



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Отчет по практической работе №5

по дисциплине «Тестирование и верификация ПО»

Выполнил:

Студент группы ИКБО-25-22

Ракитин В.А.

Проверил:

Ассистент

Овчинникова М.А.

2024 г.

СОДЕРЖАНИЕ

Введение	3
1 Описание программ, выбранных для тестирования	4
2 Проверка приложений при помощи статических анализаторов	5
3 Добавление ошибок для статистического анализа	6
4 Проверка приложений при помощи динамических анализаторов.....	8
5 Внесение ошибок для динамического анализа	9
Заключение	10
Приложение	11

Введение

Освоить принципы и методы использования статических и динамических анализаторов кода, научиться выявлять ошибки и потенциальные уязвимости на ранних этапах разработки, а также повысить качество и надёжность программного обеспечения с помощью анализа результатов автоматизированных инструментов.

1 Описание программ, выбранных для тестирования

Калькулятор на Python. Пользователь вводит два числа и операцию, которую необходимо сделать. Далее программа выдает результат. Калькулятор поддерживает операции сложения (+), вычитания (-), умножения (*), деления (/).

Список задач на Java. Пользователь может добавлять в список задачи (add), просматривать их (view), удалять задачи по индексу (remove). Для окончания работы нужно ввести exit.

2 Проверка приложений при помощи статических анализаторов

На рисунке 1.1 представлена проверка программы при помощи анализатора pylint, который проверяет оформление кода по стандарту PEP 8.

```
(venv) PS C:\Users\User\PycharmProjects\StudIT> pylint main.py  
  
-----  
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
```

Рисунок 1.1 – Проверка программы при помощи анализатора pylint

На рисунке 1.2 представлена проверка программы при помощи анализатора mypy, проверяющий на синтаксические ошибки.

```
(venv) PS C:\Users\User\PycharmProjects\StudIT> mypy main.py  
Success: no issues found in 1 source file
```

Рисунок 1.2 – Проверка программы при помощи анализатора mypy

На рисунке 1.3 представлена проверка программы при помощи анализатора pydocstyle, проверяющий документацию проекта.

```
(venv) PS C:\Users\User\PycharmProjects\StudIT> pydocstyle main.py  
(venv) PS C:\Users\User\PycharmProjects\StudIT>
```

Рисунок 1.3 – Проверка программы при помощи анализатора pydocstyle

На рисунке 1.4 представлена проверка встроенным анализатор кода в java

```
✓ Inspection Results 'Project Default' profile 5 warnings 19 typos  
  ✓ Java 4 warnings  
    > Code style issues 1 warning  
    ✓ Declaration redundancy 2 warnings  
      ✓ Declaration can have 'final' modifier 1 warning  
        ✓ 🔄 📄 TodoList 1 warning  
          Declaration can have final modifier  
        ✓ Unnecessary module dependency 1 warning  
          ✓ 📄 untyped1.test 1 warning  
            Module 'untyped1.test' sources do not depend on module 'untyped1.main' sources  
      > Java language level migration aids 1 warning  
    ✓ Proofreading 19 typos  
      ✓ Typo 19 typos  
        > ⚙️ gradle-wrapper.properties 2 typos
```

Рисунок 1.4 – Проверка кода при помощи встроенного в IDEA анализатора

3 Добавление ошибок для статистического анализа

На рисунке 1.5 представлена проверка программы с ошибками анализатором pylint.

```
(venv) PS C:\Users\User\PycharmProjects\StudIT> pylint main.py
***** Module main
main.py:4:0: C0325: Unnecessary parens after '=' keyword (superfluous-parens)
main.py:6:0: C0325: Unnecessary parens after '=' keyword (superfluous-parens)
main.py:23:0: C0304: Final newline missing (missing-final-newline)
main.py:1:0: C0114: Missing module docstring (missing-module-docstring)
main.py:1:0: C0116: Missing function or method docstring (missing-function-docstring)

-----
Your code has been rated at 7.37/10 (previous run: 7.89/10, -0.53)
```

Рисунок 1.5 – Проверка программы с ошибками анализатором pylint

На рисунке 1.6 представлена проверка программы с ошибками анализатором mypy.

```
(venv) PS C:\Users\User\PycharmProjects\StudIT> mypy main.py
Success: no issues found in 1 source file
```

Рисунок 1.6 – Проверка программы с ошибками анализатором mypy

На рисунке 1.7 представлена проверка программы с ошибками анализатором pydocstyle.

```
(venv) PS C:\Users\User\PycharmProjects\StudIT> pydocstyle main.py
main.py:1 at module level:
    D100: Missing docstring in public module
main.py:1 in public function `calculator`:
    D103: Missing docstring in public function
```

Рисунок 1.7 – Проверка программы с ошибками анализатором pydocstyle

На рисунке 1.8 представлена проверка программы с ошибками встроенным в IDEA анализатором.



Рисунок 1.8 – Проверка кода с ошибками встроенным в IDEA анализатором

4 Проверка приложений при помощи динамических анализаторов

На рисунке 1.9 представлена проверка динамическим тестом калькулятора Python.

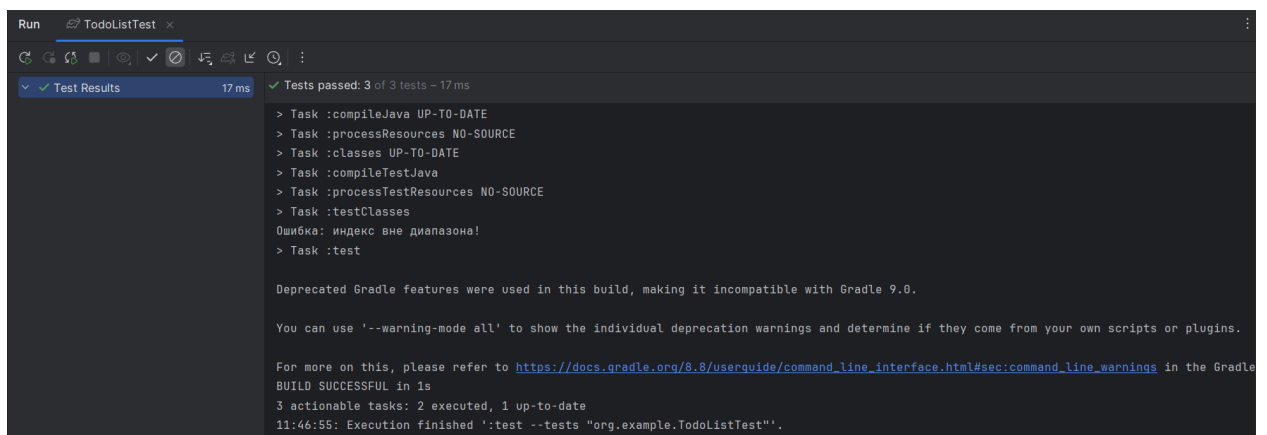
```
(venv) PS C:\Users\User\PycharmProjects\StudIT> python -m unittest -v C:\Users\User\PycharmProjects\StudIT\dd.py
test_addition (dd.TestCalculator) ... ok
test_division (dd.TestCalculator) ... ok
test_invalid_operation (dd.TestCalculator) ... ok
test_multiplication (dd.TestCalculator) ... ok
test_subtraction (dd.TestCalculator) ... ok

-----
Ran 5 tests in 0.001s

OK
```

Рисунок 1.9 – Динамическая проверка калькулятора Python

На рисунке 1.10 представлена проверка динамическим тестом Java кода на время работы и ошибки программы.



The screenshot shows the 'Run' window of an IDE with the 'Test Results' tab selected. The test 'TodoListTest' has passed in 17 ms. The output pane shows the following tasks: > Task :compileJava UP-TO-DATE, > Task :processResources NO-SOURCE, > Task :classes UP-TO-DATE, > Task :compileTestJava, > Task :processTestResources NO-SOURCE, > Task :testClasses, Ошибка: индекс вне диапазона!, and > Task :test. Below the tasks, a message states: 'Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0. You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins. For more on this, please refer to https://docs.gradle.org/8.8/userguide/command_line_interface.html#sec:command_line_warnings in the Gradle documentation. BUILD SUCCESSFUL in 1s 3 actionable tasks: 2 executed, 1 up-to-date 11:46:55: Execution finished ':test --tests "org.example.TODOListTest"'.

Рисунок 1.10 – Динамическая проверка java кода

5 Внесение ошибок для динамического анализа

На рисунке 1.11 представлена проверка динамическим тестом кода на время работы и ошибки программы.

```
(venv) PS C:\Users\User\PycharmProjects\StudIT> python -m unittest -v C:\Users\User\PycharmProjects\StudIT\dd.py
test_addition (dd.TestCalculator) ... FAIL
test_division (dd.TestCalculator) ... FAIL
test_invalid_operation (dd.TestCalculator) ... ok
test_multiplication (dd.TestCalculator) ... FAIL
test_subtraction (dd.TestCalculator) ... FAIL

=====
FAIL: test_addition (dd.TestCalculator)
-----
Traceback (most recent call last):
  File "C:\Users\User\PycharmProjects\StudIT\dd.py", line 9, in test_addition
    self.assertEqual(calculate(1, 2, '+'), 3)
AssertionError: -1 != 3

=====
FAIL: test_division (dd.TestCalculator)
-----
Traceback (most recent call last):
  File "C:\Users\User\PycharmProjects\StudIT\dd.py", line 24, in test_division
    self.assertEqual(calculate(6, 3, '/'), 2)
FAIL: test_subtraction (dd.TestCalculator)
-----
```

Рисунок 1.11 – Динамическая проверка python кода

На рисунке 1.12 представлена проверка динамическим тестом кода проверка на время работы и ошибки программы.

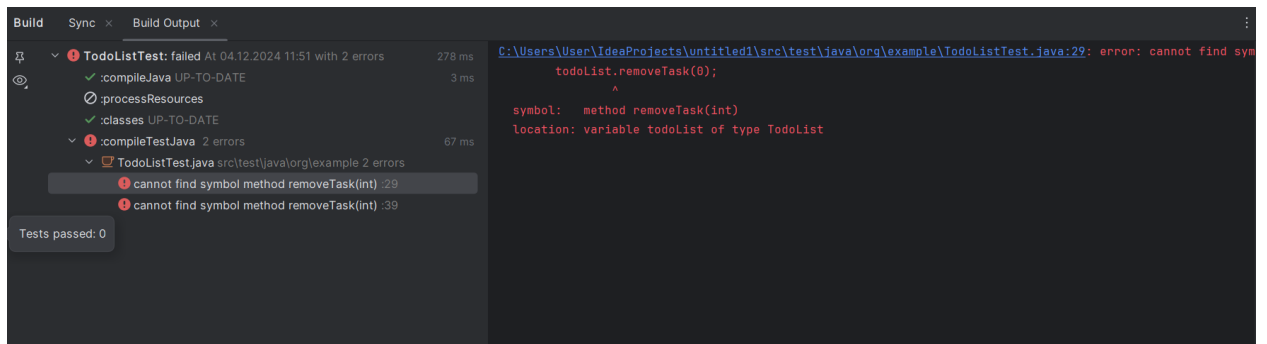


Рисунок 1.12 – Динамическая проверка java кода

Заключение

Статистические анализы находят первые ошибки, а не все ошибки кода. В ходе практической работы были освоены и изучены методы статического тестирования и динамического.

Приложение

Листинг 1 – Арифметический калькулятор на Python

```
"""
Этот модуль реализует простой калькулятор, который выполняет
основные арифметические операции.

Поддерживаемые операции: сложение (+), вычитание (-), умножение
(*) и деление (/).
"""

def calculator():
    """
    Простой калькулятор, который выполняет основные
    арифметические операции.

    Функция запрашивает у пользователя два числа и операцию,
    которую необходимо выполнить.
    Если пользователь пытается выполнить деление на ноль,
    функция выводит сообщение об ошибке.

    Ввод:
        - Первое число (float)
        - Второе число (float)
        - Операция (строка)

    Вывод:
        - Результат операции или сообщение об ошибке.
    """
    print("Введите первое число:")
    num1 = float(input())
    print("Введите второе число:")
    num2 = float(input())
    print("Выберите операцию (+, -, *, /):")
    operation = input()

    if operation == '+':
        print(f"Результат: {num1 + num2}")
    elif operation == '-':
        print(f"Результат: {num1 - num2}")
    elif operation == '*':
        print(f"Результат: {num1 * num2}")
    elif operation == '/':
        if num2 != 0:
            print(f"Результат: {num1 / num2}")
        else:
            print("Ошибка: деление на ноль!")
    else:
        print("Ошибка: неверная операция!")

# Запуск калькулятора
calculator()
```

Листинг 2 – Список задач на Java

```
package org.example;

import java.util.ArrayList;
import java.util.Scanner;

public class TodoList {
    private ArrayList<String> tasks = new ArrayList<>();

    public void addTask(String task) {
        tasks.add(task);
    }

    public void removeTask(int index) {
        if (index >= 0 && index < tasks.size()) {
            tasks.remove(index);
        } else {
            System.out.println("Ошибка: индекс вне диапазона!");
        }
    }

    public void viewTasks() {
        for (int i = 0; i < tasks.size(); i++) {
            System.out.println(i + ": " + tasks.get(i));
        }
    }

    public static void main(String[] args) {
        TodoList todoList = new TodoList();
        Scanner scanner = new Scanner(System.in);
        String command;

        while (true) {
            System.out.println("Введите команду (add, remove, view, exit):");
            command = scanner.nextLine();
            if (command.equals("add")) {
                System.out.println("Введите задачу:");
                String task = scanner.nextLine();
                todoList.addTask(task);
            } else if (command.equals("remove")) {
                System.out.println("Введите индекс задачи для удаления:");
                int index = scanner.nextInt();
                scanner.nextLine(); // очистка буфера
                todoList.removeTask(index);
            } else if (command.equals("view")) {
                todoList.viewTasks();
            } else if (command.equals("exit")) {
                break;
            } else {
                System.out.println("Ошибка: неверная команда!");
            }
        }
        scanner.close();
    }
}
```