МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА
(ЛЕНИНА)

Кафедра Вычислительной
техники

ОТЧЕТ
по лабораторной работе №8
по дисциплине
«Объектно-ориентированное программирование»
Тема: «Модульное тестирование приложения»

Преподаватель Гречухин М.Н.
Студент группы 3316 Петренко В.С.

Санкт-Петербург
2024

Цель работы: знакомство с технологией модульного тестирования Java-приложений с использованием системы JUnit.

Задачи:

1. Создать новый проект, который будет дублировать проект лабораторной работы № 3.
2. Проанализировать классы приложения и определить, какие методы необходимо протестировать.
3. Написать JUnit-тесты для выбранных методов.
4. Запустить тесты и снять с экрана скриншоты, иллюстрирующие выполнение тестов.
5. Сгенерировать документацию с помощью Javadoc и просмотреть ее в браузере.

Тестируемый метод: save_student - Метод класса MainApp, отвечающий за сохранение информации о студенте в XML-файл.

Исходные тексты классов тестов:

```
import unittest
from unittest.mock import patch, MagicMock
from app import Person, Student, Teacher, Group, MainApp, AppException  # Импортируем классы

class TestStudent(unittest.TestCase):

    @patch('app.ET.parse')  # Мокируем метод parse для работы с XML
    @patch('app.ET.ElementTree')  # Мокируем ElementTree для предотвращения реального сохранения
    def test_save_student(self, MockElementTree, MockParse):
        # Настройка мока для XML
        mock_tree = MagicMock()
        mock_root = MagicMock()
        MockParse.return_value = mock_tree
        mock_tree.getroot.return_value = mock_root

        # Создание экземпляра MainApp для вызова метода save_student
        app = MainApp(MagicMock())
        app.first_name_entry = MagicMock()
        app.last_name_entry = MagicMock()
        app.patronymic_entry = MagicMock()
        app.group_entry = MagicMock()
        app.status_entry = MagicMock()

        # Настройка значений, которые будут введены в поля
        app.first_name_entry.get.return_value = "Иван"
        app.last_name_entry.get.return_value = "Иванов"
        app.patronymic_entry.get.return_value = "Иванович"
        app.group_entry.get.return_value = "101"
        app.status_entry.get.return_value = "Активный"

        app.save_student()
```

```
        mock_tree.write.assert_called_with('students.xml')

        mock_root.append.assert_called_once()

if __name__ == "__main__":
    unittest.main()
```

Скриншоты, иллюстрирующие выполнение тестов