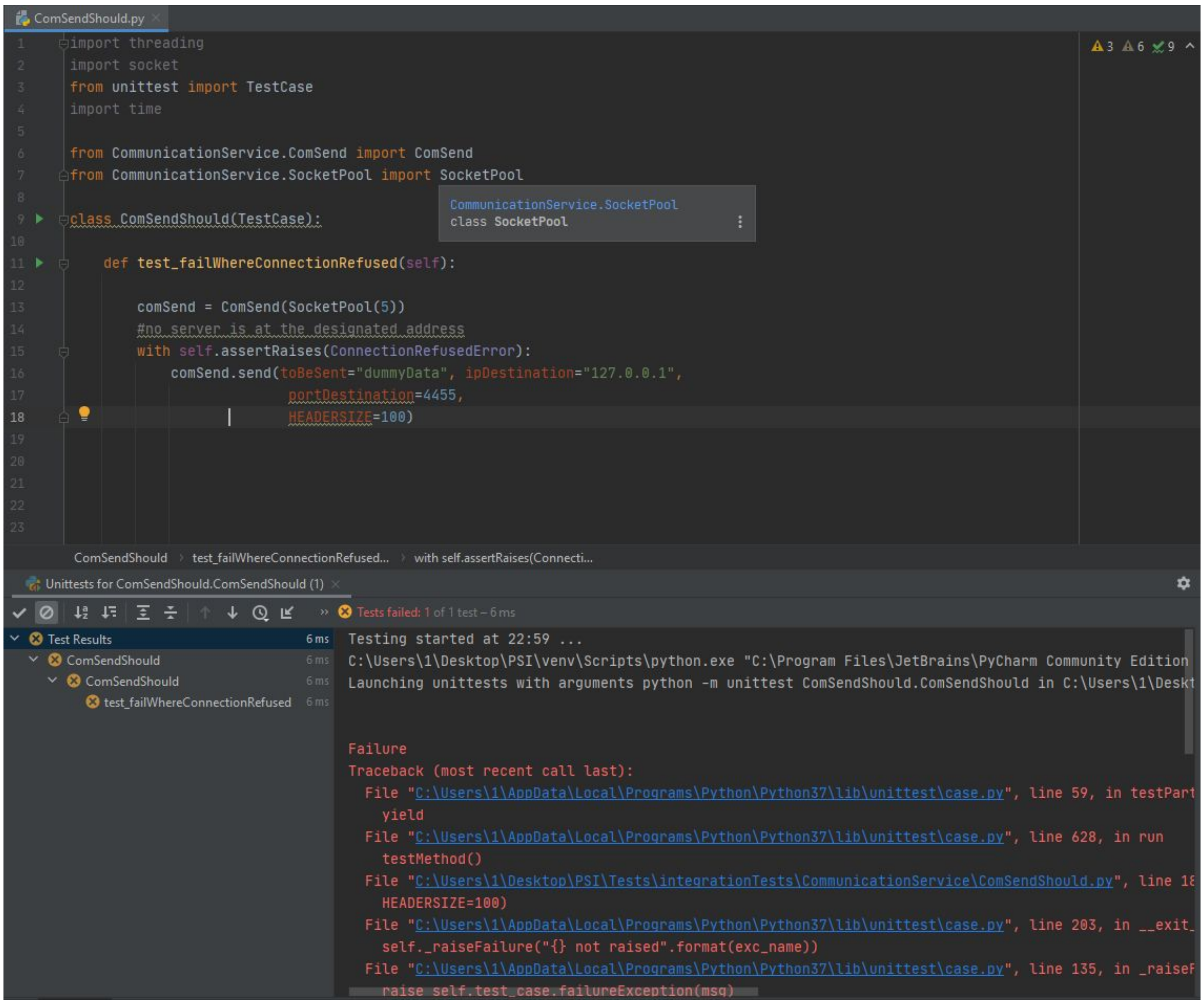
COMSEND CLASS

BEFORE IMPLEMENTING:



ComSendShould.py
CommunicationService.SocketPool
class SocketPool
ComSendShould › test_failWhereConnectionRefused... › with self.assertRaises(Connecti...
Unittests for ComSendShould.ComSendShould (1)
Tests failed: 1 of 1 test – 6 ms
Test Results 6 ms
ComSendShould 6 ms
ComSendShould 6 ms
test_failWhereConnectionRefused 6 ms

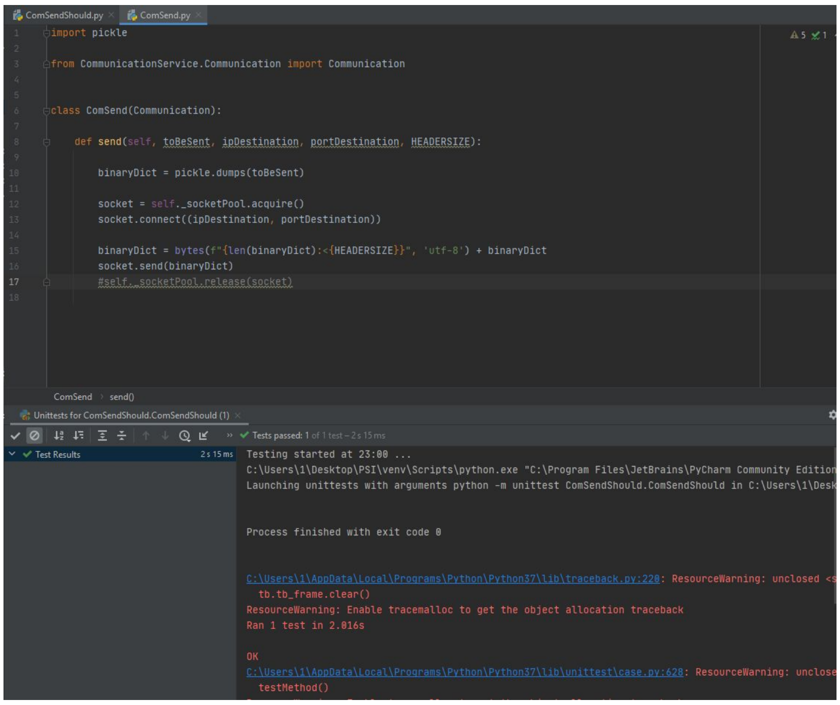
```
import threading
import socket
from unittest import TestCase
import time

from CommunicationService.ComSend import ComSend
from CommunicationService.SocketPool import SocketPool

class ComSendShould(TestCase):

    def test_failWhereConnectionRefused(self):

        comSend = ComSend(SocketPool(5))
        #no server is at the designated address
        with self.assertRaises(ConnectionRefusedError):
            comSend.send(toBeSent="dummyData", ipDestination="127.0.0.1",
                         portDestination=4455,
                         HEADERSIZE=100)
```

```
Testing started at 22:59 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition
Launching unittests with arguments python -m unittest ComSendShould.ComSendShould in C:\Users\1\Deskt


Failure
Traceback (most recent call last):
  File "C:\Users\1\AppData\Local\Programs\Python\Python37\lib\unittest\case.py", line 59, in testPart
    yield
  File "C:\Users\1\AppData\Local\Programs\Python\Python37\lib\unittest\case.py", line 628, in run
    testMethod()
  File "C:\Users\1\Desktop\PSI\Tests\integrationTests\CommunicationService\ComSendShould.py", line 18
    HEADERSIZE=100)
  File "C:\Users\1\AppData\Local\Programs\Python\Python37\lib\unittest\case.py", line 203, in __exit_
    self._raiseFailure("{} not raised".format(exc_name))
  File "C:\Users\1\AppData\Local\Programs\Python\Python37\lib\unittest\case.py", line 135, in _raiseF
    raise self.test_case.failureException(msg)
```

Explanation: In the first test, we had not caught the ConnectionRefused exception yet

The first test will fail because no ConnectionRefusedError is caught (no implemented component)

So, we'll write the code and run the tests again

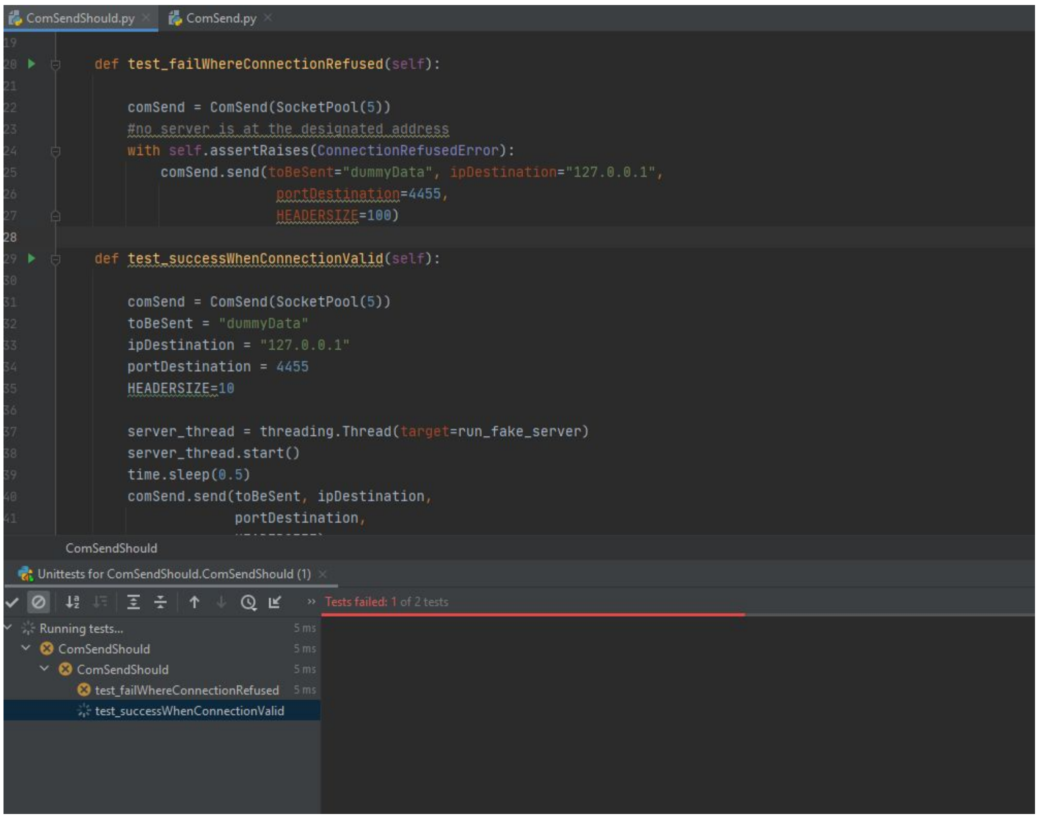


ComSendShould.py
ComSend.py
ComSend › send()
Unittests for ComSendShould.ComSendShould (1)
Tests passed: 1 of 1 test – 2 s 15 ms
Test Results 2 s 15 ms

```
import pickle

from CommunicationService.Communication import Communication


class ComSend(Communication):

    def send(self, toBeSent, ipDestination, portDestination, HEADERSIZE):

        binaryDict = pickle.dumps(toBeSent)

        socket = self._socketPool.acquire()
        socket.connect((ipDestination, portDestination))

        binaryDict = bytes(f"{len(binaryDict):<{HEADERSIZE}}", 'utf-8') + binaryDict
        socket.send(binaryDict)
        #self._socketPool.release(socket)
```

```
Testing started at 23:00 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition
Launching unittests with arguments python -m unittest ComSendShould.ComSendShould in C:\Users\1\Desk


Process finished with exit code 0


C:\Users\1\AppData\Local\Programs\Python\Python37\lib\traceback.py:220: ResourceWarning: unclosed <s
  tb.tb_frame.clear()
ResourceWarning: Enable tracemalloc to get the object allocation traceback
Ran 1 test in 2.016s

OK
C:\Users\1\AppData\Local\Programs\Python\Python37\lib\unittest\case.py:628: ResourceWarning: unclose
  testMethod()
```

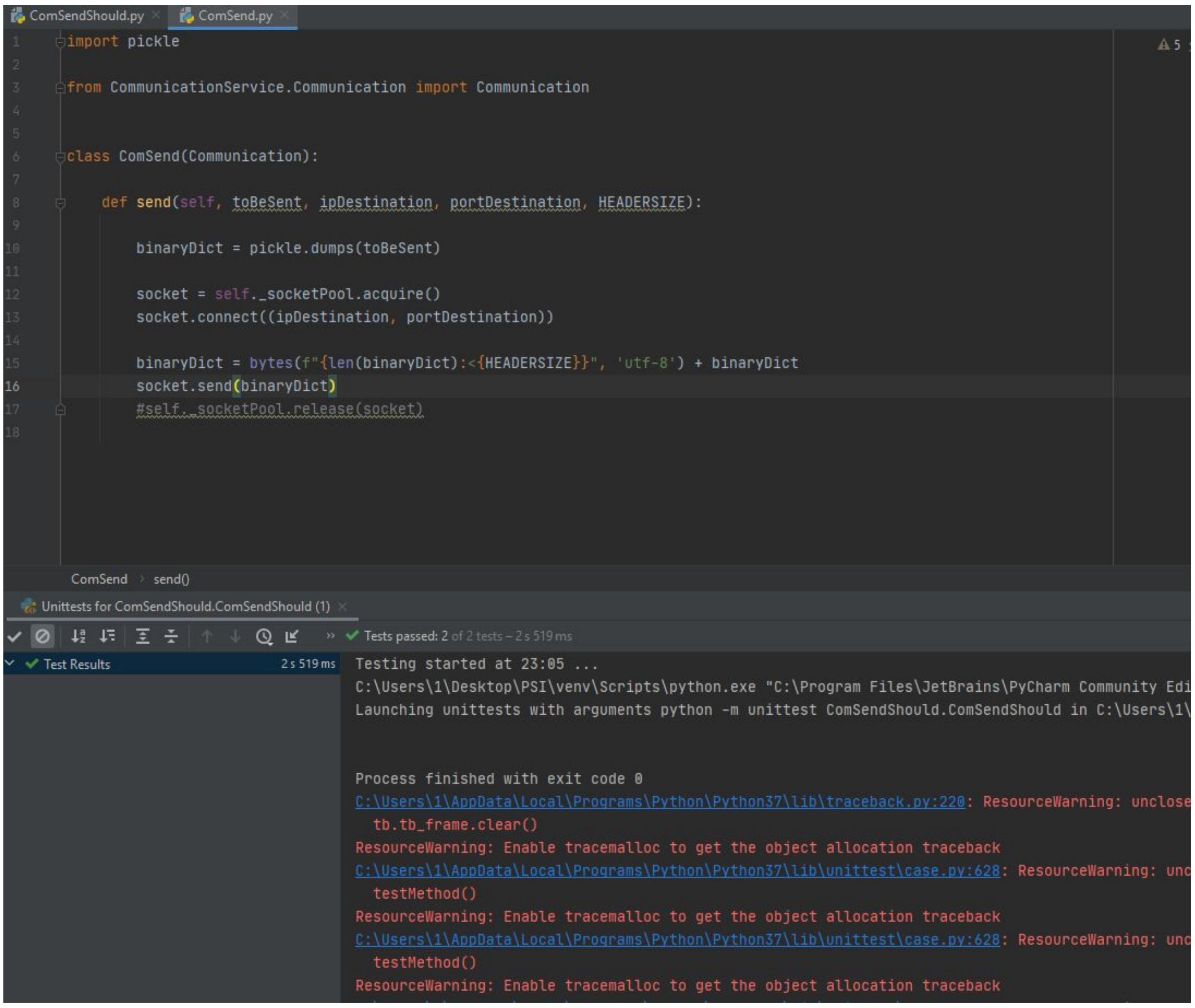
And the test will pass this time.
We add the second test, for success. It will run if we maintain this version of the ComSend class. So, we are going to comment on it for a while to see what’s happening.



```
    def test_failWhereConnectionRefused(self):

        comSend = ComSend(SocketPool(5))
        #no server is at the designated address
        with self.assertRaises(ConnectionRefusedError):
            comSend.send(toBeSent="dummyData", ipDestination="127.0.0.1",
                         portDestination=4455,
                         HEADERSIZE=100)

    def test_successWhenConnectionValid(self):

        comSend = ComSend(SocketPool(5))
        toBeSent = "dummyData"
        ipDestination = "127.0.0.1"
        portDestination = 4455
        HEADERSIZE=10

        server_thread = threading.Thread(target=run_fake_server)
        server_thread.start()
        time.sleep(0.5)
        comSend.send(toBeSent, ipDestination,
                     portDestination,
```

```
Tests failed: 1 of 2 tests
Running tests...                          5 ms
  ComSendShould                           5 ms
    ComSendShould                         5 ms
      test_failWhereConnectionRefused     5 ms
      test_successWhenConnectionValid
```

We will observe that the second test will run infinitely because the corresponding process will not join (no message is sent by the sender). We will undo the commenting code and obtain the result



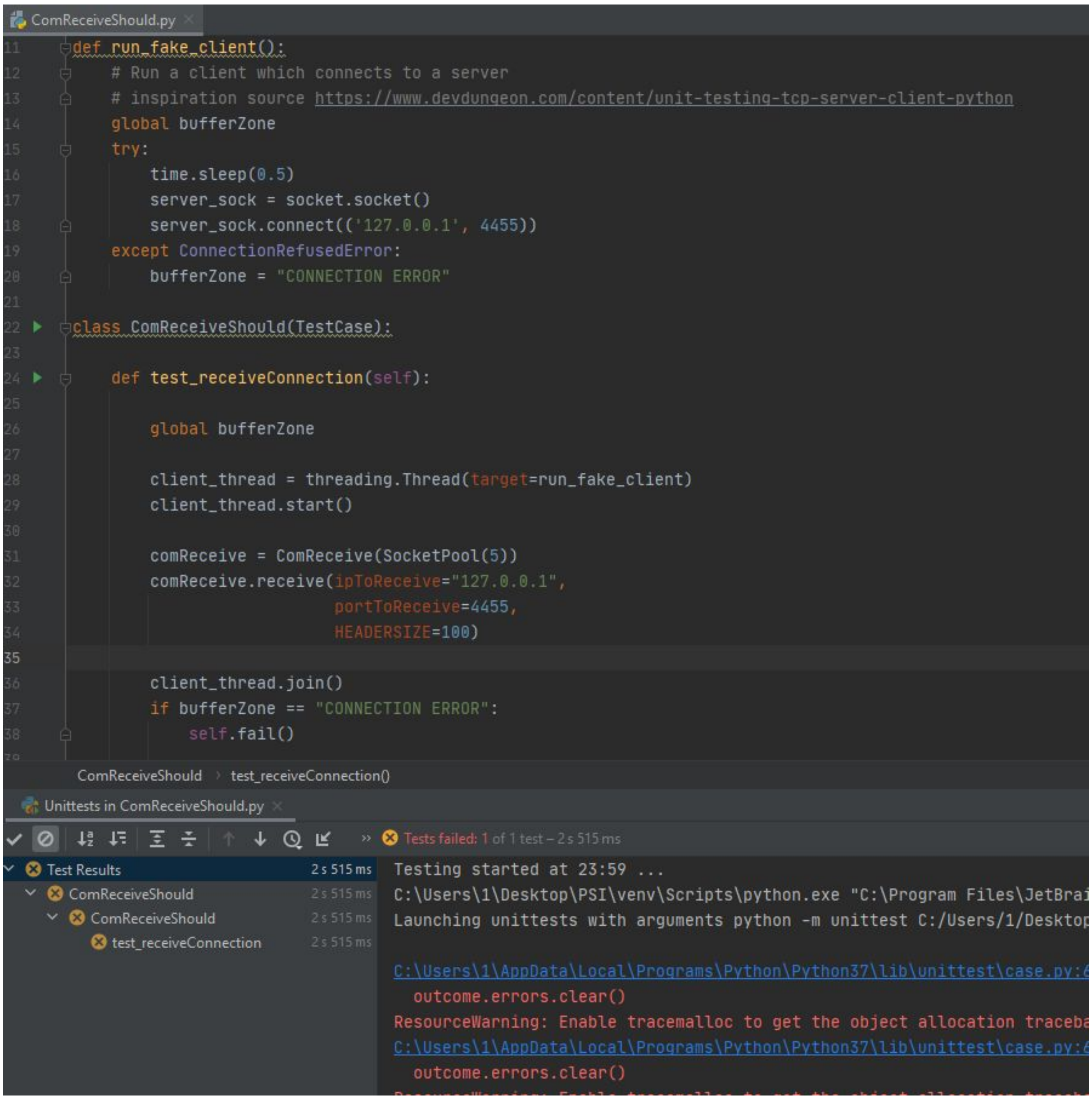
```
import pickle

from CommunicationService.Communication import Communication


class ComSend(Communication):

    def send(self, toBeSent, ipDestination, portDestination, HEADERSIZE):

        binaryDict = pickle.dumps(toBeSent)

        socket = self._socketPool.acquire()
        socket.connect((ipDestination, portDestination))

        binaryDict = bytes(f"{len(binaryDict):<{HEADERSIZE}}", 'utf-8') + binaryDict
        socket.send(binaryDict)
        #self._socketPool.release(socket)
```

```
Tests passed: 2 of 2 tests – 2 s 519 ms
Test Results    2 s 519 ms
```

```
Testing started at 23:05 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm Community Edi
Launching unittests with arguments python -m unittest ComSendShould.ComSendShould in C:\Users\1\


Process finished with exit code 0
C:\Users\1\AppData\Local\Programs\Python\Python37\lib\traceback.py:220: ResourceWarning: unclose
  tb.tb_frame.clear()
ResourceWarning: Enable tracemalloc to get the object allocation traceback
C:\Users\1\AppData\Local\Programs\Python\Python37\lib\unittest\case.py:628: ResourceWarning: unc
  testMethod()
ResourceWarning: Enable tracemalloc to get the object allocation traceback
C:\Users\1\AppData\Local\Programs\Python\Python37\lib\unittest\case.py:628: ResourceWarning: unc
  testMethod()
ResourceWarning: Enable tracemalloc to get the object allocation traceback
```

COMRECEIVE CLASS

Writing a test with no receiver implemented (dummy connection test)

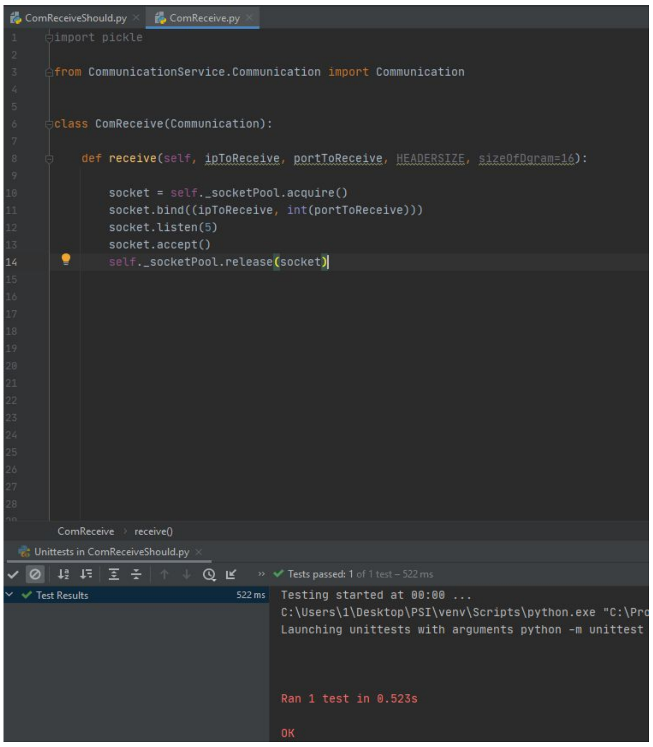


ComReceiveShould.py
ComReceiveShould › test_receiveConnection()
Unittests in ComReceiveShould.py

```python
def run_fake_client():
    # Run a client which connects to a server
    # inspiration source https://www.devdungeon.com/content/unit-testing-tcp-server-client-python
    global bufferZone
    try:
        time.sleep(0.5)
        server_sock = socket.socket()
        server_sock.connect(('127.0.0.1', 4455))
    except ConnectionRefusedError:
        bufferZone = "CONNECTION ERROR"

class ComReceiveShould(TestCase):

    def test_receiveConnection(self):

        global bufferZone

        client_thread = threading.Thread(target=run_fake_client)
        client_thread.start()

        comReceive = ComReceive(SocketPool(5))
        comReceive.receive(ipToReceive="127.0.0.1",
                           portToReceive=4455,
                           HEADERSIZE=100)

        client_thread.join()
        if bufferZone == "CONNECTION ERROR":
            self.fail()
```

```
Tests failed: 1 of 1 test – 2 s 515 ms

Test Results                        2 s 515 ms
  ComReceiveShould                  2 s 515 ms
    ComReceiveShould                2 s 515 ms
      test_receiveConnection        2 s 515 ms

Testing started at 23:59 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\JetBrai
Launching unittests with arguments python -m unittest C:/Users/1/Desktop

C:\Users\1\AppData\Local\Programs\Python\Python37\lib\unittest\case.py:6
  outcome.errors.clear()
ResourceWarning: Enable tracemalloc to get the object allocation traceba
C:\Users\1\AppData\Local\Programs\Python\Python37\lib\unittest\case.py:6
  outcome.errors.clear()
```

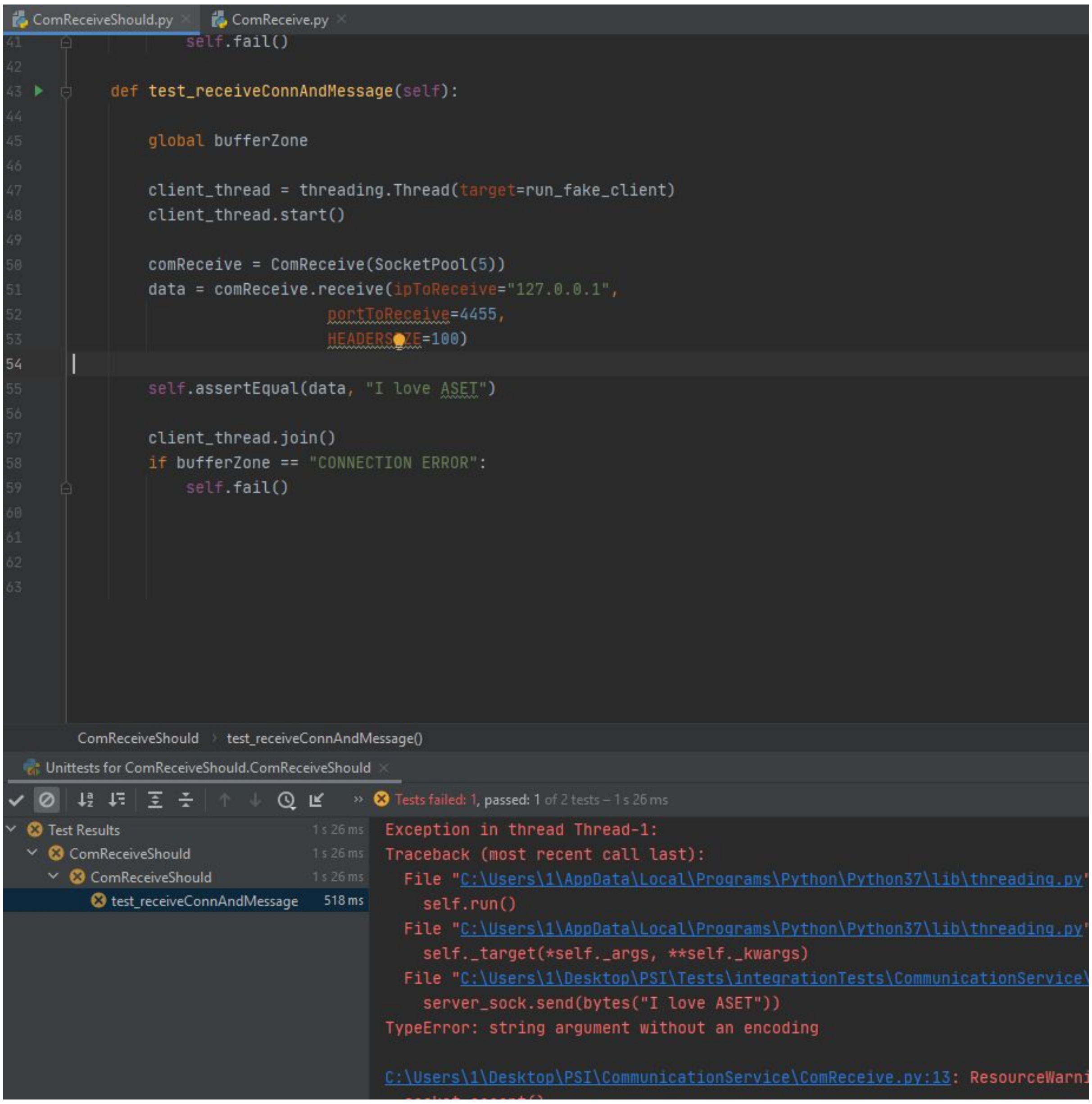
And implementing it in a first stage



```python
import pickle

from CommunicationService.Communication import Communication


class ComReceive(Communication):

    def receive(self, ipToReceive, portToReceive, HEADERSIZE, sizeOfDgram=16):

        socket = self._socketPool.acquire()
        socket.bind((ipToReceive, int(portToReceive)))
        socket.listen(5)
        socket.accept()
        self._socketPool.release(socket)
```

```
Tests passed: 1 of 1 test – 522 ms

Test Results    522 ms

Testing started at 00:00 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Pro
Launching unittests with arguments python -m unittest


Ran 1 test in 0.523s

OK
```

Now, we would like to add the feature to receive messages (as in our real app, not just to connect)



ComReceiveShould.py | ComReceive.py

```python
            self.fail()

    def test_receiveConnAndMessage(self):

        global bufferZone

        client_thread = threading.Thread(target=run_fake_client)
        client_thread.start()

        comReceive = ComReceive(SocketPool(5))
        data = comReceive.receive(ipToReceive="127.0.0.1",
                                  portToReceive=4455,
                                  HEADERSIZE=100)

        self.assertEqual(data, "I love ASET")

        client_thread.join()
        if bufferZone == "CONNECTION ERROR":
            self.fail()
```

ComReceiveShould › test_receiveConnAndMessage()

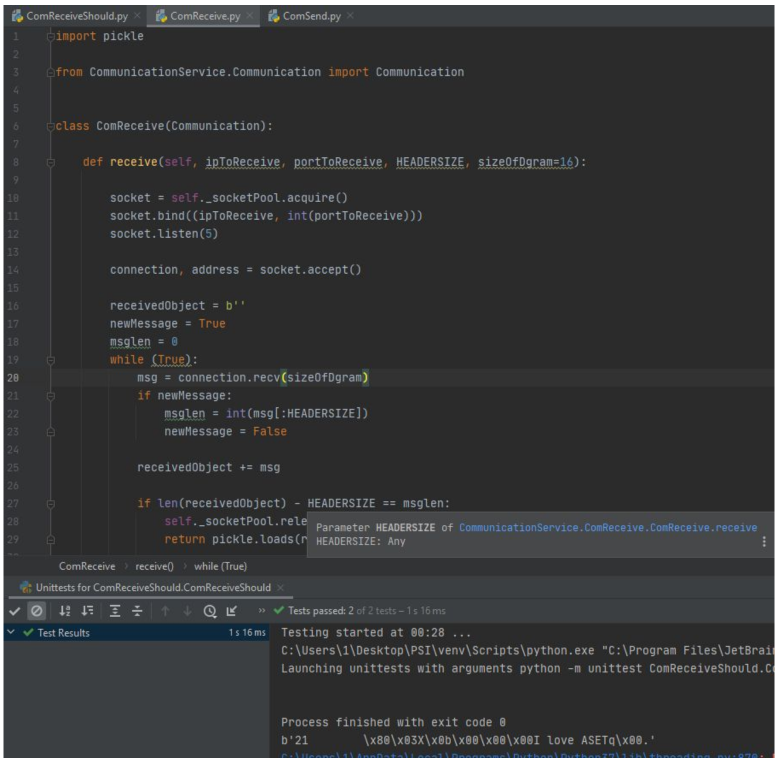
Unittests for ComReceiveShould.ComReceiveShould

Tests failed: 1, passed: 1 of 2 tests – 1 s 26 ms

```
Test Results                      1 s 26 ms
  ComReceiveShould                1 s 26 ms
    ComReceiveShould              1 s 26 ms
      test_receiveConnAndMessage  518 ms
```

```
Exception in thread Thread-1:
Traceback (most recent call last):
  File "C:\Users\1\AppData\Local\Programs\Python\Python37\lib\threading.py"
    self.run()
  File "C:\Users\1\AppData\Local\Programs\Python\Python37\lib\threading.py"
    self._target(*self._args, **self._kwargs)
  File "C:\Users\1\Desktop\PSI\Tests\integrationTests\CommunicationService\
    server_sock.send(bytes("I love ASET"))
TypeError: string argument without an encoding

C:\Users\1\Desktop\PSI\CommunicationService\ComReceive.py:13: ResourceWarni
```

Of course, we had not implemented the receiving message extension, so let's do it, arrange and test:



ComReceiveShould.py | ComReceive.py | ComSend.py

```python
import pickle

from CommunicationService.Communication import Communication


class ComReceive(Communication):

    def receive(self, ipToReceive, portToReceive, HEADERSIZE, sizeOfDgram=16):

        socket = self._socketPool.acquire()
        socket.bind((ipToReceive, int(portToReceive)))
        socket.listen(5)

        connection, address = socket.accept()

        receivedObject = b''
        newMessage = True
        msglen = 0
        while (True):
            msg = connection.recv(sizeOfDgram)
            if newMessage:
                msglen = int(msg[:HEADERSIZE])
                newMessage = False

            receivedObject += msg

            if len(receivedObject) - HEADERSIZE == msglen:
                self._socketPool.rele
                return pickle.loads(r
```

```
Parameter HEADERSIZE of CommunicationService.ComReceive.ComReceive.receive
HEADERSIZE: Any
```

ComReceive › receive() › while (True)

Unittests for ComReceiveShould.ComReceiveShould

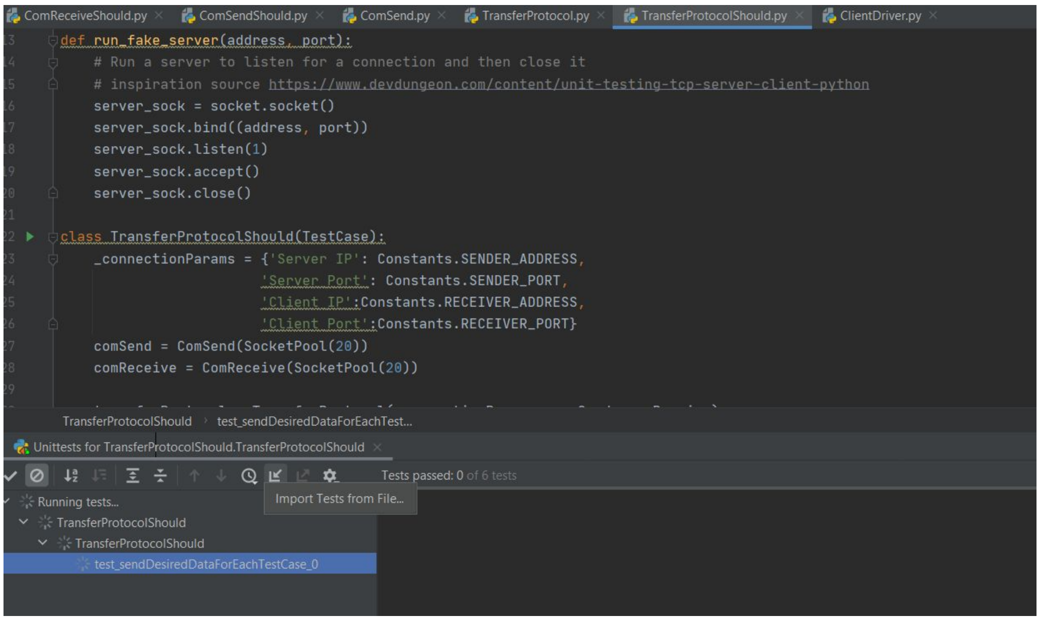
Tests passed: 2 of 2 tests – 1 s 16 ms

```
Test Results  1 s 16 ms
```

```
Testing started at 00:28 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\JetBrai
Launching unittests with arguments python -m unittest ComReceiveShould.C


Process finished with exit code 0
b'21        \x80\x03X\x0b\x00\x00\x00I love ASETq\x00.'
```

TRANSFER PROTOCOL

For each case, we are writing a parameterized test for sender methods. Without code, these tests will freeze because the dummy server is waiting continuously, as seen here

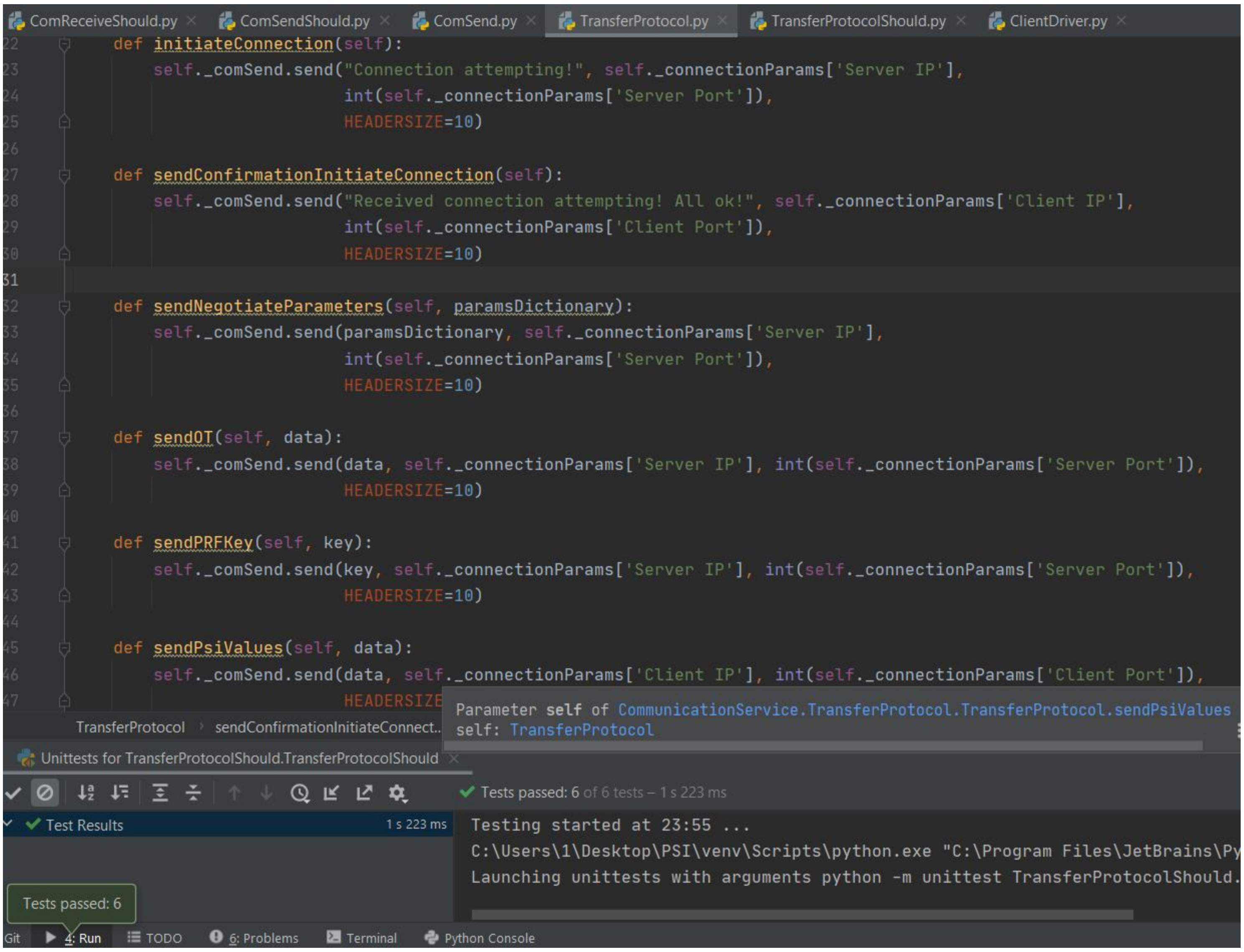


```python
def run_fake_server(address, port):
    # Run a server to listen for a connection and then close it
    # inspiration source https://www.devdungeon.com/content/unit-testing-tcp-server-client-python
    server_sock = socket.socket()
    server_sock.bind((address, port))
    server_sock.listen(1)
    server_sock.accept()
    server_sock.close()

class TransferProtocolShould(TestCase):
    _connectionParams = {'Server IP': Constants.SENDER_ADDRESS,
                         'Server Port': Constants.SENDER_PORT,
                         'Client IP':Constants.RECEIVER_ADDRESS,
                         'Client Port':Constants.RECEIVER_PORT}
    comSend = ComSend(SocketPool(20))
    comReceive = ComReceive(SocketPool(20))
```

TransferProtocolShould › test_sendDesiredDataForEachTest...
Unittests for TransferProtocolShould.TransferProtocolShould
Import Tests from File...

```
Tests passed: 0 of 6 tests
Running tests...
  TransferProtocolShould
    TransferProtocolShould
      test_sendDesiredDataForEachTestCase_0
```

Adding code, we will obtain



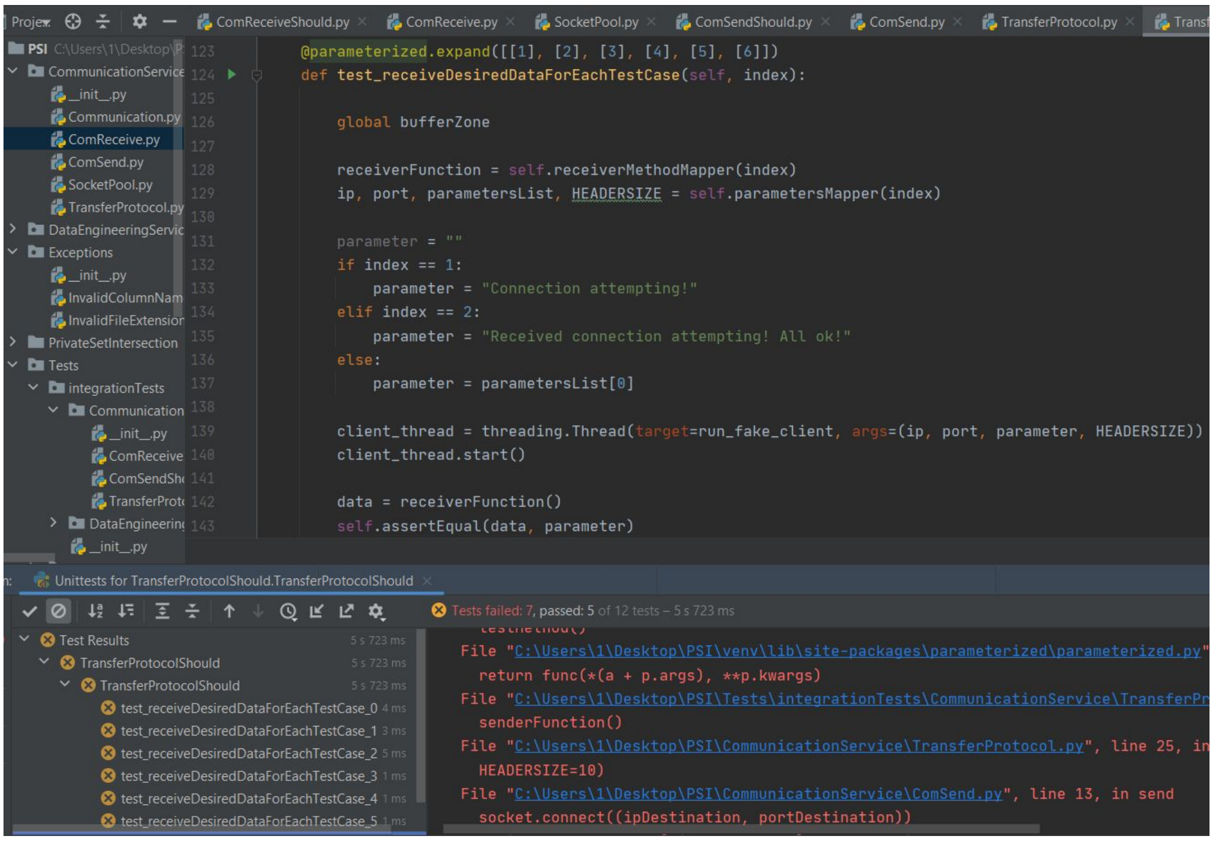
```python
    def initiateConnection(self):
        self._comSend.send("Connection attempting!", self._connectionParams['Server IP'],
                           int(self._connectionParams['Server Port']),
                           HEADERSIZE=10)

    def sendConfirmationInitiateConnection(self):
        self._comSend.send("Received connection attempting! All ok!", self._connectionParams['Client IP'],
                           int(self._connectionParams['Client Port']),
                           HEADERSIZE=10)

    def sendNegotiateParameters(self, paramsDictionary):
        self._comSend.send(paramsDictionary, self._connectionParams['Server IP'],
                           int(self._connectionParams['Server Port']),
                           HEADERSIZE=10)

    def sendOT(self, data):
        self._comSend.send(data, self._connectionParams['Server IP'], int(self._connectionParams['Server Port']),
                           HEADERSIZE=10)

    def sendPRFKey(self, key):
        self._comSend.send(key, self._connectionParams['Server IP'], int(self._connectionParams['Server Port']),
                           HEADERSIZE=10)

    def sendPsiValues(self, data):
        self._comSend.send(data, self._connectionParams['Client IP'], int(self._connectionParams['Client Port']),
                           HEADERSIZE
```

```
Parameter self of CommunicationService.TransferProtocol.TransferProtocol.sendPsiValues
self: TransferProtocol
```

TransferProtocol › sendConfirmationInitiateConnect...
Unittests for TransferProtocolShould.TransferProtocolShould
Git
4: Run
TODO
6: Problems
Terminal
Python Console

```
Tests passed: 6 of 6 tests – 1 s 223 ms
Test Results    1 s 223 ms
Tests passed: 6

Testing started at 23:55 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\JetBrains\Py
Launching unittests with arguments python -m unittest TransferProtocolShould.
```

Same for receiver methods



```python
    @parameterized.expand([[1], [2], [3], [4], [5], [6]])
    def test_receiveDesiredDataForEachTestCase(self, index):

        global bufferZone

        receiverFunction = self.receiverMethodMapper(index)
        ip, port, parametersList, HEADERSIZE = self.parametersMapper(index)

        parameter = ""
        if index == 1:
            parameter = "Connection attempting!"
        elif index == 2:
            parameter = "Received connection attempting! All ok!"
        else:
            parameter = parametersList[0]

        client_thread = threading.Thread(target=run_fake_client, args=(ip, port, parameter, HEADERSIZE))
        client_thread.start()

        data = receiverFunction()
        self.assertEqual(data, parameter)
```

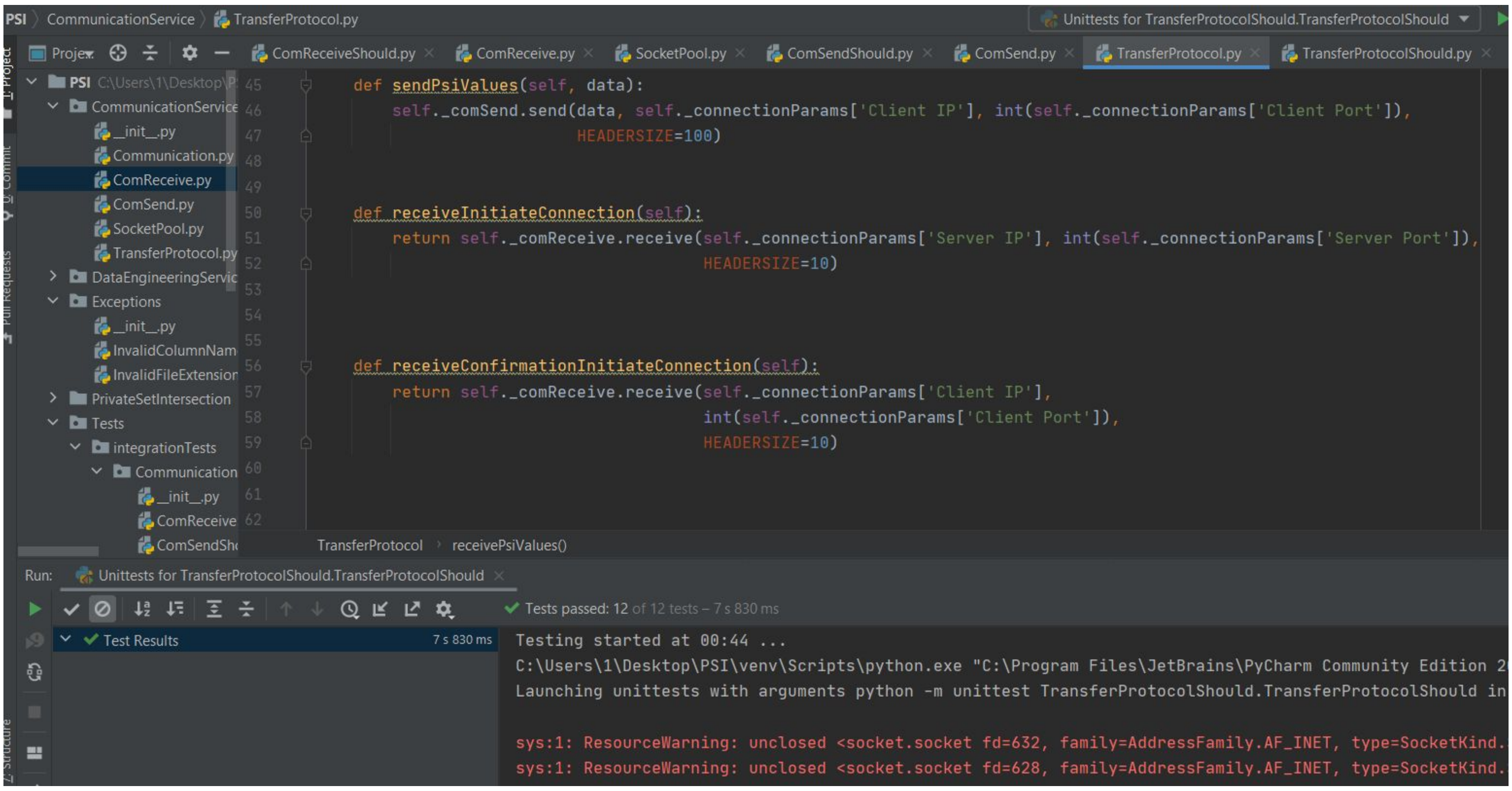
Unittests for TransferProtocolShould.TransferProtocolShould

Tests failed: 7, passed: 5 of 12 tests – 5 s 723 ms

```
Test Results                                      5 s 723 ms
  TransferProtocolShould                          5 s 723 ms
    TransferProtocolShould                        5 s 723 ms
      test_receiveDesiredDataForEachTestCase_0    4 ms
      test_receiveDesiredDataForEachTestCase_1    3 ms
      test_receiveDesiredDataForEachTestCase_2    5 ms
      test_receiveDesiredDataForEachTestCase_3    1 ms
      test_receiveDesiredDataForEachTestCase_4    1 ms
      test_receiveDesiredDataForEachTestCase_5    1 ms
```

```
File "C:\Users\1\Desktop\PSI\venv\lib\site-packages\parameterized\parameterized.py"
  return func(*(a + p.args), **p.kwargs)
File "C:\Users\1\Desktop\PSI\Tests\integrationTests\CommunicationService\TransferPr
  senderFunction()
File "C:\Users\1\Desktop\PSI\CommunicationService\TransferProtocol.py", line 25, in
  HEADERSIZE=10)
File "C:\Users\1\Desktop\PSI\CommunicationService\ComSend.py", line 13, in send
  socket.connect((ipDestination, portDestination))
```

writing code, after that:



```python
    def sendPsiValues(self, data):
        self._comSend.send(data, self._connectionParams['Client IP'], int(self._connectionParams['Client Port']),
                           HEADERSIZE=100)


    def receiveInitiateConnection(self):
        return self._comReceive.receive(self._connectionParams['Server IP'], int(self._connectionParams['Server Port']),
                                        HEADERSIZE=10)


    def receiveConfirmationInitiateConnection(self):
        return self._comReceive.receive(self._connectionParams['Client IP'],
                                        int(self._connectionParams['Client Port']),
                                        HEADERSIZE=10)
```

Run: Unittests for TransferProtocolShould.TransferProtocolShould

Tests passed: 12 of 12 tests – 7 s 830 ms

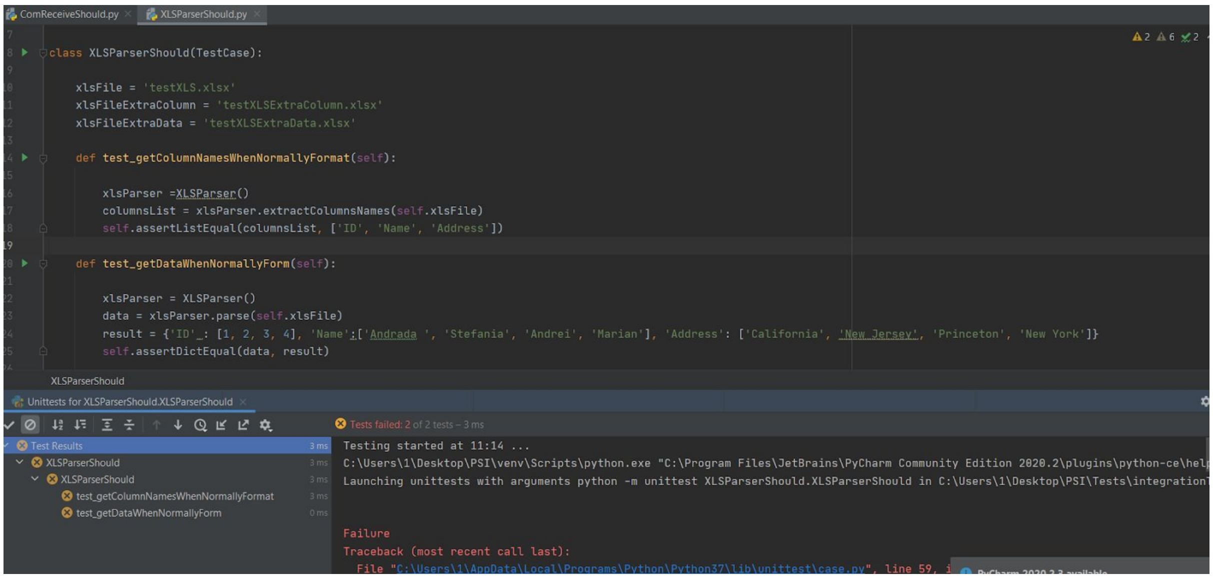
```
Test Results    7 s 830 ms
```

```
Testing started at 00:44 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2
Launching unittests with arguments python -m unittest TransferProtocolShould.TransferProtocolShould in

sys:1: ResourceWarning: unclosed <socket.socket fd=632, family=AddressFamily.AF_INET, type=SocketKind.
sys:1: ResourceWarning: unclosed <socket.socket fd=628, family=AddressFamily.AF_INET, type=SocketKind.
```

DATA ENGINEERING COMPONENT

XLSPARSER

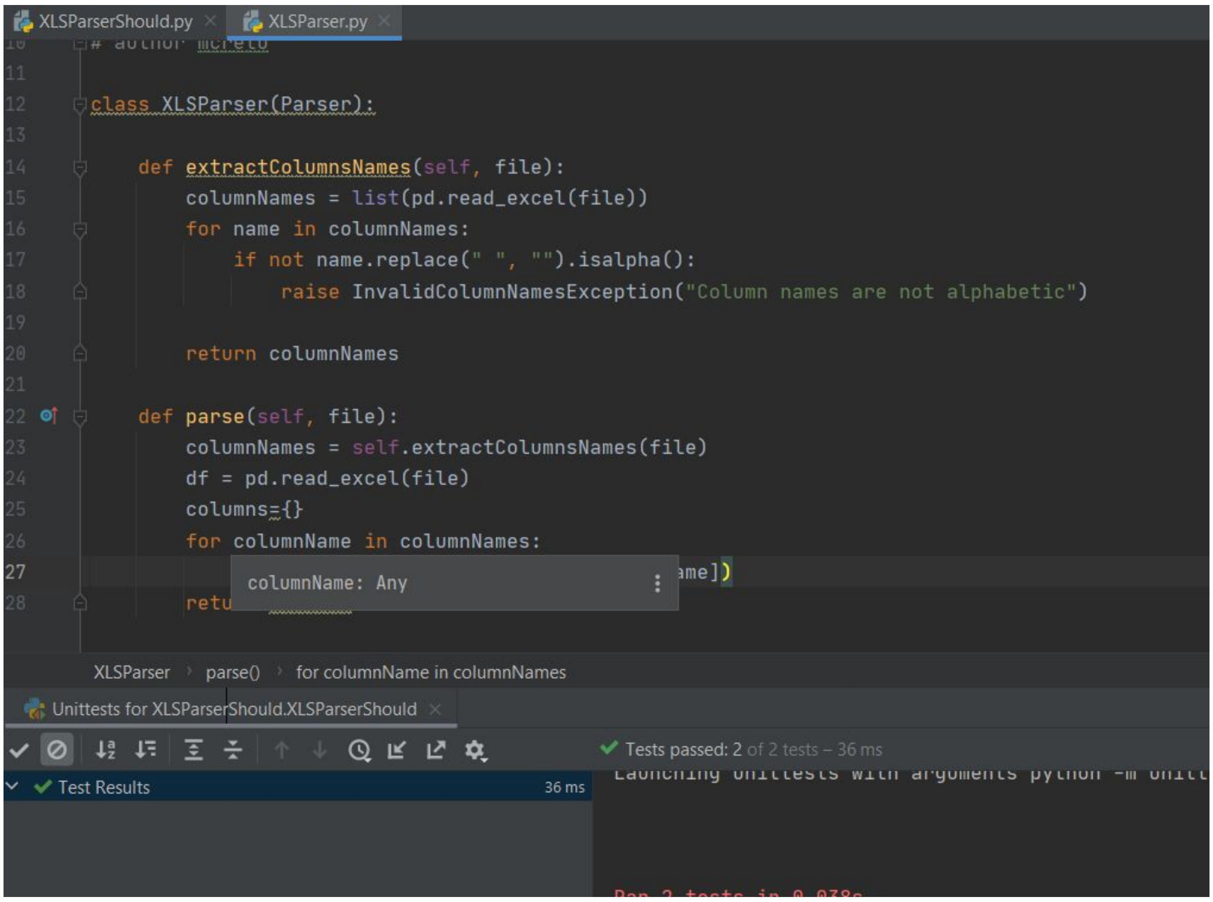
Write the first two tests



```
class XLSParserShould(TestCase):

    xlsFile = 'testXLS.xlsx'
    xlsFileExtraColumn = 'testXLSExtraColumn.xlsx'
    xlsFileExtraData = 'testXLSExtraData.xlsx'

    def test_getColumnNamesWhenNormallyFormat(self):

        xlsParser =XLSParser()
        columnsList = xlsParser.extractColumnsNames(self.xlsFile)
        self.assertListEqual(columnsList, ['ID', 'Name', 'Address'])

    def test_getDataWhenNormallyForm(self):

        xlsParser = XLSParser()
        data = xlsParser.parse(self.xlsFile)
        result = {'ID' : [1, 2, 3, 4], 'Name':['Andrada ', 'Stefania', 'Andrei', 'Marian'], 'Address': ['California', 'New Jersey', 'Princeton', 'New York']}
        self.assertDictEqual(data, result)
```

```
Tests failed: 2 of 2 tests – 3 ms

Test Results                                   3 ms
  XLSParserShould                              3 ms
    XLSParserShould                            3 ms
      test_getColumnNamesWhenNormallyFormat    3 ms
      test_getDataWhenNormallyForm             0 ms

Testing started at 11:14 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2020.2\plugins\python-ce\hel
Launching unittests with arguments python -m unittest XLSParserShould.XLSParserShould in C:\Users\1\Desktop\PSI\Tests\integration


Failure
Traceback (most recent call last):
  File "C:\Users\1\AppData\Local\Programs\Python\Python37\lib\unittest\case.py", line 59, i
```

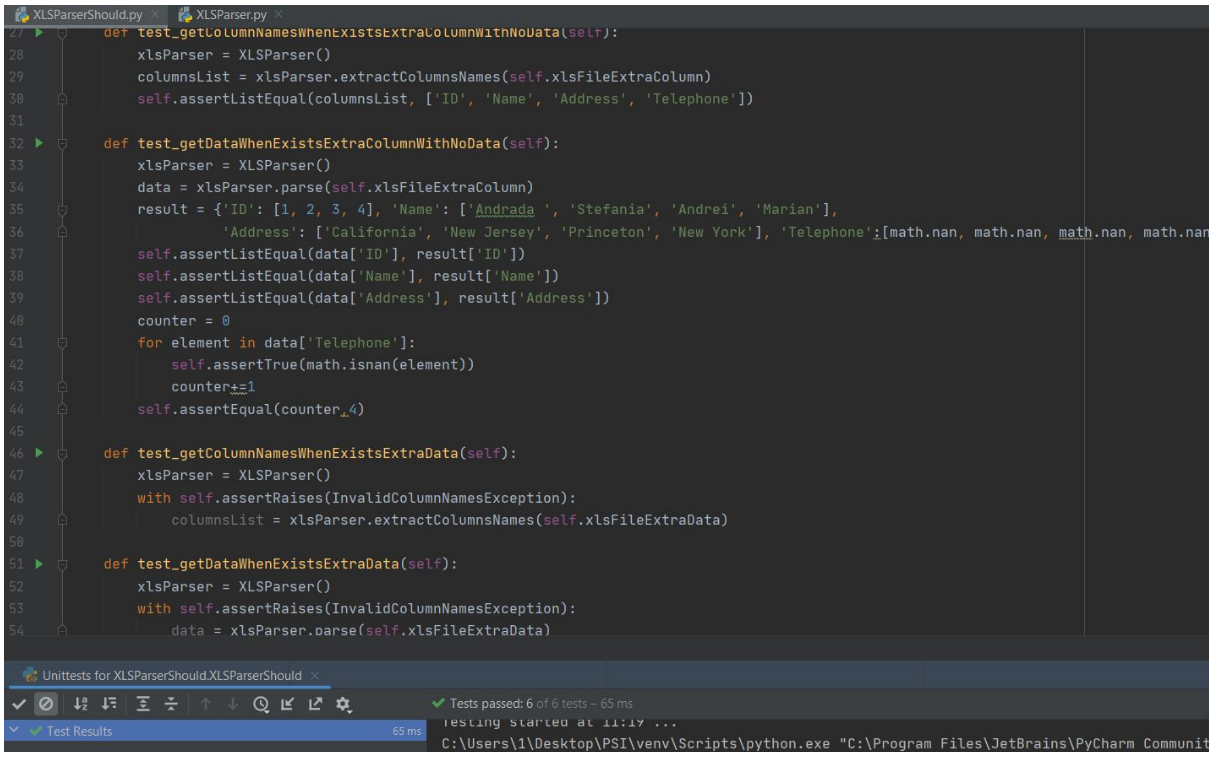
And implementing the methods



```
# author mcretu

class XLSParser(Parser):

    def extractColumnsNames(self, file):
        columnNames = list(pd.read_excel(file))
        for name in columnNames:
            if not name.replace(" ", "").isalpha():
                raise InvalidColumnNamesException("Column names are not alphabetic")

        return columnNames

    def parse(self, file):
        columnNames = self.extractColumnsNames(file)
        df = pd.read_excel(file)
        columns={}
        for columnName in columnNames:
                                          ame])
        retu
```

```
Tests passed: 2 of 2 tests – 36 ms

Test Results    36 ms

Launching unittests with arguments python -m unitt


Ran 2 tests in 0.038s
```

The second stage will be in covering some test edge cases: for an extra column with no data (where data of form nan should appear), and extra data without column designated (which throw an InvalidColumnNamesException). These are also covered case, so if we write the corresponding tests, the tests will pass with minimum refactoring, as follows:

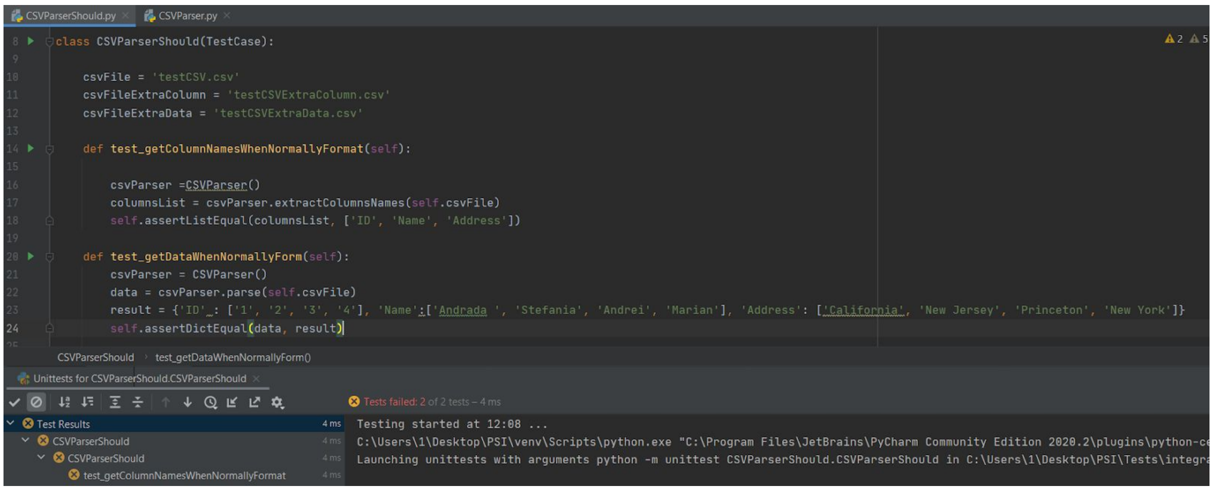


```
    def test_getColumnNamesWhenExistsExtraColumnWithNoData(self):
        xlsParser = XLSParser()
        columnsList = xlsParser.extractColumnsNames(self.xlsFileExtraColumn)
        self.assertListEqual(columnsList, ['ID', 'Name', 'Address', 'Telephone'])

    def test_getDataWhenExistsExtraColumnWithNoData(self):
        xlsParser = XLSParser()
        data = xlsParser.parse(self.xlsFileExtraColumn)
        result = {'ID': [1, 2, 3, 4], 'Name': ['Andrada ', 'Stefania', 'Andrei', 'Marian'],
                  'Address': ['California', 'New Jersey', 'Princeton', 'New York'], 'Telephone':[math.nan, math.nan, math.nan, math.nan
        self.assertListEqual(data['ID'], result['ID'])
        self.assertListEqual(data['Name'], result['Name'])
        self.assertListEqual(data['Address'], result['Address'])
        counter = 0
        for element in data['Telephone']:
            self.assertTrue(math.isnan(element))
            counter+=1
        self.assertEqual(counter,4)

    def test_getColumnNamesWhenExistsExtraData(self):
        xlsParser = XLSParser()
        with self.assertRaises(InvalidColumnNamesException):
            columnsList = xlsParser.extractColumnsNames(self.xlsFileExtraData)

    def test_getDataWhenExistsExtraData(self):
        xlsParser = XLSParser()
        with self.assertRaises(InvalidColumnNamesException):
            data = xlsParser.parse(self.xlsFileExtraData)
```

```
Tests passed: 6 of 6 tests – 65 ms
Testing started at 11:19 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm Communit
```

CSVPARSER

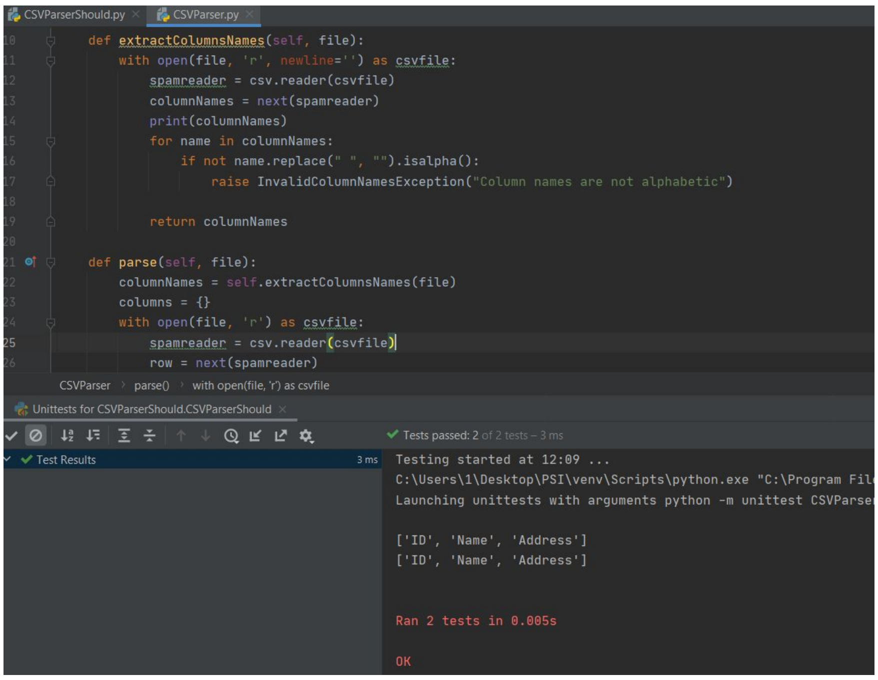
We are going to do the same here:



```
class CSVParserShould(TestCase):

    csvFile = 'testCSV.csv'
    csvFileExtraColumn = 'testCSVExtraColumn.csv'
    csvFileExtraData = 'testCSVExtraData.csv'

    def test_getColumnNamesWhenNormallyFormat(self):

        csvParser =CSVParser()
        columnsList = csvParser.extractColumnsNames(self.csvFile)
        self.assertListEqual(columnsList, ['ID', 'Name', 'Address'])

    def test_getDataWhenNormallyForm(self):
        csvParser = CSVParser()
        data = csvParser.parse(self.csvFile)
        result = {'ID' : ['1', '2', '3', '4'], 'Name':['Andrada ', 'Stefania', 'Andrei', 'Marian'], 'Address': ['California', 'New Jersey', 'Princeton', 'New York']}
        self.assertDictEqual(data, result)
```

```
Tests failed: 2 of 2 tests – 4 ms
Testing started at 12:08 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2020.2\plugins\python-ce
Launching unittests with arguments python -m unittest CSVParserShould.CSVParserShould in C:\Users\1\Desktop\PSI\Tests\integra
```

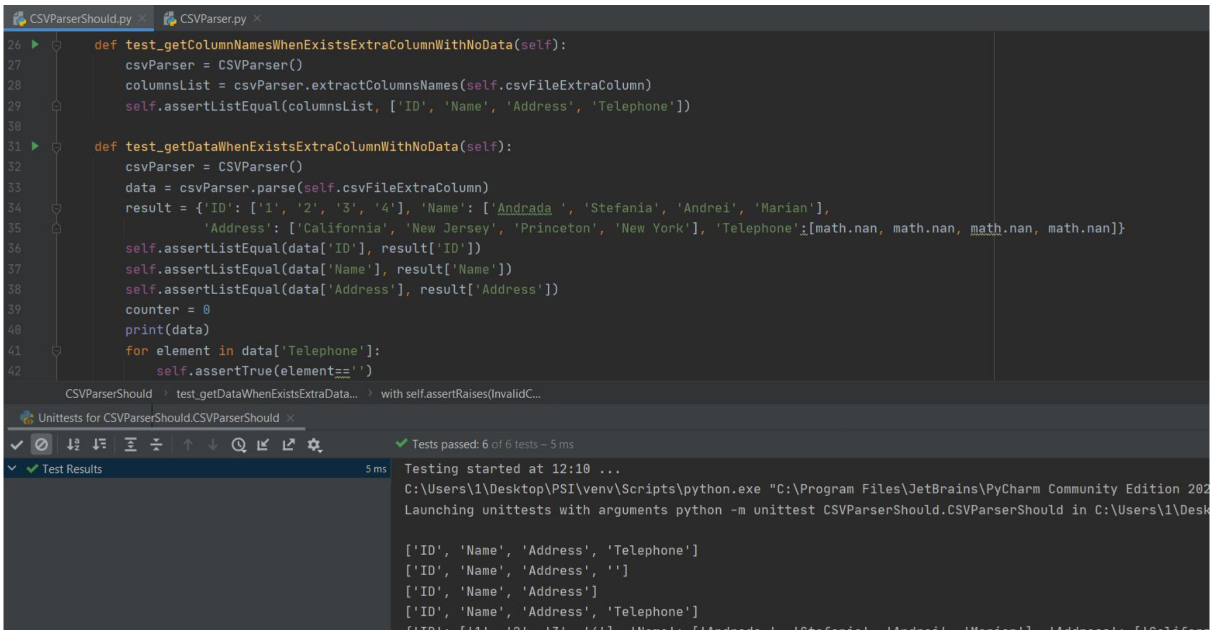
After completing the code:



```
    def extractColumnsNames(self, file):
        with open(file, 'r', newline='') as csvfile:
            spamreader = csv.reader(csvfile)
            columnNames = next(spamreader)
            print(columnNames)
            for name in columnNames:
                if not name.replace(" ", "").isalpha():
                    raise InvalidColumnNamesException("Column names are not alphabetic")

            return columnNames

    def parse(self, file):
        columnNames = self.extractColumnsNames(file)
        columns = {}
        with open(file, 'r') as csvfile:
            spamreader = csv.reader(csvfile)
            row = next(spamreader)
```

```
Tests passed: 2 of 2 tests – 3 ms
Testing started at 12:09 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Fil
Launching unittests with arguments python -m unittest CSVParser

['ID', 'Name', 'Address']
['ID', 'Name', 'Address']


Ran 2 tests in 0.005s

OK
```

Treating the same situations as above, we write tests and run code again

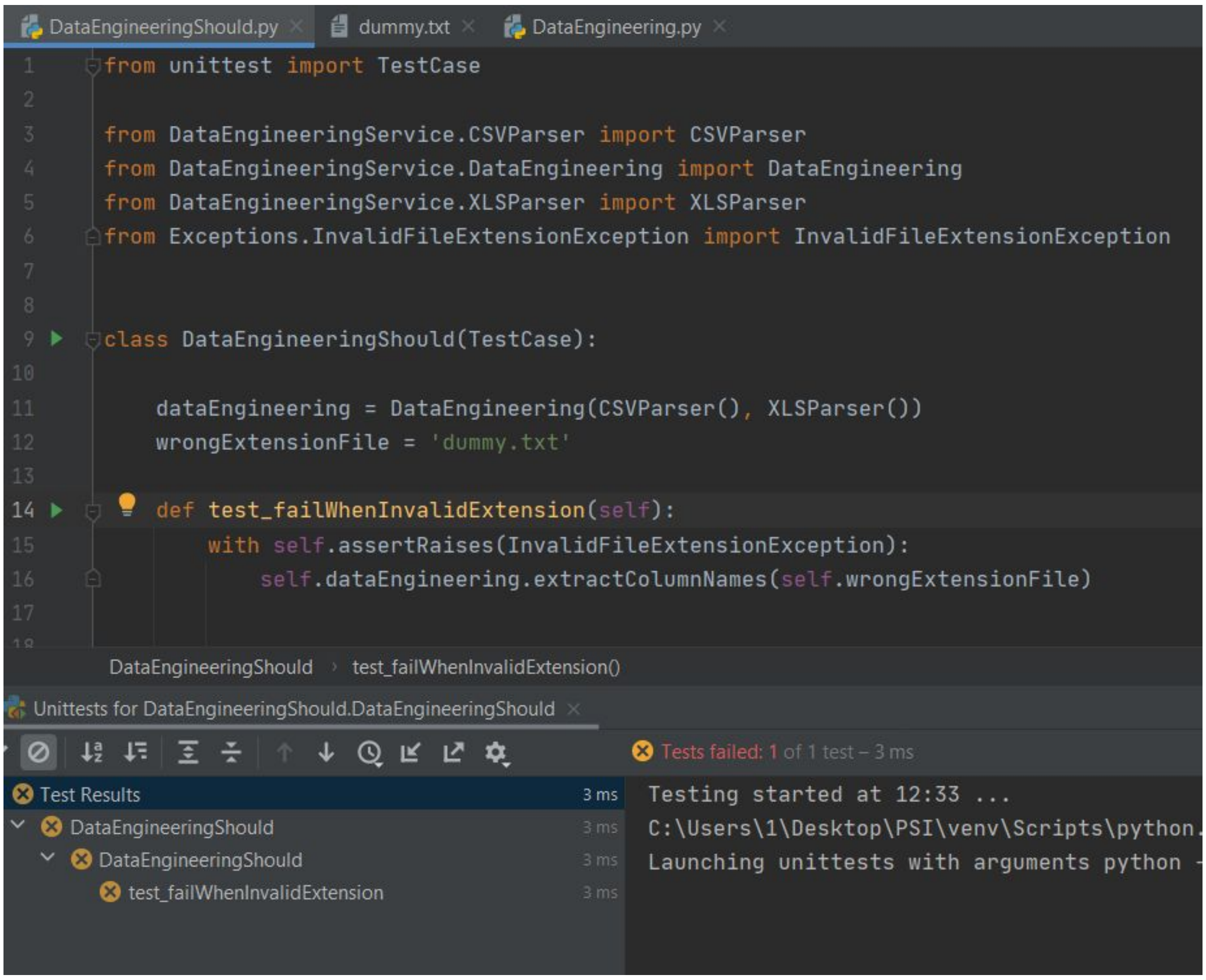


```python
    def test_getColumnNamesWhenExistsExtraColumnWithNoData(self):
        csvParser = CSVParser()
        columnsList = csvParser.extractColumnsNames(self.csvFileExtraColumn)
        self.assertListEqual(columnsList, ['ID', 'Name', 'Address', 'Telephone'])

    def test_getDataWhenExistsExtraColumnWithNoData(self):
        csvParser = CSVParser()
        data = csvParser.parse(self.csvFileExtraColumn)
        result = {'ID': ['1', '2', '3', '4'], 'Name': ['Andrada ', 'Stefania', 'Andrei', 'Marian'],
                  'Address': ['California', 'New Jersey', 'Princeton', 'New York'], 'Telephone':[math.nan, math.nan, math.nan, math.nan]}
        self.assertListEqual(data['ID'], result['ID'])
        self.assertListEqual(data['Name'], result['Name'])
        self.assertListEqual(data['Address'], result['Address'])
        counter = 0
        print(data)
        for element in data['Telephone']:
            self.assertTrue(element=='')
```

```
Tests passed: 6 of 6 tests – 5 ms
Test Results 5 ms

Testing started at 12:10 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 202
Launching unittests with arguments python -m unittest CSVParserShould.CSVParserShould in C:\Users\1\Desk

['ID', 'Name', 'Address', 'Telephone']
['ID', 'Name', 'Address', '']
['ID', 'Name', 'Address']
['ID', 'Name', 'Address', 'Telephone']
```

DATAENGINEERING class

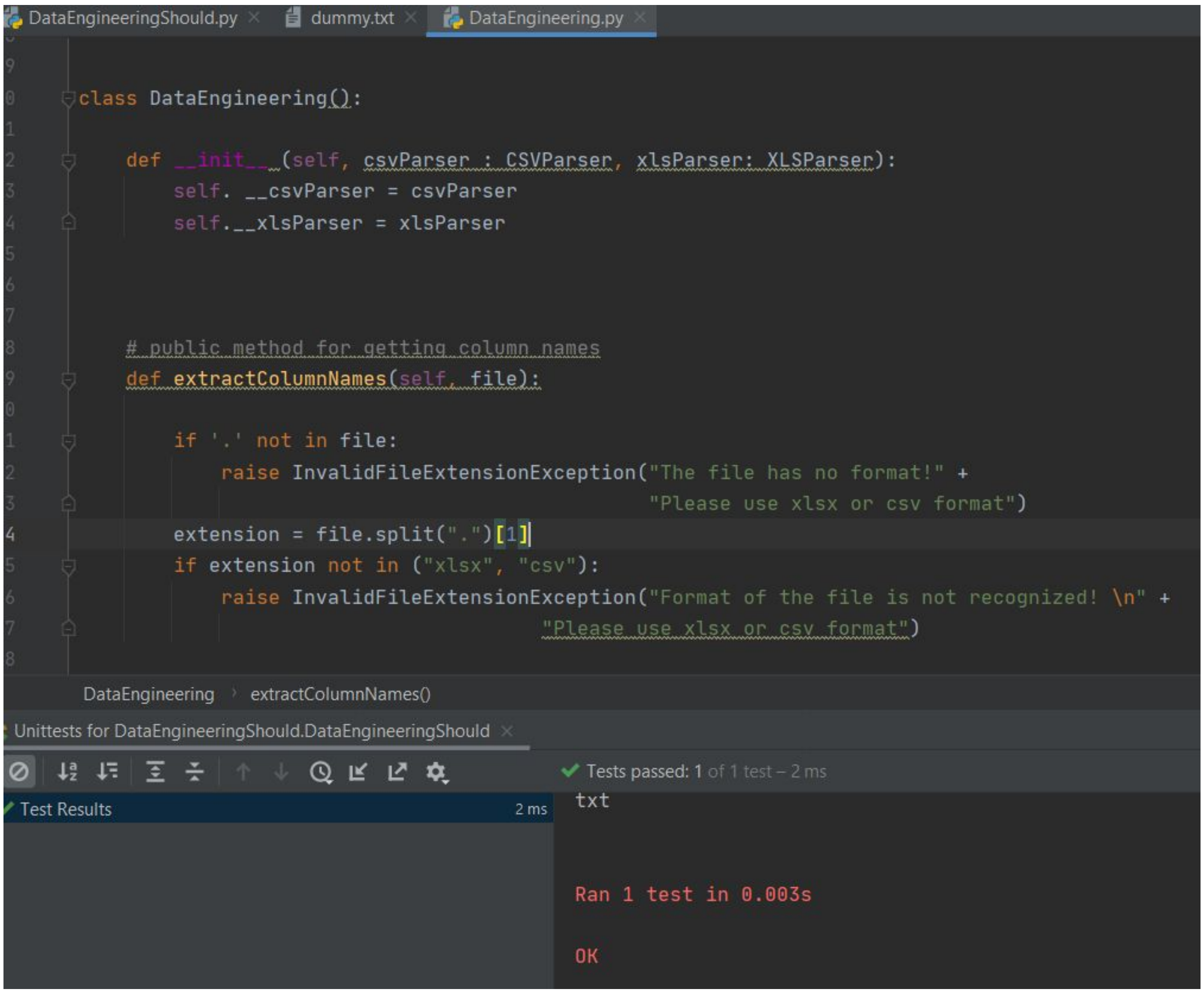
First of all, we would like to ensure that our module supports only xlsx and csv formats, so we will write a test case which shall fail (for example, for extracting columns)



```python
from unittest import TestCase

from DataEngineeringService.CSVParser import CSVParser
from DataEngineeringService.DataEngineering import DataEngineering
from DataEngineeringService.XLSParser import XLSParser
from Exceptions.InvalidFileExtensionException import InvalidFileExtensionException


class DataEngineeringShould(TestCase):

    dataEngineering = DataEngineering(CSVParser(), XLSParser())
    wrongExtensionFile = 'dummy.txt'

    def test_failWhenInvalidExtension(self):
        with self.assertRaises(InvalidFileExtensionException):
            self.dataEngineering.extractColumnNames(self.wrongExtensionFile)
```

```
Tests failed: 1 of 1 test – 3 ms
Test Results 3 ms
  DataEngineeringShould 3 ms
    DataEngineeringShould 3 ms
      test_failWhenInvalidExtension 3 ms

Testing started at 12:33 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.
Launching unittests with arguments python -
```

Filling the code, we will obtain

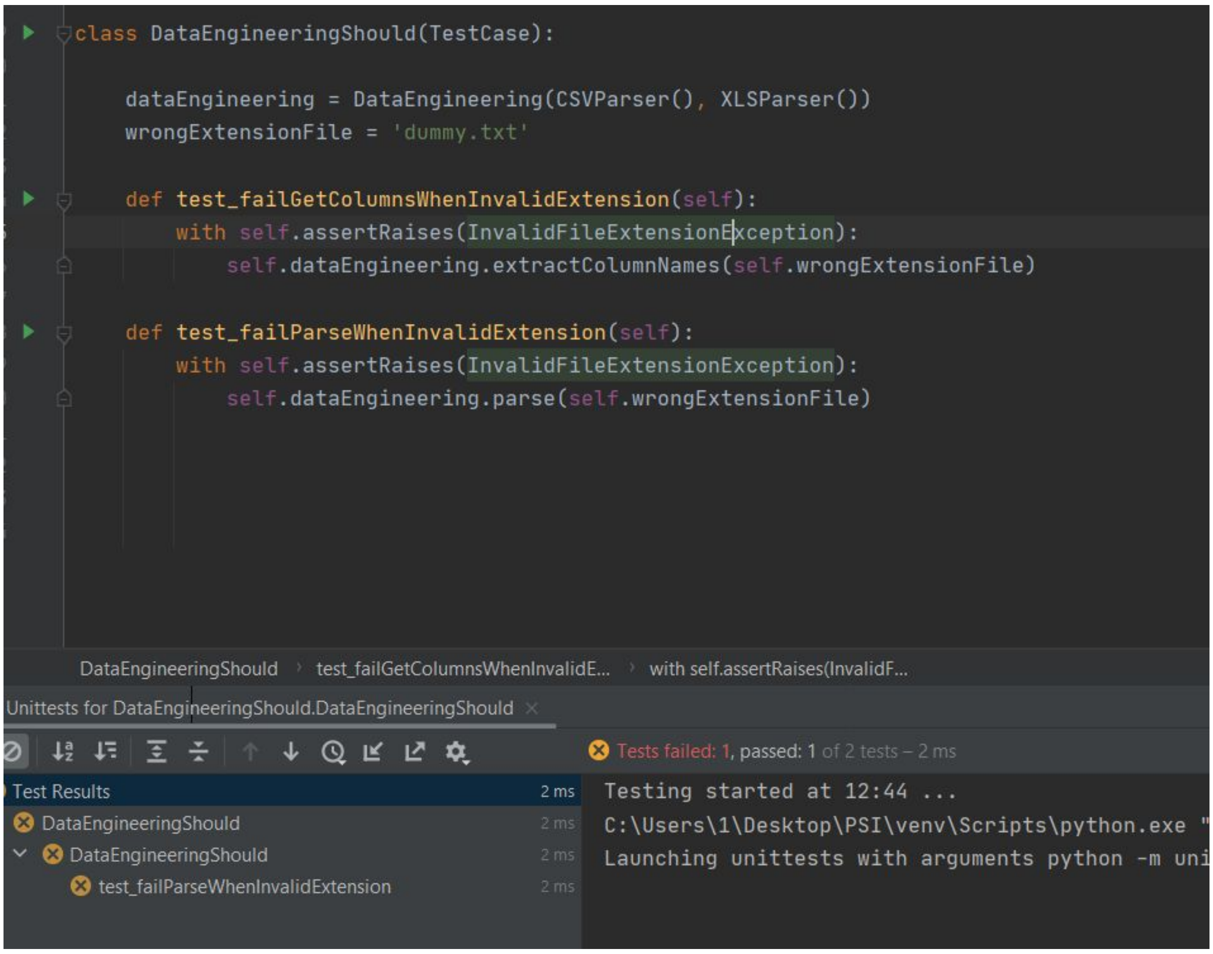


```python
class DataEngineering():

    def __init__(self, csvParser : CSVParser, xlsParser: XLSParser):
        self. __csvParser = csvParser
        self.__xlsParser = xlsParser



    # public method for getting column names
    def extractColumnNames(self, file):

        if '.' not in file:
            raise InvalidFileExtensionException("The file has no format!" +
                                                "Please use xlsx or csv format")
        extension = file.split(".")[1]
        if extension not in ("xlsx", "csv"):
            raise InvalidFileExtensionException("Format of the file is not recognized! \n" +
                                                "Please use xlsx or csv format")
```

DataEngineering › extractColumnNames()
Unittests for DataEngineeringShould.DataEngineeringShould
Test Results 2 ms

```
Tests passed: 1 of 1 test – 2 ms
txt


Ran 1 test in 0.003s

OK
```

Same for parse function: first, we must ensure that extension is valid, so we will add test

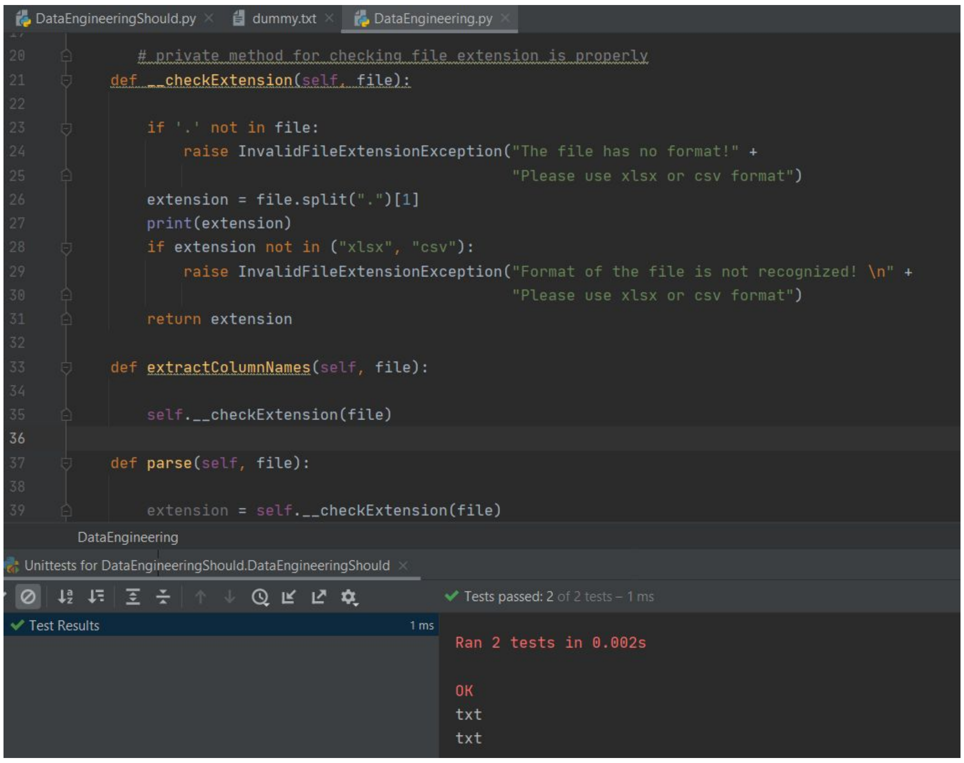


```python
class DataEngineeringShould(TestCase):

    dataEngineering = DataEngineering(CSVParser(), XLSParser())
    wrongExtensionFile = 'dummy.txt'

    def test_failGetColumnsWhenInvalidExtension(self):
        with self.assertRaises(InvalidFileExtensionException):
            self.dataEngineering.extractColumnNames(self.wrongExtensionFile)

    def test_failParseWhenInvalidExtension(self):
        with self.assertRaises(InvalidFileExtensionException):
            self.dataEngineering.parse(self.wrongExtensionFile)
```

DataEngineeringShould › test_failGetColumnsWhenInvalidE... › with self.assertRaises(InvalidF...
Unittests for DataEngineeringShould.DataEngineeringShould

```
Tests failed: 1, passed: 1 of 2 tests – 2 ms

Test Results                          2 ms
  DataEngineeringShould               2 ms
  DataEngineeringShould               2 ms
    test_failParseWhenInvalidExtension 2 ms

Testing started at 12:44 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "
Launching unittests with arguments python -m uni
```

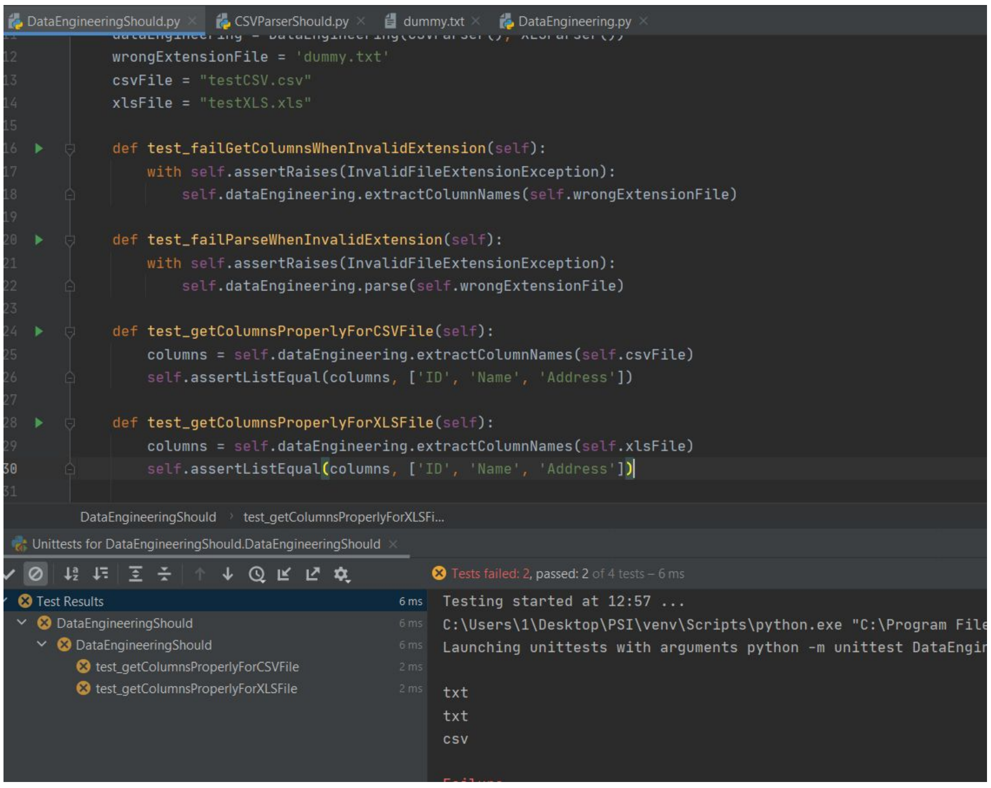
and the corresponding code (with REAFCTORING -- extracting checker extension method)



```python
    # private method for checking file extension is properly
    def __checkExtension(self, file):

        if '.' not in file:
            raise InvalidFileExtensionException("The file has no format!" +
                                                "Please use xlsx or csv format")
        extension = file.split(".")[1]
        print(extension)
        if extension not in ("xlsx", "csv"):
            raise InvalidFileExtensionException("Format of the file is not recognized! \n" +
                                                "Please use xlsx or csv format")
        return extension

    def extractColumnNames(self, file):

        self.__checkExtension(file)

    def parse(self, file):

        extension = self.__checkExtension(file)
```

```
Tests passed: 2 of 2 tests – 1 ms

Ran 2 tests in 0.002s

OK
txt
txt
```

Now, let’s write tests for columns getter

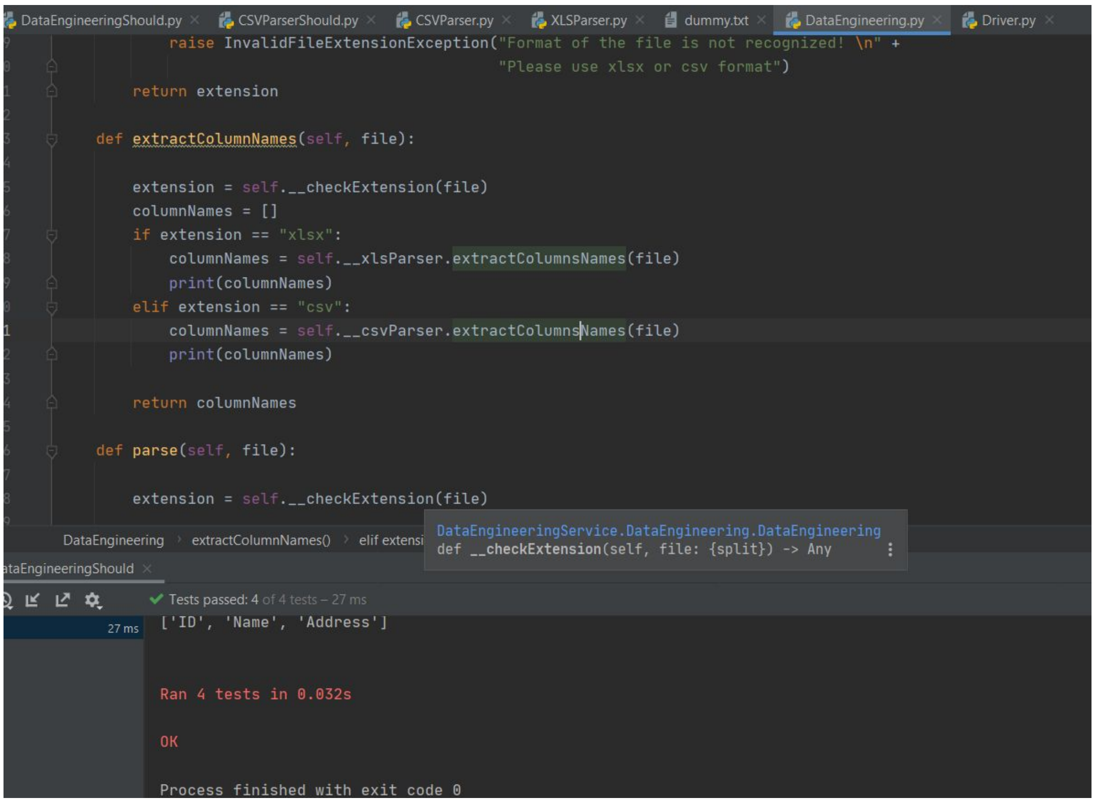


```python
    wrongExtensionFile = 'dummy.txt'
    csvFile = "testCSV.csv"
    xlsFile = "testXLS.xls"

    def test_failGetColumnsWhenInvalidExtension(self):
        with self.assertRaises(InvalidFileExtensionException):
            self.dataEngineering.extractColumnNames(self.wrongExtensionFile)

    def test_failParseWhenInvalidExtension(self):
        with self.assertRaises(InvalidFileExtensionException):
            self.dataEngineering.parse(self.wrongExtensionFile)

    def test_getColumnsProperlyForCSVFile(self):
        columns = self.dataEngineering.extractColumnNames(self.csvFile)
        self.assertListEqual(columns, ['ID', 'Name', 'Address'])

    def test_getColumnsProperlyForXLSFile(self):
        columns = self.dataEngineering.extractColumnNames(self.xlsFile)
        self.assertListEqual(columns, ['ID', 'Name', 'Address'])
```

DataEngineeringShould › test_getColumnsProperlyForXLSFi...
Unittests for DataEngineeringShould.DataEngineeringShould
Test Results 6 ms
DataEngineeringShould 6 ms
DataEngineeringShould 6 ms
test_getColumnsProperlyForCSVFile 2 ms
test_getColumnsProperlyForXLSFile 2 ms

```
Tests failed: 2, passed: 2 of 4 tests – 6 ms

Testing started at 12:57 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program File
Launching unittests with arguments python -m unittest DataEngir

txt
txt
csv
```

Writing the code, we will obtain



DataEngineeringShould.py × CSVParserShould.py × CSVParser.py × XLSParser.py × dummy.txt × DataEngineering.py × Driver.py ×

```
            raise InvalidFileExtensionException("Format of the file is not recognized! \n" +
                                                "Please use xlsx or csv format")

        return extension

    def extractColumnNames(self, file):

        extension = self.__checkExtension(file)
        columnNames = []
        if extension == "xlsx":
            columnNames = self.__xlsParser.extractColumnsNames(file)
            print(columnNames)
        elif extension == "csv":
            columnNames = self.__csvParser.extractColumnsNames(file)
            print(columnNames)

        return columnNames

    def parse(self, file):

        extension = self.__checkExtension(file)
```

DataEngineering › extractColumnNames() › elif extensi

```
DataEngineeringService.DataEngineering.DataEngineering
def __checkExtension(self, file: {split}) -> Any
```

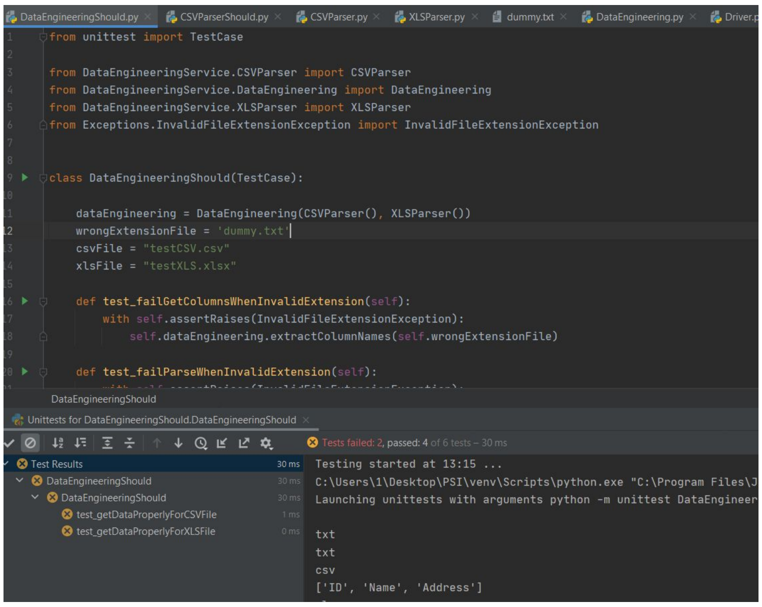
ataEngineeringShould ×

Tests passed: 4 of 4 tests – 27 ms

27 ms

```
['ID', 'Name', 'Address']


Ran 4 tests in 0.032s

OK

Process finished with exit code 0
```

and for parse function



DataEngineeringShould.py × CSVParserShould.py × CSVParser.py × XLSParser.py × dummy.txt × DataEngineering.py × Driver.p

```
from unittest import TestCase

from DataEngineeringService.CSVParser import CSVParser
from DataEngineeringService.DataEngineering import DataEngineering
from DataEngineeringService.XLSParser import XLSParser
from Exceptions.InvalidFileExtensionException import InvalidFileExtensionException


class DataEngineeringShould(TestCase):

    dataEngineering = DataEngineering(CSVParser(), XLSParser())
    wrongExtensionFile = 'dummy.txt'
    csvFile = "testCSV.csv"
    xlsFile = "testXLS.xlsx"

    def test_failGetColumnsWhenInvalidExtension(self):
        with self.assertRaises(InvalidFileExtensionException):
            self.dataEngineering.extractColumnNames(self.wrongExtensionFile)

    def test_failParseWhenInvalidExtension(self):
```

DataEngineeringShould

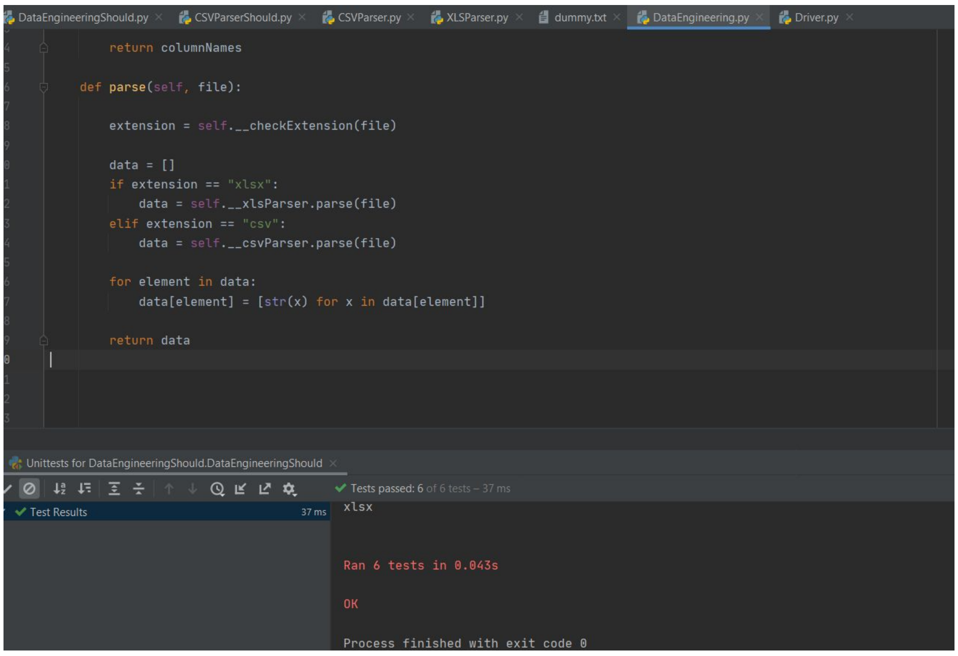
Unittests for DataEngineeringShould.DataEngineeringShould ×

Tests failed: 2, passed: 4 of 6 tests – 30 ms

```
Test Results                          30 ms
  DataEngineeringShould               30 ms
    DataEngineeringShould             30 ms
      test_getDataProperlyForCSVFile   1 ms
      test_getDataProperlyForXLSFile   0 ms
```

```
Testing started at 13:15 ...
C:\Users\1\Desktop\PSI\venv\Scripts\python.exe "C:\Program Files\J
Launching unittests with arguments python -m unittest DataEngineer

txt
txt
csv
['ID', 'Name', 'Address']
```

writing the code



DataEngineeringShould.py × CSVParserShould.py × CSVParser.py × XLSParser.py × dummy.txt × DataEngineering.py × Driver.py ×

```
        return columnNames

    def parse(self, file):

        extension = self.__checkExtension(file)

        data = []
        if extension == "xlsx":
            data = self.__xlsParser.parse(file)
        elif extension == "csv":
            data = self.__csvParser.parse(file)

        for element in data:
            data[element] = [str(x) for x in data[element]]

        return data
```

Unittests for DataEngineeringShould.DataEngineeringShould ×

Tests passed: 6 of 6 tests – 37 ms

Test Results 37 ms

```
xlsx


Ran 6 tests in 0.043s

OK

Process finished with exit code 0
```