Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Факультет информационных технологий и кибербезопасности

Кафедра информационных систем и программирования

**ОТЧЕТ**

по лабораторной работе №11.

Тема работы: «Мутационное тестирование».

Наименование дисциплины: «Тестирование и отладка программного обеспечения».

Подготовил:

Студент группы 23-КБ-ПР1

Юданов М.Д.

Краснодар

2025

**1 Цель работы**

Цель работы – изучить принципы работы мутационного тестирования и использование инструмента Pitest для создания мутационных тестов.

1. **Краткое теоретическое описание**

Мутационное тестирование – это метод проверки качества тестов, при котором исходный код программы намеренно изменяется с помощью мутаций, а тесты проверяют способность выявлять эти изменения. Каждое изменение кода создаёт мутанта, который считается убитым, если тесты его обнаруживают, или выжившим, если тесты проходят успешно. Цель такого тестирования – оценить эффективность набора тестов и выявить участки кода, недостаточно проверяемые тестами. Pitest – это инструмент для мутационного тестирования в экосистеме Java, автоматически создающий мутантов, запускающий тесты и формирующий отчёты о мутационной устойчивости кода.

**3 Задание**

1) Проанализировать качество тестов в проектах из предыдущих лабораторных работ.

2) Расширить тестовые наборы для обнаружения всех мутантов.

3) Оформить отчёт.

**4** **Ход работы**

1-3) Подключил Pitest с помощью Maven, запустил мутационное тестирование 3 вариантов тестов для графа (обычные тесты, тесты с помощью Mockito и тесте с помощью AssertJ), проверил результаты в виде отчёта .html. После первого запуска максимально улучшил тесты (некоторые граничные условия невозможно убить, также Main не был оттестирован). Были изменены классы AssertJGraphTest и Edge. Также оформил отчёт.

package example;

import org.assertj.core.api.Assertions;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Test;

import java.io.IOException;

import java.io.StringReader;

import java.io.StringWriter;

import java.util.ArrayList;

import java.util.List;

public class AssertJGraphTest {

String htmlLineConnected =

"""

<html><head></head><body>\r

<h2>1 2 true</h2>\r

<h2>2 4 true</h2>\r

<h2>4 3 true</h2>\r

<h2>3 1 true</h2>\r

<body></html>\r

""";

String htmlLineIsNotConnected =

"""

<html><head></head><body>\r

<h2>1 2 true</h2>\r

<h2>2 4 true</h2>\r

<h2>4 3 true</h2>\r

<h2>1 3 true</h2>\r

<body></html>\r

""";

String txtLineMatrixConnected =

"""

-1 0 0 1\r

1 -1 0 0\r

0 0 1 -1\r

0 1 -1 0\r

""";

String htmlLineConnectedMatrixConnected =

"""

<html><head></head><body>\r

<h2>-1 0 0 1</h2>\r

<h2>1 -1 0 0</h2>\r

<h2>0 0 1 -1</h2>\r

<h2>0 1 -1 0</h2>\r

</body></html>\r

""";

String txtEdgesConnected =

"""

1 2 true

2 4 true

4 3 true

3 1 true

""";

String txtEdgesConnectedSecond = """

1 2 false

""";

int[][] arrayFromFileConnected = {

{-1,0,0,1},

{1,-1,0,0},

{0,0,1,-1},

{0,1,-1,0}

};

int[][] arrayFromFileConnectedSecond = {

{1},

{1}

};

static List<Edge> listFromFileConnected = new ArrayList<>();

@BeforeAll

static void setListFromFileConnected() {

listFromFileConnected.add(new Edge(1, 2, true));

listFromFileConnected.add(new Edge(2, 4, true));

listFromFileConnected.add(new Edge(4, 3, true));

listFromFileConnected.add(new Edge(3, 1, true));

}

@Test

void readGraph() throws IOException {

try(StringReader reader = new StringReader(htmlLineConnected)) {

Graph graph = new Graph(new GraphReaderWriterHtml(), new GraphReaderWriterTxt());

graph.readFromFile(reader);

Assertions.assertThat(graph.incidenceMatrix)

.as("Считанная матрица равна null!")

.isNotNull()

.as("Считанная матрица пуста!")

.isNotEmpty()

.as("Считанная матрица не равна ожидаемому массиву!")

.isDeepEqualTo(arrayFromFileConnected);

}

}

@Test

void writeGraph() throws IOException {

Graph graph = new Graph(new GraphReaderWriterHtml(), new GraphReaderWriterTxt());

try(StringWriter writer = new StringWriter(); StringReader reader = new StringReader(htmlLineConnected)) {

graph.readFromFile(reader);

graph.writeToFile(writer);

Assertions.assertThat(writer.toString())

.as("Записанная строка пуста!")

.isNotEmpty()

.as("Записанная строка не равна ожидаемой матрице!")

.isEqualTo(txtLineMatrixConnected);

}

}

@Test

void graphReaderHtml() throws IOException {

try(StringReader reader = new StringReader(htmlLineConnected)) {

Assertions.assertThat(new GraphReaderWriterHtml().loadEdgeList(reader))

.as("Считанный список равен null!")

.isNotNull()

.as("Считанный список пуст!")

.isNotEmpty()

.as("Размер считанного списка не равен ожидаемому!")

.hasSize(4)

.as("Считанный список не равен ожидаемому списку!")

.isEqualTo(listFromFileConnected);

}

}

@Test

void writeGraphHtml() throws IOException {

try(StringWriter writer = new StringWriter()) {

new GraphReaderWriterHtml().saveIncidenceMatrix(writer, arrayFromFileConnected);

Assertions.assertThat(writer.toString())

.as("Записанная матрица пуста!")

.isNotEmpty()

.as("Размер записанной матрицы не совпадает с размером ожидаемой матрицы!")

.hasSize(htmlLineConnectedMatrixConnected.length())

.as("Записанная матрица не совпадает с ожидаемой!")

.isEqualTo(htmlLineConnectedMatrixConnected);

}

}

@Test

void graphReaderTxt() throws IOException {

try(StringReader reader = new StringReader(txtEdgesConnected)) {

Assertions.assertThat(new GraphReaderWriterTxt().loadEdgeList(reader))

.as("Считанный список равен null!")

.isNotNull()

.as("Считанный список пуст!")

.isNotEmpty()

.as("Размер считанного списка не равен ожидаемому!")

.hasSize(4)

.as("Считанный список не равен ожидаемому списку!")

.isEqualTo(listFromFileConnected);

}

}

@Test

void writeGraphTxt() throws IOException {

try(StringWriter writer = new StringWriter()) {

new GraphReaderWriterTxt().saveIncidenceMatrix(writer, arrayFromFileConnected);

Assertions.assertThat(writer.toString())

.as("Записанная матрица пуста!")

.isNotEmpty()

.as("Размер записанной матрицы не совпадает с размером ожидаемой матрицы!")

.hasSize(txtLineMatrixConnected.length())

.as("Записанная матрица не совпадает с ожидаемой!")

.isEqualTo(txtLineMatrixConnected);

}

}

@Test

void isConnected() throws IOException {

try(StringReader reader = new StringReader(htmlLineConnected); StringReader secondReader = new StringReader(txtEdgesConnectedSecond)) {

Graph graph = new Graph(new GraphReaderWriterHtml(), new GraphReaderWriterTxt());

graph.readFromFile(reader);

Assertions.assertThat(graph.isConnected())

.as("Граф не связный, ожидалось обратное!")

.isTrue();

Graph secondGraph = new Graph(new GraphReaderWriterTxt(), new GraphReaderWriterTxt());

secondGraph.readFromFile(secondReader);

Assertions.assertThat(secondGraph.isConnected()).isTrue();

}

}

@Test

void isNotConnected() throws IOException {

try(StringReader reader = new StringReader(htmlLineIsNotConnected)) {

Graph graph = new Graph(new GraphReaderWriterHtml(), new GraphReaderWriterTxt());

graph.readFromFile(reader);

Assertions.assertThat(graph.isConnected())

.as("Граф связный, ожидалось обратное!")

.isFalse();

}

}

@Test

void isDirected() {

Edge edgeDirected = new Edge(1, 2, true);

Edge edgeNotDirected = new Edge(1, 2, false);

Assertions.assertThat(edgeDirected.isDirected()).isTrue();

Assertions.assertThat(edgeNotDirected.isDirected()).isFalse();

}

@Test

void equalsEdge() {

Edge edgeDirected = new Edge(1, 2, true);

Edge edgeNotDirected = new Edge(1, 2, false);

Edge firstCopy = edgeDirected;

Object notEdge = null;

Assertions.assertThat(edgeDirected).isNotEqualTo(edgeNotDirected);

Assertions.assertThat(edgeDirected).isEqualTo(firstCopy);

Assertions.assertThat(edgeDirected).isNotEqualTo(notEdge);

}

}

package example;

import java.util.Objects;

public class Edge {

private int from;

private int to;

private boolean directed; // true = направленный, false = ненаправленный

public int getFrom() {

return from;

}

public void setFrom(int from) {

this.from = from;

}

public int getTo() {

return to;

}

public void setTo(int to) {

this.to = to;

}

public boolean isDirected() {

return directed;

}

public void setDirected(boolean directed) {

this.directed = directed;

}

public Edge(int from, int to, boolean directed) {

this.from = from;

this.to = to;

this.directed = directed;

}

public Edge() {

}

@Override

public boolean equals(Object object) {

if (this == object) return true;

if (object == null || getClass() != object.getClass()) return false;

Edge edge = (Edge) object;

return from == edge.from && to == edge.to && directed == edge.directed;

}

}

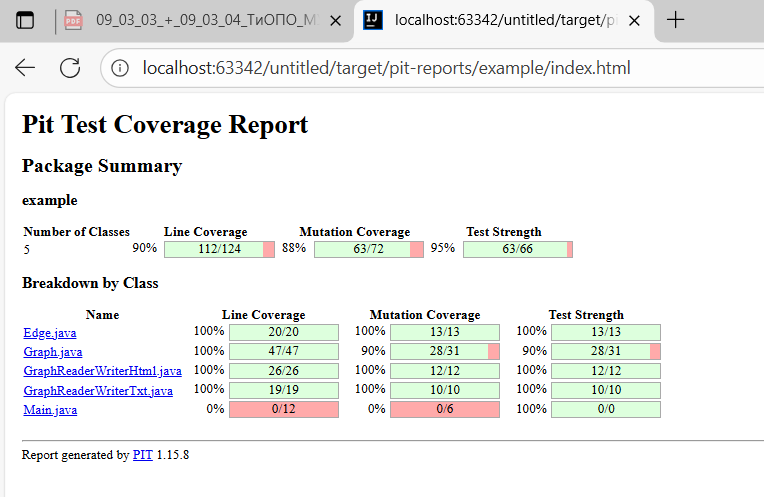


Рисунок 1 – Результат мутационного тестирования

**5 Вывод**

В ходе выполнения данной лабораторной работы удалось изучить принципы работы мутационного тестирования и использование инструмента Pitest для создания мутационных тестов.