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 №8.

Тема работы: «Тестирование с использованием тестовых двойников (Test Doubles)».

Наименование дисциплины: «Тестирование и отладка программного обеспечения».

Подготовил:

Студент группы 23-КБ-ПР1

Юданов М. Д.

Краснодар

2025

**1 Цель работы**

Цель работы – изучить подход к автоматизации процесса тестирования с использованием тестовых двойников (Test Doubles).

**2 Краткое теоретическое описание**

В тестировании сложных систем часто нужно изолировать код от внешних зависимостей – для этого применяются тестовые двойники. Они заменяют реальные объекты, чтобы упростить и контролировать выполнение тестов. Dummy – объект-заглушка, ничего не делает и используется только для передачи параметров. Stub – возвращает заранее подготовленные данные, обеспечивая предсказуемый вход. Spy – фиксирует вызовы и параметры, но не выполняет логику. Mock – проверяет, что методы вызваны с ожидаемыми параметрами, позволяет динамически менять поведение методов без повторного написания реализации интерфейса. Fake – упрощённая, но рабочая реализация интерфейса (например, in-memory БД вместо настоящей). Принципы инверсии управления (IoC) и внедрения зависимостей (DI) позволяют подставлять такие двойники вместо настоящих зависимостей, делая систему гибкой, изолированной и легко тестируемой.

**3 Задание**

1) Создать класс, реализующий граф/дерево, хранящий информацию о структуре графа, на основе некоторого внутреннего представления указанного в задании из пункта 5.1.

2) В созданном классе реализовать один из методов обработки графа/дерева в соответствии с вариантом задания из пункта 5.2.

3) В качестве источника данных использовать класс, реализующий чтение из файла произвольного формата и возвращающего данные в виде, указанном в варианте задания из пункта 5.1.

4) Реализовать возможность сохранения данных в файл произвольного формата, с помощью отдельного класса, в формате, определяемом в соответствии с вариантом задания в пункте 5.1.

5) Для обработки файла создать отдельный класс/классы, реализующий методы загрузки из файла (возвращающий считанные данных в определенном формате) и сохранения в файл (сохраняющий матрицу в определенном формате, который может отличаться от внутреннего представления графа).

6) Протестировать класс для работы с заданным форматом файлов (при тестировании класса использовать подмену класса на StringReader /StringWriter для доступа к строковому потоку с последующей проверкой поученной выходной строки).

7) Протестировать класс, реализующий непосредственные вычисления (при получении данных и сохранения использовать заглушки и фиктивные объекты).

8) Оформить отчёт.

Вариант 14. Входной формат – список рёбер, внутреннее   
представление – матрица инцидентности, выходной формат – матрица инцидентности. Необходимо реализовать алгоритм проверки графа на связность.

**4 Ход работы**

1-2) Создал класс, реализующий граф, хранящий информацию о нём в формате матрицы инцидентности и написал алгоритм обработки графа.

import java.io.IOException;

import java.io.Reader;

import java.io.Writer;

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.List;

import java.util.Queue;

public class Graph {

int[][] incidenceMatrix;

private final IGraphReader iGraphReader;

private final IGraphWriter iGraphWriter;

public Graph(IGraphReader iGraphReader, IGraphWriter iGraphWriter) {

this.iGraphReader = iGraphReader;

this.iGraphWriter = iGraphWriter;

}

public void readFromFile(Reader reader) throws IOException {

List<Edge> edges = iGraphReader.loadEdgeList(reader);

convertFromListEdgeToIncidenceMatrix(edges);

}

private void convertFromListEdgeToIncidenceMatrix(List<Edge> edges) {

int maxTop = Integer.MIN\_VALUE;

for (Edge edge: edges) {

if(edge.getFrom() > maxTop) maxTop = edge.getFrom();

if(edge.getTo() > maxTop) maxTop = edge.getTo();

}

incidenceMatrix = new int[maxTop][edges.size()];

for(int i = 0; i < edges.size(); i++) {

Edge currentEdge = edges.get(i);

for(int j = 0; j < maxTop; j++) {

if(j + 1 == currentEdge.getFrom()) {

incidenceMatrix[j][i] = currentEdge.isDirected() ? -1 : 1;

continue;

}

if(j + 1 == currentEdge.getTo()) {

incidenceMatrix[j][i] = 1;

}

}

}

}

public void writeToFile(Writer writer) throws IOException {

iGraphWriter.saveIncidenceMatrix(writer, incidenceMatrix);

}

public boolean isConnected() {

for(int i = 0; i < this.incidenceMatrix.length; i++) {

LinkedList<Integer> queue = new LinkedList<>();

queue.add(i);

boolean[] res = isTopConnected(this, new ArrayList<Integer>(), queue);

for(boolean visit: res) {

if(!visit) return false;

}

}

return true;

}

private static boolean[] isTopConnected(Graph graph, List<Integer> visited, Queue<Integer> connected) {

if(connected.isEmpty()) {

boolean[] res = new boolean[graph.incidenceMatrix.length];

for(int visit: visited) {

res[visit] = true;

}

return res;

}

int top = connected.poll();

for(int i = 0; i < graph.incidenceMatrix[0].length; i++) {

if(graph.incidenceMatrix[top][i] != 0) {

for(int j = 0; j < graph.incidenceMatrix.length; j++) {

if(top != j && graph.incidenceMatrix[j][i] == 1) {

connected.add(j);

break;

}

}

}

}

visited.add(top);

while (visited.contains(connected.peek())) connected.poll();

return isTopConnected(graph, visited, connected);

}

}

3-5) На шаге 1 в классе уже были предусмотрены возможности сохранения и считывания данных о графе в и из файлов произвольного формата. Были созданы интерфейсы IGraphReader и IGraphWriter и их реализации для .html и .txt.

import java.io.BufferedReader;

import java.io.IOException;

import java.io.Reader;

import java.io.Writer;

import java.util.ArrayList;

import java.util.List;

public interface IGraphReader {

List<Edge> loadEdgeList(Reader reader) throws IOException;

}

import java.io.IOException;

import java.io.Writer;

public interface IGraphWriter {

void saveIncidenceMatrix(Writer writer, int[][] incidenceMatrix) throws IOException;

}

import java.io.BufferedReader;

import java.io.IOException;

import java.io.Reader;

import java.io.Writer;

import java.util.ArrayList;

import java.util.List;

public class GraphReaderWriterHtml implements IGraphReader, IGraphWriter {

@Override

public List<Edge> loadEdgeList(Reader reader) throws IOException {

List<Edge> res = new ArrayList<>();

BufferedReader bufferedReader = new BufferedReader(reader);

String line;

while ((line = bufferedReader.readLine()) != null) {

if(!line.contains("h2")) continue;

line = line.substring(line.indexOf('>') + 1, line.lastIndexOf('<'));

String[] items = line.split(" ");

Edge currentEdge = new Edge();

currentEdge.setFrom(Integer.parseInt(items[0]));

currentEdge.setTo(Integer.parseInt(items[1]));

currentEdge.setDirected(Boolean.parseBoolean(items[2]));

res.add(currentEdge);

}

return res;

}

@Override

public void saveIncidenceMatrix(Writer writer, int[][] incidenceMatrix) throws IOException {

writer.append("<html><head></head><body>").append(System.lineSeparator());

for(int[] row: incidenceMatrix) {

writer.append("<h2>");

for (int i = 0; i < row.length; i++) {

writer.append(String.valueOf(row[i]));

if (i < row.length - 1) {

writer.append(' ');

}

}

writer.append("</h2>");

writer.append(System.lineSeparator());

}

writer.append("</body></html>");

writer.append(System.lineSeparator());

}

}

import java.io.BufferedReader;

import java.io.IOException;

import java.io.Reader;

import java.io.Writer;

import java.util.ArrayList;

import java.util.List;

public class GraphReaderWriterTxt implements IGraphWriter, IGraphReader {

@Override

public List<Edge> loadEdgeList(Reader reader) throws IOException {

List<Edge> res = new ArrayList<>();

BufferedReader bufferedReader = new BufferedReader(reader);

String line;

while ((line = bufferedReader.readLine()) != null) {

String[] items = line.split(" ");

Edge currentEdge = new Edge();

currentEdge.setFrom(Integer.parseInt(items[0]));

currentEdge.setTo(Integer.parseInt(items[1]));

currentEdge.setDirected(Boolean.parseBoolean(items[2]));

res.add(currentEdge);

}

return res;

}

@Override

public void saveIncidenceMatrix(Writer writer, int[][] incidenceMatrix) throws IOException {

for(int[] row: incidenceMatrix) {

for (int i = 0; i < row.length; i++) {

writer.append(String.format("%2d", row[i]));

if (i < row.length - 1) {

writer.append(' ');

}

}

writer.append(System.lineSeparator());

}

}

}

6-8) Провёл необходимые тесты с подменой объектов и оформил отчёт.

import org.junit.jupiter.api.Assertions;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Test;

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

public class GraphTest {

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 txtMatrixConnected =

"""

1 2 true

2 4 true

4 3 true

3 1 true

""";

int[][] arrayFromFileConnected = {

{-1,0,0,1},

{1,-1,0,0},

{0,0,1,-1},

{0,1,-1,0}

};

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.assertArrayEquals(arrayFromFileConnected, graph.incidenceMatrix);

}

}

@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.assertEquals(txtLineMatrixConnected, writer.toString());

}

}

@Test

void graphReaderHtml() throws IOException {

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

Assertions.assertEquals(listFromFileConnected, new GraphReaderWriterHtml().loadEdgeList(reader));

}

}

@Test

void writeGraphHtml() throws IOException {

try(StringWriter writer = new StringWriter()) {

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

Assertions.assertEquals(htmlLineConnectedMatrixConnected, writer.toString());

}

}

@Test

void graphReaderTxt() throws IOException {

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

Assertions.assertEquals(listFromFileConnected, new GraphReaderWriterTxt().loadEdgeList(reader));

}

}

@Test

void writeGraphTxt() throws IOException {

try(StringWriter writer = new StringWriter()) {

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

Assertions.assertEquals(txtLineMatrixConnected, writer.toString());

}

}

@Test

void isConnected() throws IOException {

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

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

graph.readFromFile(reader);

Assertions.assertTrue(graph.isConnected());

}

}

@Test

void isNotConnected() throws IOException {

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

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

graph.readFromFile(reader);

Assertions.assertFalse(graph.isConnected());

}

}

}

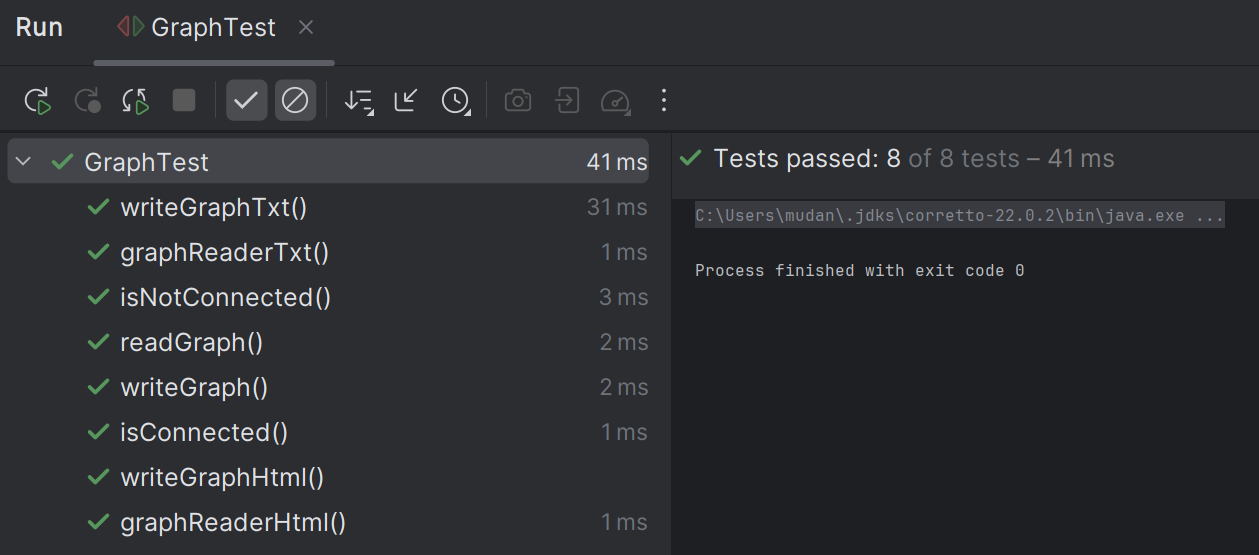


Рисунок 1 – Результат тестирования

**5 Вывод**

В ходе выполнения данной лабораторной работы удалось изучить подход к автоматизации процесса тестирования с использованием тестовых двойников (Test Doubles).