|  |  |  |
| --- | --- | --- |
| Защищено:  Нардид А.Н.    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |  | Демонстрация:  Нардид А.Н.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

**Отчет по рубежному контролю № 2 по курсу**

**Парадигмы и конструкции языков программирования**

**ГУИМЦ**

#### Тема работы: "Рубежный контроль представляет собой разработку тестов на языке Python."

4

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы ИУ5Ц-53Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Рабинович А. Д. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

Москва, МГТУ - 2024 г.

# Тема и задание для выполнения рубежного контроля.

Тема работы: Рубежный контроль представляет собой разработку тестов на языке Python.

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

# Листинг программы

**syntax\_analysis.py**

from operator import itemgetter

class Syntax:

def \_\_init\_\_(self, id, name, lang\_id):

self.id = id

self.name = name

self.lang\_id = lang\_id

class Language:

def \_\_init\_\_(self, id, name):

self.id = id

self.name = name

class SyntaxLang:

def \_\_init\_\_(self, lang\_id, syntax\_id):

self.lang\_id = lang\_id

self.syntax\_id = syntax\_id

def one\_to\_many\_relationship(languages, syntaxes):

return [(s.name, l.name) for l in languages for s in syntaxes if s.lang\_id == l.id]

def many\_to\_many\_relationship(languages, syntaxes, syntaxes\_langs):

many\_to\_many\_temp = [(l.name, sl.lang\_id, sl.syntax\_id)

for l in languages for sl in syntaxes\_langs if l.id == sl.lang\_id]

return [(s.name, l\_name) for l\_name, \_, syntax\_id in many\_to\_many\_temp for s in syntaxes if s.id == syntax\_id]

def task\_v1\_python\_syntax(one\_to\_many):

return [name for name, lang in one\_to\_many if lang == 'Python']

def task\_v2\_min\_syntax(languages, one\_to\_many):

res\_2\_unsorted = []

for l in languages:

l\_syntaxes = list(filter(lambda x: x[1] == l.name, one\_to\_many))

if l\_syntaxes:

min\_syntax = min([syntax for syntax, \_ in l\_syntaxes])

res\_2\_unsorted.append((l.name, min\_syntax))

return sorted(res\_2\_unsorted, key=lambda x: x[1])

def task\_v3\_syntax\_language(many\_to\_many):

return sorted(many\_to\_many, key=lambda x: x[0])

**test\_syntax\_analysis.py**

import unittest

from syntax\_analysis import (

Language, Syntax, SyntaxLang,

one\_to\_many\_relationship, many\_to\_many\_relationship,

task\_v1\_python\_syntax, task\_v2\_min\_syntax, task\_v3\_syntax\_language

)

class TestSyntaxAnalysis(unittest.TestCase):

def setUp(self):

self.languages = [

Language(1, 'Python'),

Language(2, 'C++'),

Language(3, 'JavaScript'),

]

self.syntaxes = [

Syntax(1, 'Цикл for', 1),

Syntax(2, 'Условие if', 2),

Syntax(3, 'Функция', 1),

Syntax(4, 'Объект', 3),

Syntax(5, 'Массив', 2),

]

self.syntaxes\_langs = [

SyntaxLang(1, 1),

SyntaxLang(1, 3),

SyntaxLang(2, 2),

SyntaxLang(2, 5),

SyntaxLang(3, 4),

]

def test\_task\_v1(self):

one\_to\_many = one\_to\_many\_relationship(self.languages, self.syntaxes)

result = task\_v1\_python\_syntax(one\_to\_many)

self.assertEqual(result, ['Цикл for', 'Функция'])

def test\_task\_v2(self):

one\_to\_many = one\_to\_many\_relationship(self.languages, self.syntaxes)

result = task\_v2\_min\_syntax(self.languages, one\_to\_many)

self.assertEqual(result, [('C++', 'Массив'), ('JavaScript', 'Объект'), ('Python', 'Функция')])

def test\_task\_v3(self):

many\_to\_many = many\_to\_many\_relationship(self.languages, self.syntaxes, self.syntaxes\_langs)

result = task\_v3\_syntax\_language(many\_to\_many)

self.assertEqual(result, [

('Массив', 'C++'),

('Объект', 'JavaScript'),

('Условие if', 'C++'),

('Функция', 'Python'),

('Цикл for', 'Python'),

])

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()