Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1
«Модульное тестирование»

по дисциплине «Тестирование программного обеспечения»

Вариант: 413732

Студент:
Анкудинов Кирилл Константинович
Группа:
P3318
Преподаватель:
Пименов Дмитрий Анатольевич

Санкт-Петербург, 2026

Оглавление

1. Методика тестирования

Лабораторная работа состоит из трёх заданий, для каждого из которых применялось модульное тестирование методом «белого ящика» с использованием фреймворка **JUnit 5**. Сборка и тестирование проекта выполняется с помощью **Gradle**. Тестируемый код написан на **Java 17**.

Для каждого задания тесты запускались командой:

```
./gradlew test        # All tests
./gradlew :task1:test # Tests of task 1
./gradlew :task2:test # Tests of task 2
./gradlew :task3:test # Tests of task 3
```

2. Задание 1. Разложение $\cos(x)$ в степенной ряд

2.1. Описание реализации

Функция $\cos(x)$ вычисляется через ряд Тейлора:

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - ...$$

Перед вычислением аргумент приводится к диапазону $[-\pi, \pi]$. Суммирование продолжается до тех пор, пока абсолютное значение очередного слагаемого не станет меньше `Double.MIN_VALUE`, либо не будет достигнуто максимальное число итераций.

Особые случаи обрабатываются явно: при $x = \mathrm{NaN}$ или $x = \pm\infty$ метод возвращает `NaN`.

2.2. Тестовые случаи

Тесты разделены на две группы:

1. **Граничные значения** (`"Corner values"`) — 13 тест-кейсов:
2. **Диапазон** $[-\pi, \pi]$ (`"[-PI; PI] range"`) — 16 тест-кейсов из файла `values.csv`:

Для обеих групп используется `assertEquals(...)` с `TOLERANCE = 1e-5`.

2.3. Результаты

```
> Task :task1:test
cos(x) to Tailor series > Corner values > cos(-9999.9) PASSED
cos(x) to Tailor series > Corner values > cos(-3.141592653589793)
PASSED
cos(x) to Tailor series > Corner values > cos(-3.141592653589793)
PASSED
cos(x) to Tailor series > Corner values > cos(-1.0E-8) PASSED
cos(x) to Tailor series > Corner values > cos(-0.0) PASSED
cos(x) to Tailor series > Corner values > cos(0.0) PASSED
cos(x) to Tailor series > Corner values > cos(1.0E-8) PASSED
cos(x) to Tailor series > Corner values > cos(3.141592653589793)
PASSED
cos(x) to Tailor series > Corner values > cos(3.141592653589793)
PASSED
cos(x) to Tailor series > Corner values > cos(9999.9) PASSED
cos(x) to Tailor series > Corner values > cos(NaN) PASSED
cos(x) to Tailor series > Corner values > cos(Infinity) PASSED
cos(x) to Tailor series > Corner values > cos(-Infinity) PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(-3.14159265) =
```

```
-1.00000000 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(-2.72271363) =
-0.91354546 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(-2.30383461) =
-0.66913061 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(-1.88495559) =
-0.30901699 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(-1.46607657) =
0.10452846 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(-1.04719755) =
0.50000000 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(-0.62831853) =
0.80901699 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(-0.20943951) =
0.97814760 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(0.20943951) =
0.97814760 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(0.62831853) =
0.80901699 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(1.04719755) =
0.50000000 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(1.46607657) =
0.10452846 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(1.88495559) =
-0.30901699 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(2.30383461) =
-0.66913061 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(2.72271363) =
-0.91354546 PASSED
cos(x) to Tailor series > [-PI; PI] range > cos(3.14159265) =
-1.00000000 PASSED
```

Все 29 тестов пройдены успешно. Неудачных тестов нет.

3. Задание 2. Биномиальная куча

3.1. Описание реализации

Биномиальная куча (Binomial Heap) — структура данных, представляющая собой набор биномиальных деревьев, упорядоченных по степени. Реализованы операции: `insert`, `getMin`, `extractMin`. Слияние куч (`merge`) является вспомогательной операцией, используемой внутри остальных.

Для сохранения трэйса выполнения, в реализацию добавлено поле класса `TraversalTracer`. Характерные точки выделены в `enum TracePoint`:

Точка	**Место вызова**
`INSERT`	Начало метода `insert`
`MERGE`	Начало метода `merge`
`MERGE_LISTS`	Слияние двух корневых списков по степени в `mergeLists`
`CONSOLIDATE`	Устранение деревьев одинаковой степени в `consolidate`
`EXTRACT_MIN`	Начало метода `extractMin`
`GET_MIN`	Начало метода `getMin`
`REVERSE`	Разворот списка дочерних узлов в `reverseList`

3.2. Тестовые случаи

Сценарий	**Ожидаемый результат**
Вставка в пустую кучу	`INSERT → MERGE`
Вставка второго элемента (слияние двух T_0)	`INSERT → MERGE → MERGE_LISTS`
Вставка третьего элемента (слияние T_1 и T_0)	`INSERT → MERGE → MERGE_LISTS → CONSOLIDATE`
`getMin` при одном элементе	`GET_MIN`
`extractMin` при одном элементе	`EXTRACT_MIN`
`extractMin` при двух элементах (с разворотом детей)	`EXTRACT_MIN → REVERSE → MERGE`
Корректность порядка извлечения (5 элементов)	Проверка значений, без трэйса
`getMin` на пустой куче	`NoSuchElementException`
`extractMin` на пустой куче	`NoSuchElementException`

3.3. Результаты

```
> Task :task2:test
Binomial Heap > extractMinon heap with one element with reverse
PASSED
Binomial Heap > Correctness of operations - several inserts and
extractMins PASSED
Binomial Heap > Merge of two T0 PASSED
Binomial Heap > Exception when extractMin on empty heap PASSED
Binomial Heap > getMin on heap with one element PASSED
Binomial Heap > extractMin on heap with one element PASSED
Binomial Heap > Exception when getMin on empty heap PASSED
Binomial Heap > Merge of T1 and T0 PASSED
Binomial Heap > Insert into empty heap PASSED
```

Все 9 тестов пройдены успешно. Неудачных тестов нет.

4. Задание 3. Доменная модель

4.1. Исходный текст

«Они сидели на мостовой и смотрели с некоторым беспокойством, как огромные дети тяжело прыгают по песку, а дикие лошади с грохотом везут по небу в Неизведанные Области свежие запасы армированных изгородей.»

4.2. Доменная модель

Из текста выделены следующие сущности и их отношения:

Класс / тип	Ключевые атрибуты
`Person`	`location`, `emotion`
`Child`	`size`, `location`
`Horse`	`wild: boolean`, `location`
`Supply`	`fences: List<ReinforcedFence>`
`ReinforcedFence`	`freshness: Freshness`, `reinforced = true`
`Location`	`getName(): String`
`Pavement`	impl. `Location`
`Sand`	impl. `Location`
`Sky`	impl. `Location`
`UnknownRegions`	impl. `Location`
`Size`	`HUGE`, `NORMAL`
`Freshness`	`FRESH`, `STALE`

`Emotion`	`CALM`, `SOMEWHAT_WORRIED`, `WORRIED`
`Observable`	Маркерный интерфейс

4.3. Тестовые случаи

Проверяют, что сущности, их поведение и свойства соответствуют тому, что описано в исходном текста

Тест	**Что проверяется**
`theyAreOnThePavement`	Оба человека находятся на мостовой
`pavementIsCalledPavement`	Название локации «мостовая»
`watchingHugeChildCausesSomeWorry`	Наблюдение за ребёнком → `SOMEWHAT_WORRIED`
`worryIsSomeNotExtreme`	Состояние не равно `WORRIED` (не полное беспокойство)
`childrenAreHuge`	Размер ребёнка — `HUGE`
`childrenAreNotNormalSized`	Размер не равен `NORMAL`
`childrenJumpOnSand`	Ребёнок прыгает на песок
`sandIsCalledSand`	Название локации «песок»
`childHasNoLocationBeforeJumping`	Новый ребёнок не имеет местоположения
`horsesAreWild`	Лошадь является дикой
`horsesArriveAtUnknownRegions`	Лошадь прибывает в Неизведанные Области
`unknownRegionsIsCalledCorrectly`	Название «Неизведанные Области»
`routeGoesthroughTheSky`	Название маршрута «небо»
`horseHasNoDestinationBeforePulling`	Новая лошадь не имеет местоположения
`suppliesAreFresh`	Запас из свежих изгородей — свежий
`supplyWithOneStaleFenceIsNotFresh`	Запас с одной несвежей изгородью — не свежий
`allFencesInSupplyAreReinforced`	Все изгороди в запасе армированы
`fenceIsAlwaysReinforced...`	Армированность не зависит от свежести

4.4. Результаты

```
> Task :task3:test
AppTest > horseHasNoDestinationBeforePulling() PASSED
AppTest > sandIsCalledSand() PASSED
AppTest > horsesAreWild() PASSED
```

```
AppTest > worryIsSomeNotExtreme() PASSED
AppTest > childrenAreNotNormalSized() PASSED
AppTest > childrenAreHuge() PASSED
AppTest > fenceIsAlwaysReinforcedRegardlessOfFreshness() PASSED
AppTest > pavementIsCalledPavement() PASSED
AppTest > childrenJumpOnSand() PASSED
AppTest > supplyWithOneStaleFenceIsNotFresh() PASSED
AppTest > watchingHugeChildCausesSomeWorry() PASSED
AppTest > horsesArriveAtUnknownRegions() PASSED
AppTest > routeGoesthroughTheSky() PASSED
AppTest > unknownRegionsIsCalledCorrectly() PASSED
AppTest > childHasNoLocationBeforeJumping() PASSED
AppTest > theyAreOnThePavement() PASSED
AppTest > allFencesInSupplyAreReinforced() PASSED
AppTest > suppliesAreFresh() PASSED
```

Все 18 тестов пройдены успешно. Неудачных тестов нет.

5. Заключение

В ходе лабораторной работы было реализовано и протестировано три программных модуля. Итоговая статистика:

Подпроект	**Всего**	**Пройдено**	**Ошибок**	**Пропущено**
task1 ($\cos(x)$)	29	29	0	0
task2 (Binomial Heap)	9	9	0	0
task3 (Domain model)	18	18	0	0
Итого	**56**	**56**	**0**	**0**

Все 56 тестов пройдены успешно, сбоев нет.