.

$$\boxed{V(G) = 4}$$

Formula.

$$2) V(G) = E - N + 2$$

$$V(G) = 7 - 5 + 2$$

$$\boxed{V(G) = 4}$$

B) Based on the Cyclomatic complexity value, derive the basis set of test paths.

Path 1:  $1 \rightarrow 3 \rightarrow 5$

Path 2:  $1 \rightarrow 2 \rightarrow 3 \rightarrow 5$

Path 3:  $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$

Path 4:  $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 5$ .



B- Consider the following code.

```

i = 0;
n = 4; // Number of nodes present in the graph.
while (i < n-1) do
    j = i+1;
    while (j < n) do
        if A[i] < A[j] then
            swap (A[i], A[j]);
        end do;
        i = i+1;
    end do;
end do;

```

⇒ Draw the flow Graph of the Code.

⇒ Calculate the Cyclomatic Complexity of the Graph.

A)  $V(G)$  The number of regions of the flow graph  
 $V(G) = 5$

B)  $V(G) = E - N + 2$   
 $V(G) = 13 - 10 + 2$   
 $V(G) = 5$

