Модуль автоматической генерации тестовых сценариев для лабораторной работы по программированию

Пользовательская документация

Содержание

1 Введение

1.1 Сведения о документе

Данный документ описывает интерфейс и возможности программного модуля автоматической генерации тестовых сценариев для лабораторной работы по программированию (далее - модуль).

1.2 Назначение модуля

Модуль предназначен для проверки соответствия программного кода требованиям ЛР, а также для тестирования работоспособности реализованных команд.

Модуль предоставляет следующие возможности:

- Проверка корректности работы команд add, update, remove by id, clear, execute script, save;
- Проверка соответствия контракта
    - Наличие всех полей в классах,
    - Корректность типов;
- Генерация тестовых данных
    - Корректные данные,
    - Некорректные данные.

1.3 Основные понятия и критерии к написанию ЛР

Модуль представлен в виде пары файлов.

Для использования тестовых сценариев должны быть добавлены эти файлы в проект:

- pom.xml – библиотеки необходимые для работы с тестами,
- Java-модуль - Java-класс с тестами и сгенерированными данными.
    - Должен располагаться в папке test/java

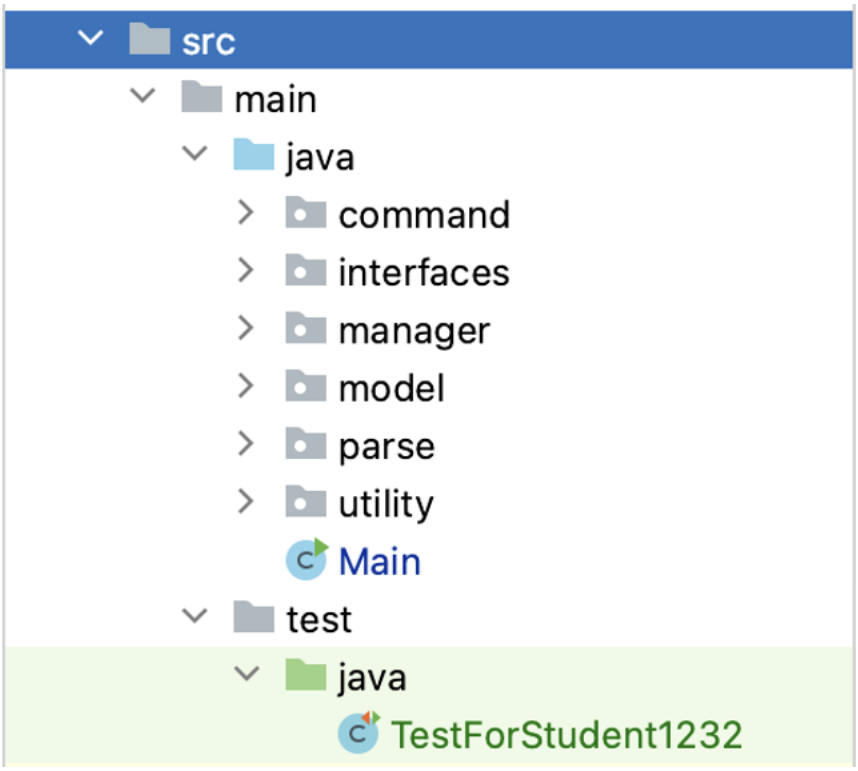

Рисунок 1 – Расположение файла с тестами

Запуск тестов осуществляется стандартно. При желании запустить все тесты, необходимо нажать на Run ‘TestForStudent1232’, иначе на конкретный тест.

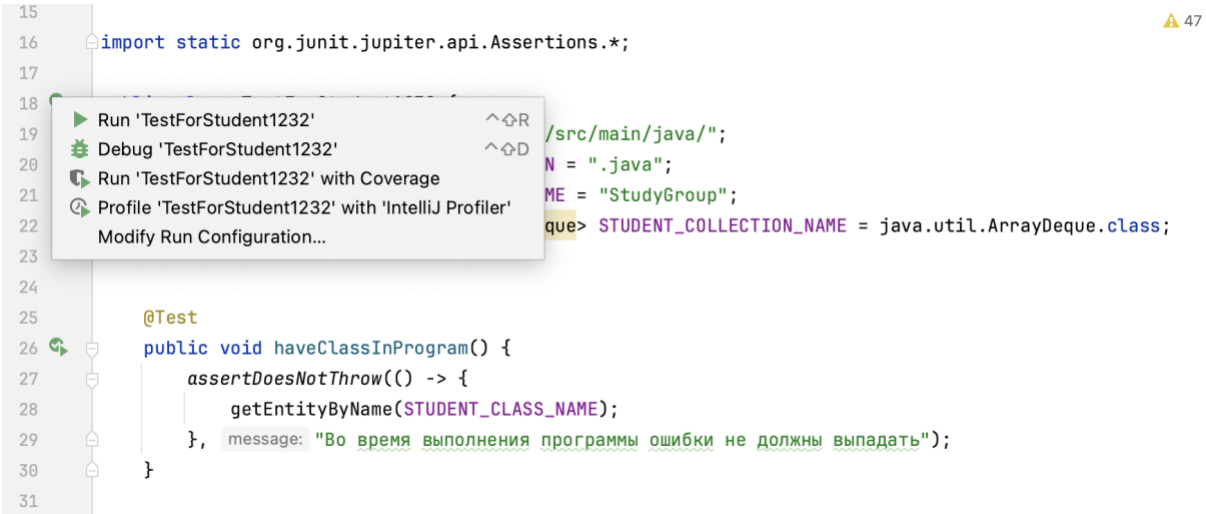

Рисунок 2 – Запуск всех тестов модуля

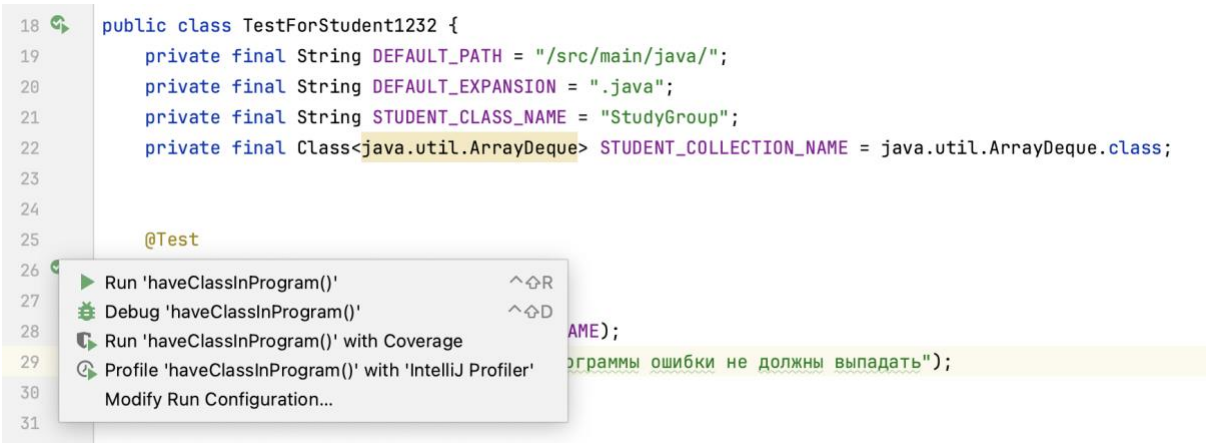

Рисунок 3 – Запуск конкретного теста модуля

К критериям для написания программного кода относится:

- Создание 1-го считывателя (например, Scanner) на все Java-приложение

Листинг 1 – Пример создания Сканера

```
import java.util.Scanner;

public class Reader {
    private final Scanner scanner = new Scanner(System.in);

    public String nextLine() {
        return scanner.nextLine();
    }
}
```

- Элементы коллекции должны храниться (сохраняться) в файле collection.txt
- Все сущности, с которыми осуществляются взаимодействия в рамках ЛР, должны находиться в пакете entity

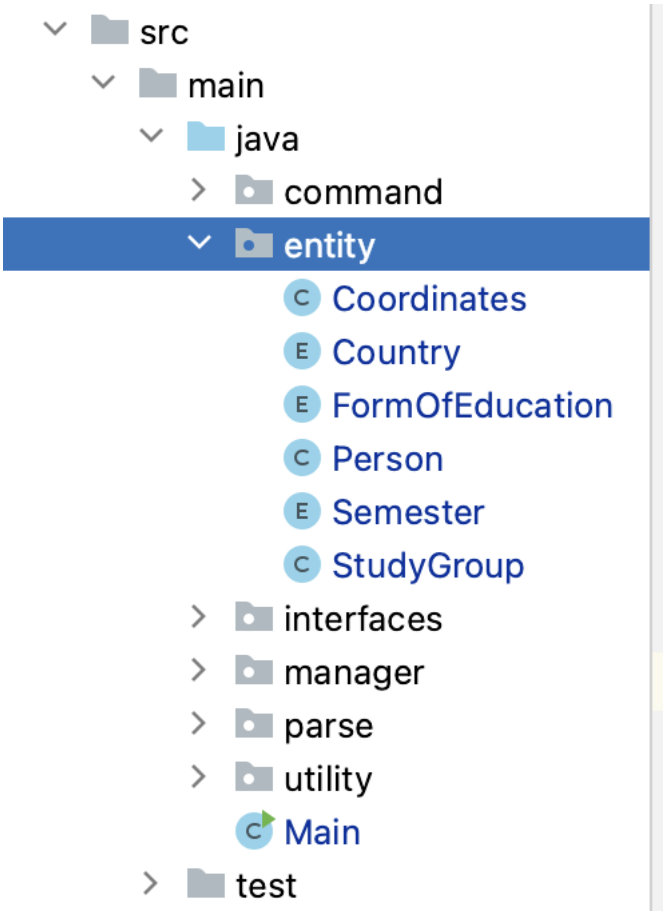

Рисунок 4 – Пример пакета entity

- Коллекция с необходимыми элементами должна быть статической (модификатор static)

Листинг 1 – Пример создания коллекции

```
private static final ArrayDeque<StudyGroup>
studyGroupCollection= new ArrayDeque<>();
```

1.4 Получение тестовых сценариев

- Перейти на сайт кафедры в раздел Программирование;
- Затем перейти к ЛР №5 и ввести вариант, указанный в журнале;
- После введения варианта отобразятся 2 кнопки, при нажатии на которые в буфер обмена будут копироваться либо сгенерированные файлы, либо файл с зависимостями pom.xml;

```
*
public enum Semester {
    FIRST,
    SECOND,
    SIXTH,
    SEVENTH,
    EIGHTH;
}
public enum Country {
    CHINA,
    INDIA,
    THAILAND,
    JAPAN;
}
```

Скопировать сгенерированные тесты к варианту

Скопировать Maven-файл, необходимый для запуска тестов

Отчёт по работе должен содержать:

1. Текст задания.
2. Диаграмма классов разработанной программы.
3. Исходный код программы.
4. Выводы по работе.

Рисунок 5 – Пример отображения сгенерированных данных

- Необходимо добавить оба файла к себе в проект;
- Можно приступать к написанию и тестированию программного кода.

1.5 Получение отчетов генерации

После запуска тестов, в консоли появляются результаты тестирования;

В случае, если тестирование прошло успешно, в консоли появятся шаги взаимодействия с командами, а также успешный результат Test passed: 37 of 37 tests

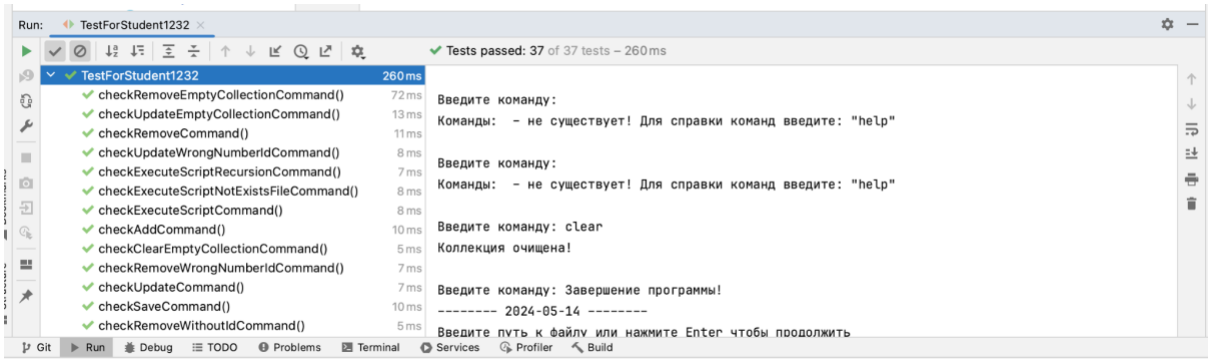

Рисунок 6 – Отчет при успешном прохождении тестов

В ином случае, в консоли появится описание причины падения теста. Например, если указать тип переменной, не соответствующий варианту ЛР, модуль сообщит об этом: “В классе - StudyGroup отсутствует поле или не указано название поля или тип. Названия поля должно быть - id. Тип поля должен соответствовать - Integer”, а также вернется неуспешный результат: Test failed: 1, passed: 36 of 37 tests

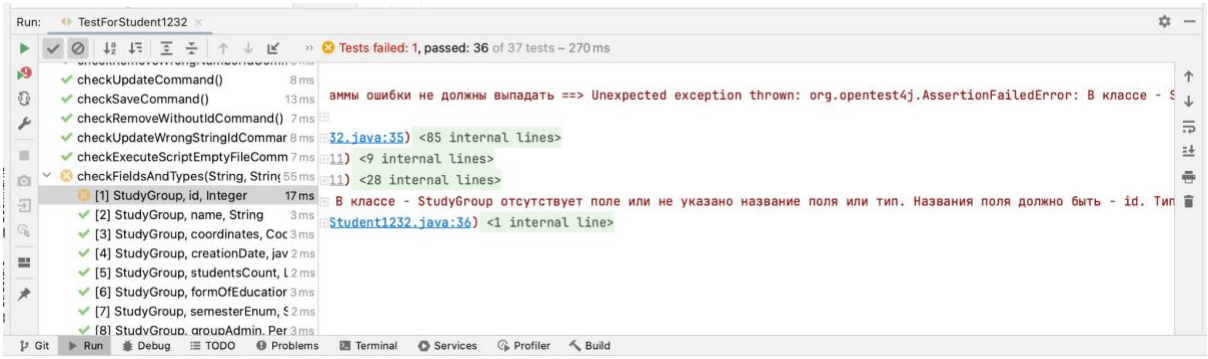

Рисунок 7 – Отчет при неуспешном прохождении тестов

Слева располагается история запусков тестов, можно нажать на тест, который не прошел и увидеть ошибку.

1.6 Критерии приемки ЛР

1. Обязательно должны быть пройдены все тесты;
2. Иначе, обратиться к преподавателю и объяснить причину, по которой тест не работает. Если преподаватель не сочтет проблему значимой, допускается принятие данного этапа.

1.7 Решение проблем с тестами

1. Обратиться к документации, внимательно перечитать пункты, проверить основные критерии к написанию ЛР;

2. Если проблема не решилась, следует обратиться к преподавателю практики;
3. В случае отсутствия возможности решить проблему, обратиться на почту Alina20021805@mail.ru с вопросом

1.8 Изменение тестовых сценариев

Изменение тестов не допускается. Преподаватель должен попросить студента сгенерировать новые тестовые данные на защите (повторно ввести вариант ЛР, добавить Java-файл с тестами в проект и запустить). Если программный код реализован корректно, любые тестовые данные к варианту должны вернуть одинаковый результат - успешно.