# 4 Лабораторная работа №4
# Исследование способов интеграционного тестирования программного обеспечения

## 4.1. Цель работы

Исследовать основные принципы интеграционного тестирования программного обеспечения. Приобрести практические навыки организации интеграционных тестов для объектно-ориентированных программ.

## 4.2. Постановка задачи

Был выдан вариант 14.

Выбрать в качестве тестируемого взаимодействие двух или более классов, спроектированных в лабораторных работах №1 – 4. Составить спецификацию тестового случая. Реализовать тестируемые классы. Выполнить тестирование с выводом результатов на экран и сохранением в log-файл. Проанализировать результаты тестирования, сделать выводы.

## 4.3. Ход выполнения работы

### 4.3.1. Спецификация тестового случая

В соответствии с задачей была составлена следующая спецификация:

- Названия взаимодействующих классов: Matrix; MatrixColumn.
- Название теста: test_replaceColumnData.
- Описание теста: тест проверяет метод replaceColumnWith — функцию замены данных указанного столбца в матрице. В тесте метод вызывается с двумя параметрами: 1 — номер заменяемого столбца и [0, 0, 0, 0] — массив новых данных

- Начальные условия: матрица $\begin{pmatrix} 2 & & \\ 1 & 2 & 3 \\ 1 & 4 & 5 \\ 2 & 78 & 11 \\ 1 & 100 & 5 \end{pmatrix}$.
- Ожидаемый результат: матрица $\begin{pmatrix} 0 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 78 & 11 \\ 0 & 100 & 5 \end{pmatrix}$.

### 4.3.2. Текст программы

На основе спецификации был создан тестовый драйвер MatrixIntegrateTest, реализующий следующие методы:

- В конструктор тестового драйвера передаются начальная матрица и список потоков, в которые будут записаны результаты каждого теста.
- Сами методы, реализующие тестовые случаи.
- Метод dump, выводящий текущее содержимое матрицы в поток консольного вывода или файловый. Формат вывода: матрица до изменения, команда, матрица после изменения.

Полный текст драйвера представлен в листинге 1.

Листинг 1 — Тестовый драйвер MatrixIntegrateTest

```
package org.cory7666.softwaretestingexample;
import java.io.PrintStream;
import java.util.List;
import org.cory7666.softwaretestingexample.task1.Matrix;

public class MatrixIntegrateTest implements AutoCloseable
{
  private final List<PrintStream> streams;
  private final Matrix input;
  public MatrixIntegrateTest (Matrix input, List<PrintStream> streams)
  { this.streams = streams; this.input = input; }
  public MatrixIntegrateTest test (int column, int[] newData)
  {
    writeString("-=== Начало теста. ===-");
    writeString("До изменения:"); dump();
    writeString(String.format("Замена данных в позиции %d.", column));
    input.replaceColumnWith(column, newData);
    writeString("После изменения:"); dump();
    writeString("-=== Конец теста. ===-\n");
    return this;
}
  private void dump ()
  { var matrixString = input.toString(); writeString(matrixString); }
  private void writeString (String x)
  { streams.forEach(stream -> stream.println(x)); }
  @Override
  public void close () throws Exception
  { streams.forEach(x -> x.close()); }
```

```
}
```

### 4.3.3. Тестовый запуск драйвера и анализ полученных данных

После написания кода программа была скомпилирована и запущена. Рисунок 4.1 демонстрирует вывод тестового драйвера.

```
❯ java -jar integrate-test.jar $(cat lab4-args)
-=== Начало теста. ===-
До изменения:
1       2       3
1       4       5
2       78      11
1       100     5
Замена данных в позиции 0.
После изменения:
0       2       3
0       4       5
0       78      11
0       100     5
-=== Конец теста. ===-
```

*Рисунок 4.1 — Результаты запуска тестового драйвера*

Полученные данных были проанализированы и сравнены со спецификацией. Результаты тестирования полностью соответствуют ожидаемым результатам.

## Выводы

При выполнении данной лабораторной работы были получены навыки составления интеграционных тестов. Было выделено, что интеграционные тесты следует применять для тестирования взаимодействия двух или нескольких модулей.

# ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

**1. Для чего применяется интеграционное тестирование программного обеспечения?**

Интеграционное тестирование – это тип тестирования, при котором программные модули объединяются логически и тестируются как группа. Как правило, программный продукт состоит из нескольких программных модулей, написанных разными программистами. Целью тестирования является выявление багов при взаимодействии между этими программными модулями и в первую очередь направлен на проверку обмена данными между этими самими модулями.

**2. Какие существуют типы взаимодействий объектов?**

Взаимодействия неявно предполагаются в спецификации класса, в которой установлены ссылки на другие объекты. Выявить такие взаимодействующие классы можно, используя отношения ассоциации, агрегирования и композиции, представленные на диаграмме классов. Связи такого рода преобразуются в интерфейсы класса, а тот или иной класс взаимодействует с другими классами посредством одного или нескольких способов:

1) Общедоступная операция имеет один или большее число формальных параметров объектного типа. Сообщение устанавливает ассоциацию между получателем и параметром, которая позволяет получателю взаимодействовать с этим параметрическим объектом.

2) Общедоступная операция возвращает значения объектного типа. На класс может быть возложена задача создания возвращаемого объекта, либо он может возвращать модифицированный параметр.

3) Метод одного класса создает экземпляр другого класса как часть своей реализации.

4) Метод одного класса ссылается на глобальный экземпляр некоторого другого класса. Разумеется, принципы хорошего тона в проектировании рекомендуют

минимальное использование глобальных объектов. Если реализация какого-либо класса ссылается на некоторый глобальный объект, рассматривайте его как неявный параметр в методах, которые на него ссылаются.

**3. Исходя из каких соображений выполняется выбор тестовых случаев при интеграционном тестировании?**

Исчерпывающее тестирование, другими словами, прогон каждого возможного тестового случая, покрывающего каждое сочетание значений – это, вне всяких сомнений, надежный подход к тестированию. Однако во многих ситуациях количество тестовых случаев достигает таких больших значений, что обычными методами с ними справиться попросту невозможно. Если имеется принципиальная возможность построения такого большого количества тестовых случаев, на построение и выполнение которых не хватит никакого времени, должен быть разработан систематический метод определения, какими из тестовых случаев следует воспользоваться. Если есть выбор, то мы отдаем предпочтение таким тестовым случаям, которые позволяют найти ошибки, в обнаружении которых мы заинтересованы больше всего.

Существуют различные способы определения, какое подмножество из множества всех возможных тестовых случаев следует выбирать. При любом подходе мы заинтересованы в том, чтобы систематически повышать уровень покрытия.

**4. Какие разделы включает спецификация тестового случая для интеграционного тестирования?**

- Идентификация - уникальный идентификатор тестового случая, который используется для управления тестами и отслеживания результатов.
- Описание - описание тестового случая, включая цель тестирования, действия пользователя и ожидаемый результат.

- Предусловия - начальные условия, которые необходимы для запуска теста, например, состояние программы или данные, на которых тестирование проводится.
- Шаги выполнения - последовательность шагов, которые необходимо выполнить для проведения тестирования, включая ввод данных, выполнение действий и проверку результатов.
- Ожидаемый результат - описание ожидаемых результатов после выполнения шагов тестирования.
- Фактический результат - результаты, полученные в результате выполнения теста, включая ошибки, предупреждения и любые другие наблюдения.
- Статус - результат выполнения теста, например, пройден, не пройден или отменен.
- Исполнитель - информация об исполнителе теста, кто проводил тестирование и когда оно было проведено.
- Тестовые данные - данные, которые использовались для проведения теста, включая входные данные и данные, полученные в результате выполнения теста.
- Среда тестирования - информация о среде, в которой проводилось тестирование, включая операционную систему, версии программного обеспечения и аппаратное обеспечение.
- Зависимости - список всех зависимостей, необходимых для запуска теста, включая другие компоненты программного обеспечения, базы данных и т.д.
- Комментарии - любые дополнительные комментарии, которые могут помочь понять результаты тестирования или описать особенности теста.

Спецификация тестового случая для интеграционного тестирования помогает определить, как различные компоненты программного обеспечения взаимодействуют друг с другом, и как они работают вместе для обеспечения целостности системы. Она также помогает документировать результаты тестирования, что может

быть полезно при поиске и исправлении ошибок и дефектов в программном обеспечении.