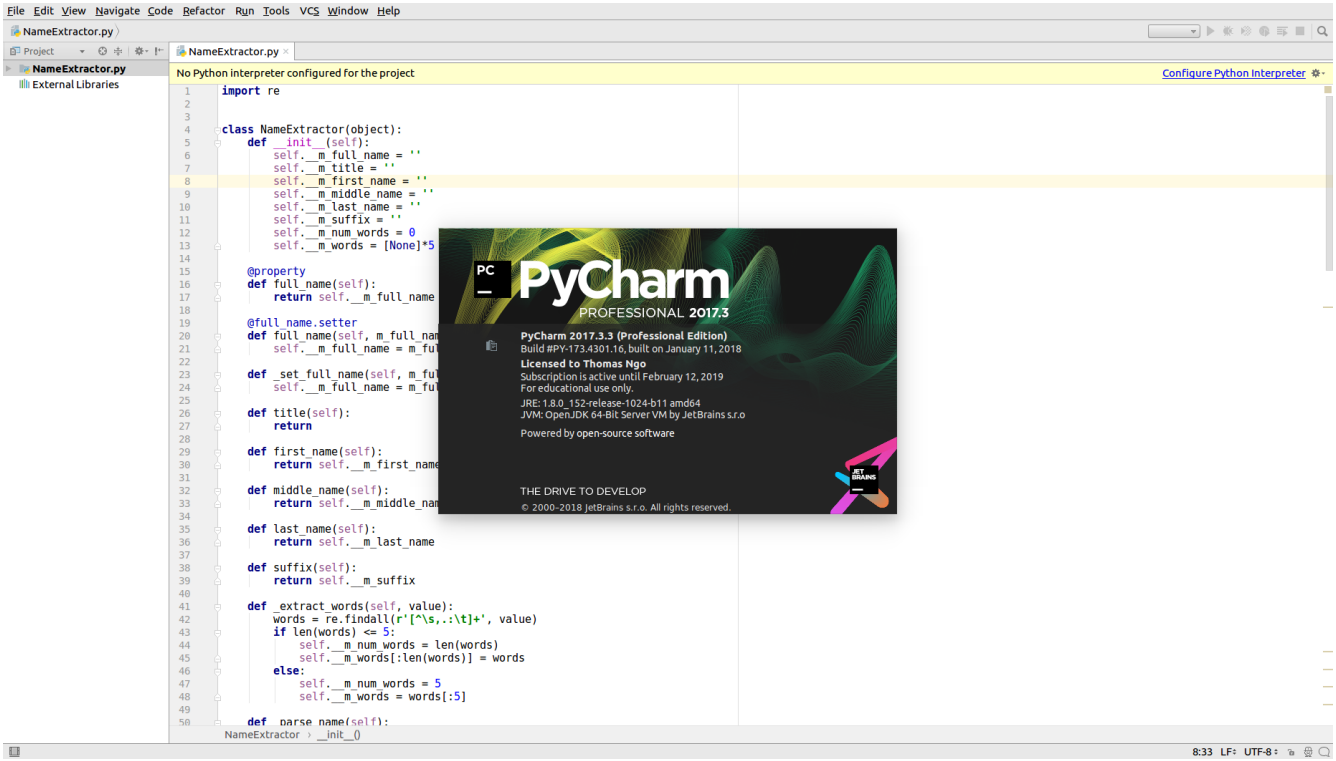
Thomas Ngo
CPSC463-01
Professor Ning chen
Assignment 4

INTRODUCTION:

The main purpose of this paper is about unit testing. With a small portion of the source code like unit, component and function in the same program that is tested. In other words, the notion of unit testing means that every separate part of the product is tested individually to ensure each of them functions and runs as the developer anticipated. This activity makes sure that every unit corresponds correctly to the design specification. After studying the example given by the assignment's instruction, I decided to choose framework named "unittest" and its programming language "python" as my main focus during the course. Also, I have implemented the class NameExtractor along with its members such as ExtractWords, FindTitle, FindSuffix, ParseName in Python language in order to practice implementing unit testing with the testing framework. In addition, even though they are not the requirements in this assignment, I also implemented some extra functions like FirstName, MiddleName and LastName to see check if my implementation is correct or not. With that said, I provided the screenshot of my IDE for python,which is Pycharm and the source code below.

Screenshot of Pycharm on Ubuntu 16.04



Source code:

NameExtractor.py

```
import re
class NameExtractor(object):
    def __init__(self):
        self.__m_full_name = ''
        self.__m_title = ''
        self.__m_first_name = ''
        self.__m_middle_name = ''
        self.__m_last_name = ''
        self.__m_suffix = ''
        self.__m_num_words = 0
        self.__m_words = [None]*5
    @property
    def full_name(self):
        return self.__m_full_name
    @full_name.setter
    def full_name(self, m_full_name):
        self.__m_full_name = m_full_name
    def _set_full_name(self, m_full_name):
        self.__m_full_name = m_full_name
    def title(self):
        return
    def first_name(self):
        return self.__m_first_name
    def middle_name(self):
        return self.__m_middle_name
    def last_name(self):
        return self.__m_last_name
    def suffix(self):
        return self.__m_suffix
    def _extract_words(self, value):
        words = re.findall(r'[^\s,.:\t]+', value)
        if len(words) <= 5:
            self.__m_num_words = len(words)
            self.__m_words[:len(words)] = words
        else:
            self.__m_num_words = 5
            self.__m_words = words[:5]
    def _parse_name(self):
        if self.__m_full_name is not None and self.__m_full_name != '':
            self._extract_words(self.__m_full_name)
            self._find_title()
            self._find_suffix()
            self._find_last_name()
            self._find_first_name()
            self._find_middle_name()
    def _find_title(self):
```

```
        title_list = ['Mr.', 'Mr', 'Ms.', 'Ms', 'Miss.', 'Miss', 'Dr.',
'Dr',
                      'Mrs.', 'Mrs','Fr.','Capt.','Lt.', 'Gen.',
'President',
                      'Sister', 'Father', 'Brother', 'Major']
        if self.__m_words is not None:
            if self.__m_words[0] in title_list:
                self.__m_title = self.__m_words[0]
                return 0
            return -1
        return -1
    def _find_suffix(self):
        suffix_list = ['DDS', 'CFA', 'CEO', 'CFO', 'Esq', 'CPA', 'MBA',
'PhD',
                       'MD', 'DC', 'Sr', 'Jr', 'II', 'III', 'IV']
        if self.__m_words[4] is not None:
            self.__m_suffix = self.__m_words[4]
            return 0
        else:
            if self.__m_words[2] is not None and self.__m_words[2] in \
                    suffix_list:
                self.__m_suffix = self.__m_words[2]
                return 0
            if self.__m_words[3] is not None and self.__m_words[3] in \
                    suffix_list:
                self.__m_suffix = self.__m_words[3]
                return 0
        return -1
    def _find_first_name(self):
        if self.__m_num_words >= 2 and self.__m_title == '':
            self.__m_first_name = self.__m_words[0]
            return 0
        if self.__m_num_words > 2 and self.__m_title != '':
            self.__m_first_name = self.__m_words[1]
            return 0
        if self.__m_num_words == 5:
            self.__m_first_name = self.__m_words[1]
            return 0
        return -1
    def _find_middle_name(self):
        if self.__m_num_words == 5 or self.__m_num_words == 4 and \
                self.__m_suffix == '':
            self.__m_middle_name = self.__m_words[2]
            return 0
        if (self.__m_num_words == 4 and self.__m_title == '') or (
                self.__m_num_words == 4 and self.__m_suffix == ''):
            self.__m_middle_name = self.__m_words[1]
            return 0
        if self.__m_num_words == 3 and self.__m_suffix == '' and \
                self.__m_title == '':
```

```
            self.__m_middle_name = self.__m_words[1]
        return -1
    def _find_last_name(self):
        if self.__m_num_words == 1:
            self.__m_last_name = self.__m_words[0]
            return 0
        if self.__m_num_words == 2:
            self.__m_last_name = self.__m_words[1]
            return 0
        if self.__m_num_words == 5:
            self.__m_last_name = self.__m_words[3]
            return 0
        if (self.__m_num_words == 3 and self.__m_suffix == '') or (
                self.__m_num_words == 4 and self.__m_suffix != ''):
            self.__m_last_name = self.__m_words[2]
            return 0
        else:
            if self.__m_num_words == 3:
                self.__m_last_name = self.__m_words[1]
            elif self.__m_num_words == 4:
                self.__m_last_name = self.__m_words[3]
                return 0
            return -1
    def test_print(self):
        print 'full name:', self.__m_full_name
        print 'title:', self.__m_title
        print 'first name:', self.__m_first_name
        print 'middle name:', self.__m_middle_name
        print 'last name:', self.__m_last_name
        print 'suffix:', self.__m_suffix
        print 'num words:', self.__m_num_words
        print 'words:', self.__m_words
        return 'complete\n'
class ENameExtractorError:
    def __init__(self):
        pass
    def e_name_extractor_error(self, message):
        pass
def main():
    name = NameExtractor()
    name.full_name = 'Mr. John Brown PhD'
    name._parse_name()
    print name.test_print()
    name1 = NameExtractor()
    name1.full_name = 'Mr.    John Brown'
    name1._parse_name()
    print name1.test_print()
    name2 = NameExtractor()
    name2.full_name = 'John Brown, PhD'
    name2._parse_name()
```

```
    print name2.test_print()
if __name__ == '__main__':
    main()
```