



МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение  
высшего образования*

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

---

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных  
технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 3**

**по дисциплине**

**«Тестирование и верификация программного обеспечения»**

Выполнил:

студент группы ИКБО-36-22

Шумахер М.Е.

Проверил:

Доцент

Чернов Е.А.

Москва 2024

## 1 Реализация методологии TDD

Вариант 14: Шифратор и дешифратор файлов в заданной папке (шифруем все файлы и расшифровываем несколькими методами).

Создадим тесты, описывающие поведение функции.

```
D: > code > 5 > Тестирование > 3 > test.py > ...
1  import unittest
2  import os
3  from main import *
4
5  class TestFileEncryption(unittest.TestCase):
6      test_folder = 'D:\code\5\Тестирование\3\test_folder'
7
8      def test_encrypt_decrypt_file_caesar(self):
9          encrypt_file(os.path.join(self.test_folder, "test.txt"), 3)
10         with open(os.path.join(self.test_folder, "test.txt"), 'r') as f:
11             content = f.read()
12             self.assertEqual(content, "Khoor Zruog!")
13         decrypt_file(os.path.join(self.test_folder, "test.txt"), 3)
14         with open(os.path.join(self.test_folder, "test.txt"), 'r') as f:
15             content = f.read()
16             self.assertEqual(content, "Hello World!")
17
18         def test_encrypt_decrypt_file_xor(self):
19             encrypt_file_xor(os.path.join(self.test_folder, "test.txt"), 123)
20             with open(os.path.join(self.test_folder, "test.txt"), 'r') as f:
21                 content = f.read()
22                 self.assertNotEqual(content, "3\xe\x17\x17\x14[\x14\t\x17\x1fc")
23             decrypt_file_xor(os.path.join(self.test_folder, "test.txt"), 123)
24             with open(os.path.join(self.test_folder, "test.txt"), 'r') as f:
25                 content = f.read()
26                 self.assertEqual(content, "Hello World!")
27
28     if __name__ == '__main__':
29         unittest.main()
```

Рисунок 1 – Код тестов

Здесь мы используем 2 метода шифрования и прогоняем в тестах для проверки правильности шифрования и дешифрования.

Теперь реализуем функции шифрования и дешифрования.

```

D: > code > 5 > Тестирование > 3 > main.py > ...
1  def caesar_cipher(text, shift):
2      encrypted = ""
3      for char in text:
4          if char.isalpha():
5              shift_base = 65 if char.isupper() else 97
6              encrypted += chr((ord(char) + shift - shift_base) % 26 + shift_base)
7          else:
8              encrypted += char
9      return encrypted
10
11 def encrypt_file(file_path, shift):
12     with open(file_path, 'r') as file:
13         content = file.read()
14     encrypted_content = caesar_cipher(content, shift)
15     with open(file_path, 'w') as file:
16         file.write(encrypted_content)
17
18 def decrypt_file(file_path, shift):
19     with open(file_path, 'r') as file:
20         content = file.read()
21     decrypted_content = caesar_cipher(content, -shift)
22     with open(file_path, 'w') as file:
23         file.write(decrypted_content)
24
25 def xor_cipher(text, key):
26     encrypted = "".join(chr(ord(char) ^ key) for char in text)
27     return encrypted
28
29 def encrypt_file_xor(file_path, key):
30     with open(file_path, 'r') as file:
31         content = file.read()
32     encrypted_content = xor_cipher(content, key)
33     with open(file_path, 'w') as file:
34         file.write(encrypted_content)
35
36 def decrypt_file_xor(file_path, key):
37     with open(file_path, 'r') as file:
38         content = file.read()
39     decrypted_content = xor_cipher(content, key)
40     with open(file_path, 'w') as file:
41         file.write(decrypted_content)

```

Рисунок 2 – Код реализации функций

Запускаем тесты и проверяем успех их прохождения.

```

PS C:\Users\shumila> python -u "d:\code\5\Тестирование\3\test.py"
..
-----
Ran 2 tests in 0.003s

OK

```

Рисунок 3 – Запуск тестов

Как видно из рисунка, тесты были пройдены успешно.

## 2 Реализация методологии BDD

Описание сценариев BDD для функциональности шифратора и дешифратора:

Feature: File encryption and decryption

Scenario: Encrypt and decrypt file using Caesar cipher

Given a folder with a file containing "Hello World!"

When I encrypt the file with Caesar cipher and shift 3

Then the file content should be "Khoor Zruog!"

When I decrypt the file with Caesar cipher and shift 3

Then the file content must be "Hello World!"

Scenario: Encrypt and decrypt file using XOR cipher

Given a folder with a file containing "Hello World!"

When I encrypt the file with XOR cipher and key 123

Then the file content should not be "Hello World!"

When I decrypt the file with XOR cipher and key 123

Then the file content should be "Hello World!"

Автоматизация сценариев с использованием behave python:

```

D: > code > 5 > Тестирование > 3 > steps > steps.py > ...
5 # Задаем путь к тестовой папке
6 TEST_FOLDER = r'D:\code\5\Тестирование\3\test_folder'
7
8 @given('a folder with a file containing "Hello World!')
9 def create_test_file(context):
10     context.test_file = os.path.join(TEST_FOLDER, "test.txt")
11     with open(context.test_file, 'w') as f:
12         f.write("Hello World!")
13
14 @when('I encrypt the file with Caesar cipher and shift 3')
15 def encrypt_caesar_file(context):
16     encrypt_file(context.test_file, 3)
17
18 @then('the file content should be "Khoor Zruog!')
19 def check_caesar_encryption(context):
20     with open(context.test_file, 'r') as f:
21         content = f.read()
22     assert content == "Khoor Zruog!"
23
24 @when('I decrypt the file with Caesar cipher and shift 3')
25 def decrypt_caesar_file(context):
26     decrypt_file(context.test_file, 3)
27
28 @then('the file content must be "Hello World!')
29 def check_caesar_decryption(context):
30     with open(context.test_file, 'r') as f:
31         content = f.read()
32     assert content == "Hello World!"
33
34 @when('I encrypt the file with XOR cipher and key 123')
35 def encrypt_xor_file(context):
36     encrypt_file_xor(context.test_file, 123)
37
38 @then('the file content should not be "Hello World!')
39 def check_xor_encryption(context):
40     with open(context.test_file, 'r') as f:
41         content = f.read()
42     assert content != "Hello World!"
43
44 @when('I decrypt the file with XOR cipher and key 123')
45 def decrypt_xor_file(context):
46     decrypt_file_xor(context.test_file, 123)
47
48 @then('the file content should be "Hello World!')
49 def check_xor_decryption(context):
50     with open(context.test_file, 'r') as f:
51         content = f.read()
52     assert content == "Hello World!"
53

```

Рисунок 4 – Тесты на behave

Пишем реализацию шифратора дешифратора.

```
D: > code > 5 > Тестирование > 3 > main.py > ...
1  def caesar_cipher(text, shift):
2      encrypted = ""
3      for char in text:
4          if char.isalpha():
5              shift_base = 65 if char.isupper() else 97
6              encrypted += chr((ord(char) + shift - shift_base) % 26 + shift_base)
7          else:
8              encrypted += char
9      return encrypted
10
11 def encrypt_file(file_path, shift):
12     with open(file_path, 'r') as file:
13         content = file.read()
14     encrypted_content = caesar_cipher(content, shift)
15     with open(file_path, 'w') as file:
16         file.write(encrypted_content)
17
18 def decrypt_file(file_path, shift):
19     with open(file_path, 'r') as file:
20         content = file.read()
21     decrypted_content = caesar_cipher(content, -shift)
22     with open(file_path, 'w') as file:
23         file.write(decrypted_content)
24
25 def xor_cipher(text, key):
26     encrypted = "".join(chr(ord(char) ^ key) for char in text)
27     return encrypted
28
29 def encrypt_file_xor(file_path, key):
30     with open(file_path, 'r') as file:
31         content = file.read()
32     encrypted_content = xor_cipher(content, key)
33     with open(file_path, 'w') as file:
34         file.write(encrypted_content)
35
36 def decrypt_file_xor(file_path, key):
37     with open(file_path, 'r') as file:
38         content = file.read()
39     decrypted_content = xor_cipher(content, key)
40     with open(file_path, 'w') as file:
41         file.write(decrypted_content)
```

Рисунок 5 – Код реализации функций

Запускаем тесты и смотрим на их результат.

```

PS D:\code\5\Тестирование\3> behave
Feature: File encryption and decryption # steps/encryption.feature:1

  Scenario: Encrypt and decrypt file using Caesar cipher # steps/encryption.feature:3
    Given a folder with a file containing "Hello World!" # steps/steps.py:8
    When I encrypt the file with Caesar cipher and shift 3 # steps/steps.py:14
    Then the file content should be "Khoor Zruog!" # steps/steps.py:18
    When I decrypt the file with Caesar cipher and shift 3 # steps/steps.py:24
    Then the file content must be "Hello World!" # steps/steps.py:28

  Scenario: Encrypt and decrypt file using XOR cipher # steps/encryption.feature:10
    Given a folder with a file containing "Hello World!" # steps/steps.py:8
    When I encrypt the file with XOR cipher and key 123 # steps/steps.py:34
    Then the file content should not be "Hello World!" # steps/steps.py:38
    When I decrypt the file with XOR cipher and key 123 # steps/steps.py:44
    Then the file content should be "Hello World!" # steps/steps.py:48

1 feature passed, 0 failed, 0 skipped
2 scenarios passed, 0 failed, 0 skipped
10 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.005s

```

Рисунок 6 – Тестирование

Все тесты пройдены успешно.

### Вывод

В процессе работы, применяя методологии TDD и BDD, был создан модуль программы, начальная реализация которого велась через написание тестов, что позволило эффективно выявить ошибки на ранних этапах разработки. Юнит-тесты в рамках TDD помогли контролировать соответствие программы ожидаемому поведению, а сценарии BDD обеспечили прозрачность требований.