Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Тестирование программного обеспечения»

Отчет

По лабораторной работе №1

Вариант 235725

Выполнили:
Нуцалханов Н. Г.
Грибов М. О.
P33101

Преподаватель:
Машина Е. А.

Санкт-Петербург, 2024 г.

Задание

Этапы

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

Вариант

1. Функция sec(x)
2. Программный модуль для поразрядной сортировки массива (http://www.cs.usfca.edu/~galles/visualization/RadixSort.html)
3. Описание предметной области:
   “*Форд отказался от попыток уснуть. В углу его каюты стоял маленький компьютер. Он посидел за ним немного, пытаясь сочинить новую статью о вогонах для "Путеводителя", но не смог выдумать ничего достаточно едкого и бросил. Он надел халат и решил сходить на мостик.*”

Выполнение

Исходный код:
https://github.com/nutsalhan87/itmo/tree/main/software_testing/lab1

Функция sec(x)

Тестовый класс выглядит так:

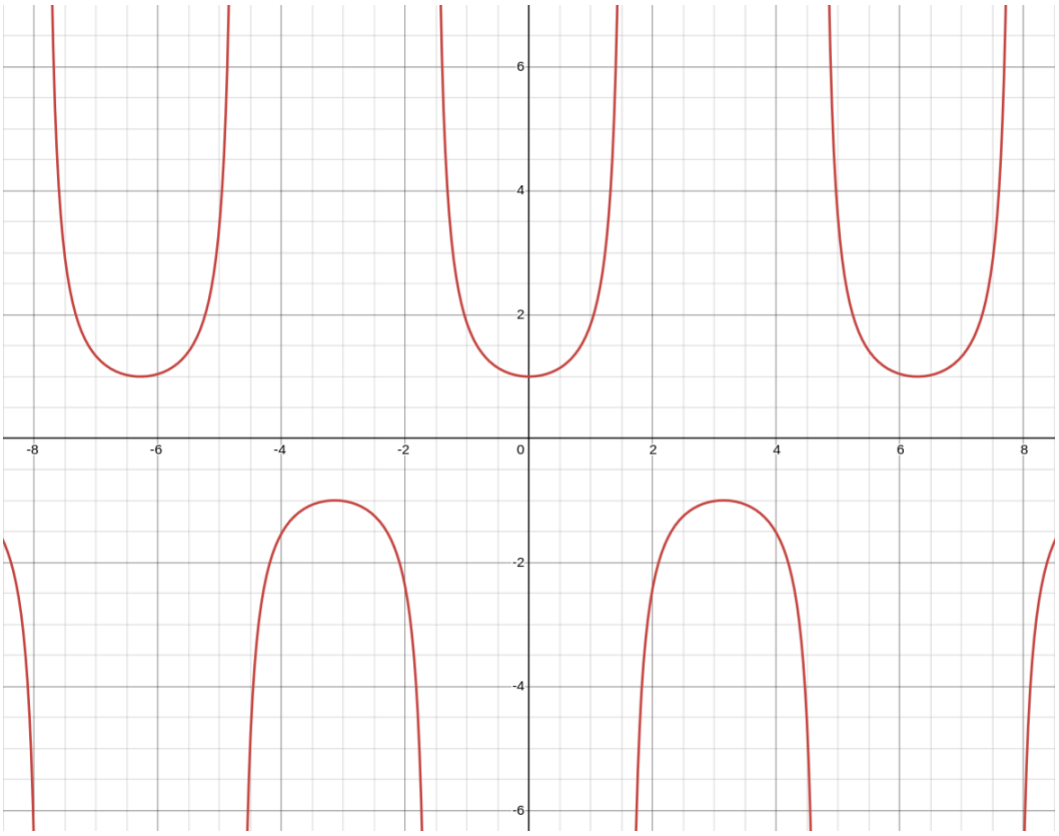
```
public class SecTest {
    private final double eps = 0.00001;

    @ParameterizedTest
    @ValueSource(doubles = {0, 2 * Math.PI, -2 * Math.PI, 4 * Math.PI, -4 * Math.PI})
    public void minExtremumBeyondPi(double x) {
        Assertions.assertEquals(1d, Sec.sec(x), eps);
    }

    @ParameterizedTest
    @ValueSource(doubles = {Math.PI, -Math.PI, 3 * Math.PI, -3 * Math.PI})
    public void maxExtremumBeyondPi(double x) {
        Assertions.assertEquals(-1d, Sec.sec(x), eps);
```

```
    }

    @ParameterizedTest
    @ValueSource(doubles = {0, 2 * Math.PI, -2 * Math.PI, 4 * Math.PI, -4 * Math.PI})
    public void nearPiDiv2ModifiedPositive(double x) {
        Assertions.assertEquals(100_000, Sec.sec(x + Math.PI / 2 - 0.00001), eps);
        Assertions.assertEquals(100_000, Sec.sec(x - Math.PI / 2 + 0.00001), eps);
    }

    @ParameterizedTest
    @ValueSource(doubles = {Math.PI, Math.PI, 3 * Math.PI, -3 * Math.PI})
    public void nearPiDiv2ModifiedNegative(double x) {
        Assertions.assertEquals(-100_000, Sec.sec(x + Math.PI / 2 - 0.00001), eps);
        Assertions.assertEquals(-100_000, Sec.sec(x - Math.PI / 2 + 0.00001), eps);
    }

    @ParameterizedTest
    @ValueSource(doubles = {Math.PI / 2, 3 * Math.PI / 2, -Math.PI / 2, -3 * Math.PI / 2})
    public void piDiv2(double x) {
        Assertions.assertEquals(Double.NaN, Sec.sec(x), eps);
    }
}
```

Первые пара тестов проверяет в экстремумах функции — это значения 1 и -1.

Вторая пара тестов проверяет значения около разрывов.

Пятый тест проверяет значения в точках разрыва функции.



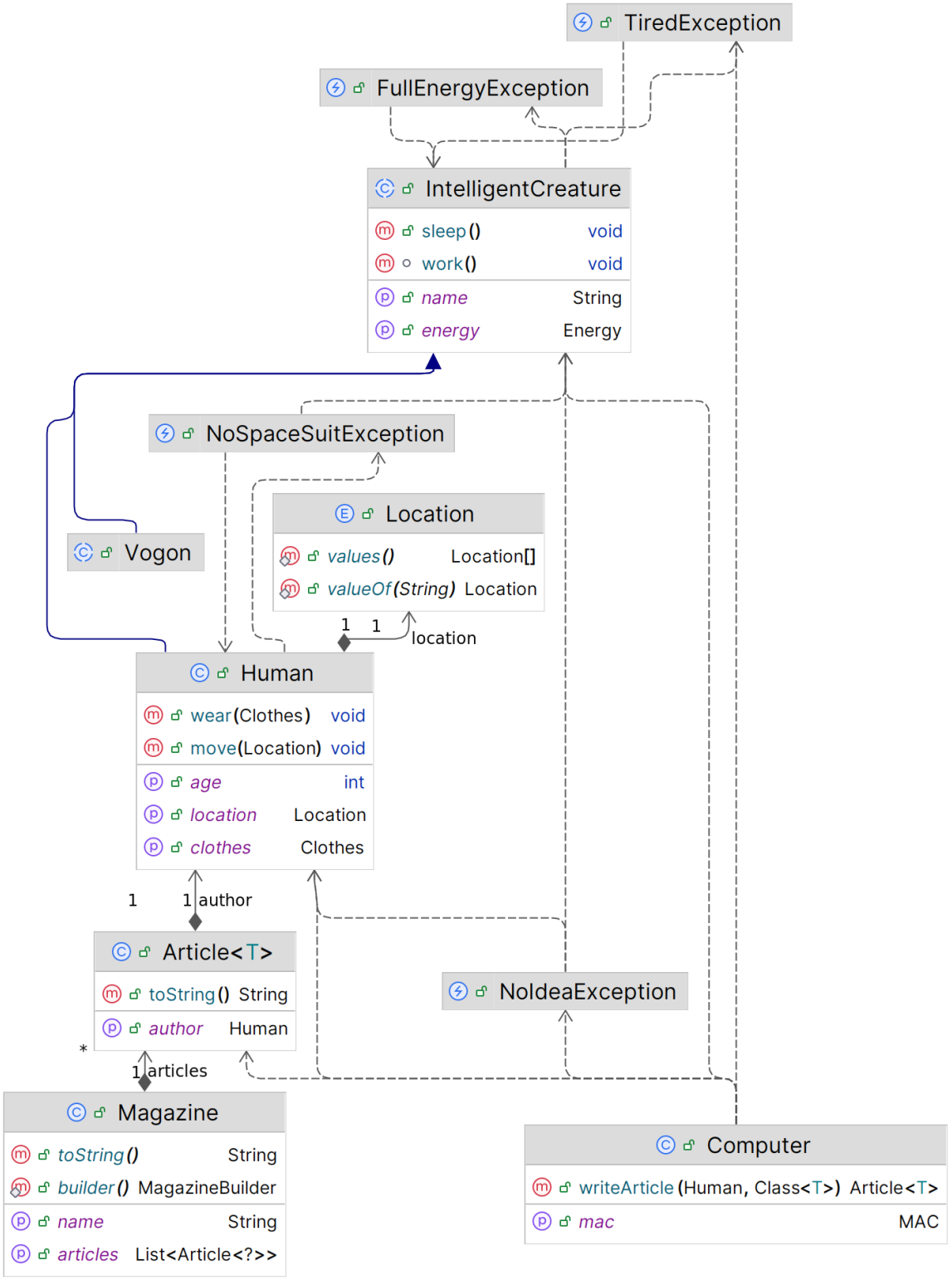
Поразрядная сортировка

Тестовый класс выглядит так:

```
public class RadixSortTest {
   @Test
   void empty() {
       int[] source = {};
       Assertions.assertArrayEquals(source, RadixSort.sort(source));
   }

   @Test
   void oneElement() {
       int[] source = {5};
       Assertions.assertArrayEquals(source, RadixSort.sort(source));
   }

   @Test
   void multipleElements() {
       int[] source = {906, 204, 868, 40, 463, 68, 515, 960, 913, 362, 338, 104, 31, 505,
986, 737, 97, 0, 549};
       int[] expected = Arrays.stream(source).sorted().toArray();
       Assertions.assertArrayEquals(expected, RadixSort.sort(source));
   }
}
```

В ходе тестов банально проверяются три случая: пустой массив, массив из одного элемента и массив из нескольких элементов.

Доменная модель



Тестовый класс выглядит так:

```
public class DomainTest {
   @Test
   void scenario() {
       var ford = new Human("Ford", 33, Location.CREW_QUARTERS);
       var computer = new Computer(Computer.MAC.random());
       Assertions.assertThrows(FullEnergyException.class, ford::sleep);
       Assertions.assertThrows(NoIdeaException.class, () -> computer.writeArticle(ford,
Vogon.class));
       ford.wear(Human.Clothes.BATHROBE);
       Assertions.assertEquals(Human.Clothes.BATHROBE, ford.getClothes());

       Assertions.assertAll(() -> ford.move(Location.BRIDGE));
   }

   @Test
   void mac() {
       Assertions.assertEquals("2A-00-F3-16-CD-01",
Computer.MAC.fromLong(0x2a00f316cd01L).toString());
   }

   @ParameterizedTest
   @ValueSource(classes = {Human.class, Location.class, Magazine.class, Computer.class,
Article.class})
   void article(Class c){
       var ford = new Human("Ford", 33, Location.CREW_QUARTERS);
       var computer = new Computer(Computer.MAC.random());
       Assertions.assertEquals(c.getSimpleName(), computer.writeArticle(ford,
c).toString());
       var stanford = new Human("Stanford", 33, Location.CREW_QUARTERS);
       Assertions.assertEquals(stanford, computer.writeArticle(stanford, c).getAuthor());
   }

   @Test
   void spacesuit(){
       var ford = new Human("Ford", 33, Location.CREW_QUARTERS);
       Assertions.assertThrows(NoSpaceSuitException.class, () ->
ford.move(Location.EXTERIOR));
       ford.wear(Human.Clothes.SPACE_SUIT);
       Assertions.assertAll(()->ford.move(Location.EXTERIOR));
   }

   @Test
   void energy(){
       var ford = new Human("Ford", 33, Location.CREW_QUARTERS);
       var computer = new Computer(Computer.MAC.random());
       Assertions.assertThrows(FullEnergyException.class, ford::sleep);
       Assertions.assertEquals(ford.getEnergy(), IntelligentCreature.Energy.FULL);
       computer.writeArticle(ford, Human.class);
       Assertions.assertEquals(ford.getEnergy(), IntelligentCreature.Energy.LOW);
       Assertions.assertThrows(TiredException.class, ()->computer.writeArticle(ford,
Magazine.class));
       Assertions.assertAll(ford::sleep);
       Assertions.assertEquals(ford.getEnergy(), IntelligentCreature.Energy.FULL);
   }

   @Test
   void human_creation(){
       Assertions.assertThrows(IllegalArgumentException.class, () -> new Human("Toddler",
-1, Location.BRIDGE));

       var ford = new Human("Ford", 33, Location.EXTERIOR);
       Assertions.assertEquals(Human.Clothes.SPACE_SUIT, ford.getClothes());
   }
}
```

Тестовый метод *scenario* проверяет работоспособность сценария из задания. Остальные же методы проверяют работоспособность отдельных частей доменной модели.