**BCPR301 – Advanced Programming**

**Assessment4 Marking Sheet**

Student Name/ ID Simon England

# The compulsory (i.e., ZERO mark if not being provided):

1. You MUST supply a filled self-marking sheet to indicate how many marks you think you can get for each part based on the marking guide provided below.

# Your repository link:

# Marking guide (max 13 \* N marks in total where N = 3):

1. Smell detection (4 \* N marks)
2. Identification of N bad smells in the programs provided. For the sake of learning, please try to identify **different types of bad smells**. (N marks)

Switch Statement, duplication, middle man and long method

1. The location of each bad smell identified (N marks)

Switch statement:

File – uml\_interpreter.py

Line – 15 to 30

Duplication:

File – uml\_interpreter.py

Line – 64 to 72, 74 to 82 and 84 to 92

Middle man:

File – uml\_interpreter.py

Line – 64 to 72, 74 to 82 and 84 to 92

Long method:

File – class\_maker.py

Line – 142 to 163

1. And discussion on the reasons why you think that the ones you identify are bad smells in a concise fashion. Please do not simply copy general reasons from somewhere and paste them in your submission (N marks)

Switch statement:

If I want to add more uml components, I would have to add more else if statements to the code.

Duplication:

The code within the 3 methods have the same logic but slightly different values inside (middle man duplication).

Middle man:

The code reservice a value and holds a value to pass onto another method and returns the value of the method (middle man duplication). Will combine middle man and duplication bad smell into middle man duplication bad smell.

Long method:

There are 21 and the method does multiple things to a string before returning a value.

1. And brief discussion on the refactoring strategies/ approaches you are going to use to remove each bad smell (N marks)

Switch statement:

I will use decompose conditional to remove and’s and or’s (eg line == ‘thing’ && line != ‘other’ into is\_in\_line()) (Source Making, n.d.)

I will use extract method to move code into a easier to edit method (eg if is\_in\_line (): do something… into if is\_in\_line (): do\_something()) (Refactoring Guru, n.d.)

This will reduce the bad smell but won’t get rid of the bad smell

Middle man duplication:

I will use Remove Middle Man to get rid of all middle man and duplicated code. (Refactoring Guru, n.d.)

Long method:

I will use extract method to move code that are in loops, if conditions or code that is grouped together. (Refactoring Guru, n.d.)

1. Tests development (4 \* N marks)
2. To develop a set of tests for the methods/ classes/ modules/ packages encompassed by the bad smells you previously identified (3 \* N marks)
3. Please also use coverage package to generate a HTML report in order to show your code branch coverage **== 100%**. And all tests should be able to be run together by running a single .py file (N marks)
4. Refactoring (5 \* N marks)
5. Identifying the worst smell and the reasons why it is the worst one (N marks)

Worst bad smell:

Switch statement is the worst bad smell because if I need to add more uml components I must add more else if statements to the code.

Second worst bad smell:

Long method is the second worst bad smell because you need to read the method a few times before you understand what it is trying to do.

Least worst bad smell:

Middle man duplication is the least worst because it’s the easiest to understand what it’s doing and how to get rid of it.

1. Version control via a remote repository and testing (N marks)
2. Modification to remove the worst smell and PEP8 validation (2 \* N marks)
3. Effectively evaluations (N marks)

Switch statement:

Bad smell still present but made certain areas easier to understand. Did not solve the problem of more uml components added. This was the problem main problem of the switch statement and it’s still present and couldn’t find away to remove this bad smell.

Long method:

Bad smell almost gone apart from the if else statement left in method\_maker. The if else statement sets two different values and without modifying original code it’s not possible to use extract method.