Университет ИТМО
Факультет ПИиКТ

Тестирование программного обеспечения

Лабораторная работа №1

Вариант 4452

Выполнила: Наумова Н.А.
Группа P33022
Преподаватель: Исаев И.В.

Санкт-Петербург
2021 г.

Цель работы

Познакомиться с модульным тестированием, изучить основные техники проведения тестирования, познакомиться с Java-фреймворком для организации юнит-тестирования JUnit 5.

Задание

1. Функция sec(x)
2. Программный модуль для работы с B деревьями (количество элементов в ключе - до 5, http://www.cs.usfca.edu/~galles/visualization/BTree.html)
3. Описание предметной области:

Форд продолжал считать вслух. Это одно из самых агрессивных действий, которые вы можете применить к компьютеру, равносильное тому, чтобы медленно приближаться в темноте к человеку, повторяя: "Умри... умри... умри...".
Наконец, Эдди тихо сказал:

-- Я вижу, нам придется поработать над нашими отношениями, -- и дверь открылась.

На них дунул ледяной ветер. Они, ежась, спустились по трапу на безжизненную поверхность Магратеи.

Вопросы к защите лабораторной работы:

1. Понятие тестирования ПО. Основные определения.
2. Цели тестирования. Классификация тестов.
3. Модульное тестирование. Понятие модуля.
4. V-образная модель. Статическое и динамическое тестирование.
5. Валидация и верификация. Тестирование методом "чёрного" и "белого" ящика.
6. Тестовый случай, тестовый сценарий и тестовое покрытие.
7. Анализ эквивалентности.
8. Таблицы решений и таблицы переходов.
9. Регрессионное тестирование.
10. Библиотека JUnit. Особенности API. Класс `junit.framework.Assert`.
11. Отличия JUnit 3 от JUnit 4.

Код

https://github.com/mmmlpmsw/software_testing_lab1

Вывод

1. Для разложения в ряд функции $y = sec(x)$ воспользуемся следующей формулой:

$sec(x) = \sum_{n=0}^{\infty} \frac{(-1)^n E_{2n}}{(2n)!} x^{2n}$ для всех $|x| < \frac{\pi}{2}$, где E_{2n} – числа Эйлера

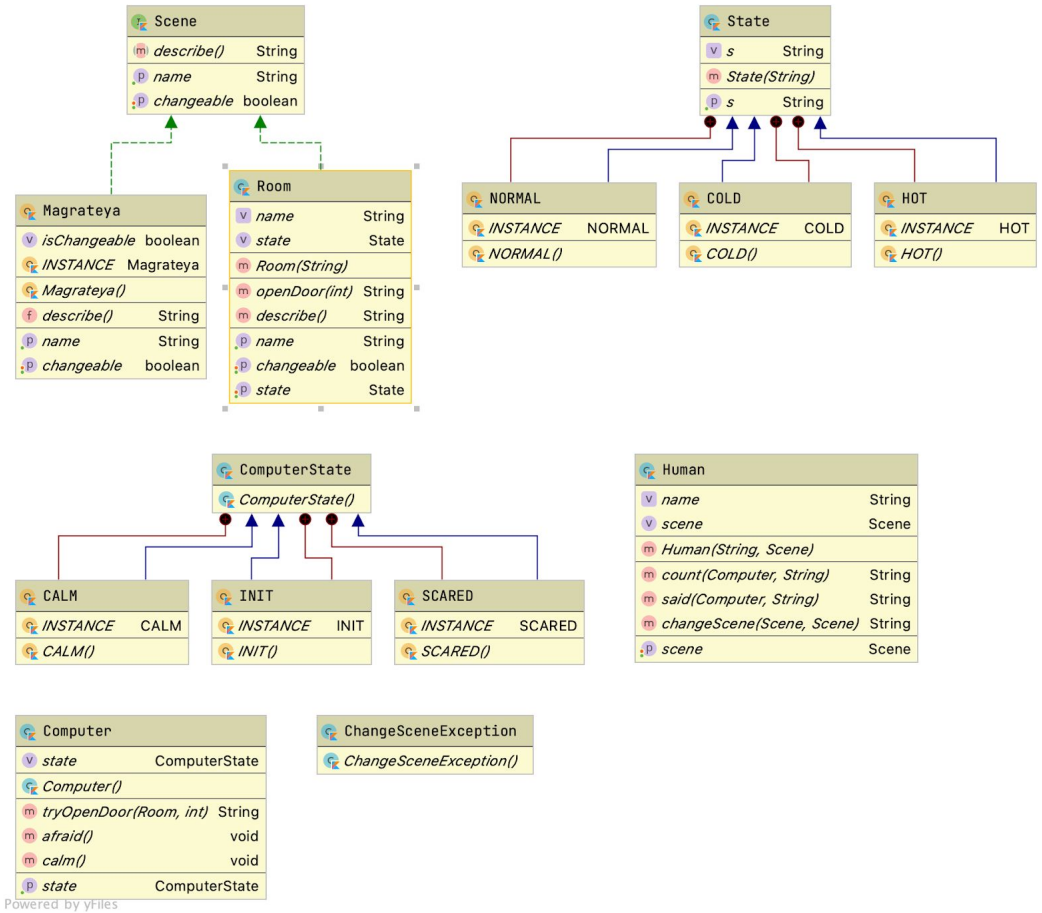
Затем выполним некоторые преобразования аргумента, чтобы функция могла считать значения на всей числовой прямой, а не только на интервале $(-\frac{\pi}{2}; \frac{\pi}{2})$.

Далее проведем тестирование данной функции. Для этого используем анализ эквивалентности: разбиваем функцию на классы эквивалентности (эквивалентного поведения в рамках одного класса), после чего из каждого такого “класса” берем по одному значению. Кроме того, отдельно проверяем граничные и недопустимые значения.

2. Был протестирован программный модуль для работы с B-деревом. Были выбраны характерные точки алгоритма и исходные наборы данных и выполнено сравнение с эталоном. Были протестированы различные случаи поведения: добавления, поиска, а также удаления - из корня, из листа, из промежуточного элемента, не оставлено также без внимания и просеивание элементов.

В данном случае было важно выделять характерные точки в алгоритме для тестирования и осуществить подбор нетривиальных входных данных для покрытия крайних точек алгоритма. Осуществить это помог в некоторой степени code coverage - он показал, какие методы остались не проверены и какая часть реализованного кода оказалось не покрыта тестами, в результате чего были подобраны новые тест-кейсы и усовершенствованы уже имеющиеся.

3. UML-диаграмма



Для тестирования доменной модели было разработано тестовое покрытие, покрывающее всю функциональность доменной модели. В данном случае доменная модель была представлена в виде конечного автомата, после чего выполнялась проверка переходов из одного состояния в другое.

В ходе выполнения работы был получен навык проведения модульного тестирования ПО.