|  |
| --- |
| Университет итмо |
| Лабораторная работа №1 по дисциплине «Тестирование программного обеспечения» |
| Вариант 213 |
| Группа Р3402 |
| **Выполнили:**  **Лалетина Екатерина Александровна**  **Орлова Кристина Александровна** |
| **Преподаватель: Харитонова Анастасия Евгеньевна** |

|  |
| --- |
| *10.12.19* |

Задание

1. Для указанной функции провести модульное тестирование разложения функции в степенной ряд. Выбрать достаточное тестовое покрытие.

2. Провести модульное тестирование указанного алгоритма. Для этого выбрать характерные точки внутри алгоритма, и для предложенных самостоятельно наборов исходных данных записать последовательность попадания в характерные точки. Сравнить последовательность попадания с эталонной.

3. Сформировать доменную модель для заданного текста. Разработать тестовое покрытие для данной доменной модели.

1. Функция sec(x)

2. Программный модуль для работы с хеш-таблицей с открытой адресацией (Hash Integer, http://www.cs.usfca.edu/~galles/visualization/ClosedHash.html)

3. Описание предметной области:

«Охранник обхватил их обоих за шеи и, почтительно поклонившись спине капитана, выволок с мостика, не обращая внимания на их сопротивление. Стальная дверь закрылась, и капитан снова остался один. Он задумчиво промурлыкал что-то и полистал свою записную книжку со стихами».

Функция

* Пример тестовых сценариев

|  |
| --- |
| @Test |
|  | public void test1() { |
|  | double x = Math.toRadians(-450); |
|  | double expected = Double.POSITIVE\_INFINITY; |
|  | double actual = Sec.count(x, EPS); |
|  |  |
|  | assertEquals(expected, actual, DELTA); |
|  | } |
|  |  |
|  | @Test |
|  | public void test2() { |
|  | double x = Math.toRadians(-360); |
|  | double expected = 1 / Math.cos(x); |
|  | double actual = Sec.count(x, EPS); |
|  |  |
|  | assertEquals(expected, actual, DELTA); |
|  | } |
|  |  |
|  | @Test |
|  | public void test3() { |
|  | double x = Math.toRadians(-270); |
|  | double expected = Double.POSITIVE\_INFINITY; |
|  | double actual = Sec.count(x, EPS); |
|  |  |
|  | assertEquals(expected, actual, DELTA); |
|  | } |

Алгоритм

* Пример тестовых сценариев

|  |
| --- |
|  |
| @Test |
|  | public void testSizeOfEmptyTable() { |
|  | assertEquals(0, hashTable.size()); |
|  | } |
|  |  |
|  | @Test |
|  | public void testEmptinessOfEmptyTable() { |
|  | assertEquals(true, hashTable.isEmpty()); |
|  | } |
|  |  |
|  | @Test |
|  | public void testContainedInEmptyTable() { |
|  | int value = (int) (Math.random() \* 100); |
|  | assertEquals(false, hashTable.contains(value)); |
|  | } |
|  |  |

Доменная модель

* Пример тестовых сценариев

|  |
| --- |
| @Test |
|  | public void testNullOnViolatersBeforeStart() { |
|  | Guard guard = new Guard(); |
|  | Violater violaters = guard.getViolaters(); |
|  |  |
|  | assertNull(violaters); |
|  | } |
|  |  |
|  | @Test |
|  | public void testParamsViolatersAfterStart() { |
|  | Guard guard = new Guard(); |
|  | guard.protectCaptain(); |
|  | Violater violaters = guard.getViolaters(); |
|  |  |
|  | assertEquals(violaters.getCount(), 2); // "обоих" |
|  | assertEquals(violaters.getPunishments(), 3); // всего предусмотрено 3 наказания |
|  | assertEquals(violaters.getPunished(), true); // уже наказаны |
|  | assertEquals(violaters.getViolate(), true); // уже виновны |
|  | assertEquals(violaters.getOpposition(), true); // оказывают сопротивление |
|  | } |
|  |  |
|  | @Test |
|  | public void testOtherPunishmentsNumOnViolaters() { |
|  | Guard guard = new Guard(); |
|  | guard.protectCaptain(); |
|  | Violater violaters = guard.getViolaters(); |
|  |  |
|  | violaters.setPunishments(4); // 4 > 3 |
|  | assertEquals(violaters.getPunished(), false); |
|  |  |
|  | violaters.setPunishments(2); // 2 < 3 |
|  | assertEquals(violaters.getPunished(), false); |
|  | } |

Выводы

В результате данной работы были повторены навыки написания кода для реализации работы математической функции; структуры данных; доменной модели. Было получено представление о том, как писать простейшие тесты, позволяющие протестировать функционал написанных систем.