

Отчет по лабораторной работе №7 предмета ООП

Выполнил Брусенцев Степан Эдуардович
Студент группы 6204-010302D

Задание на лабораторную работу:

Внести изменения в существующий набор типов табулированных функций, позволяющие обрабатывать точки функций по порядку (паттерн «Итератор»), а также выбирать тип объекта табулированной функции при его неявном создании (паттерн «Фабричный метод» и средства рефлексии).

Задание 1

В интерфейс TabulatedFunction добавлено наследование от параметризованного интерфейса Iterable<FunctionPoint>.

В класс ArrayTabulatedFunction добавлен метод iterator(), возвращающий анонимный класс итератора:

Итератор работает напрямую с внутренним массивом functionPoints

Метод next() возвращает копию точки (new FunctionPoint(...)), защищая инкапсуляцию

Метод remove() выбрасывает UnsupportedOperationException

При отсутствии следующего элемента выбрасывается

NoSuchElementException

В класс LinkedListTabulatedFunction добавлен аналогичный метод iterator():

Итератор работает с внутренней структурой связанного списка

Хранит ссылку на текущий узел для эффективного перехода

Также возвращает копии точек и соблюдает те же исключения

В методе main() добавлена проверка работы через for-each цикл для обоих типов функций, демонстрирующая корректность итерации по всем точкам.

Задание 2

Создан интерфейс TabulatedFunctionFactory с тремя фабричными методами createTabulatedFunction().

В классы ArrayTabulatedFunction и LinkedListTabulatedFunction добавлены публичные вложенные классы-фабрики, реализующие интерфейс.

В TabulatedFunctions добавлено статическое поле-фабрика и метод setTabulatedFunctionFactory() для его замены.

Добавлены три фабричных метода createTabulatedFunction() в TabulatedFunctions, которые делегируют создание текущей фабрике.

Все прямые создания табулированных функций в TabulatedFunctions

заменены на вызовы фабричных методов.

Задание 3

Добавлены три перегруженных метода `createTabulatedFunction()` в класс `TabulatedFunctions`, использующие рефлексию:

Методы принимают параметр `Class<? extends TabulatedFunction>` для указания создаваемого класса.

Через рефлексию находится нужный конструктор и создаётся экземпляр.

Исключения рефлексии преобразуются в `IllegalArgumentException`.

В `main()` проверена работа рефлексивного создания объектов.

Все задания лабораторной работы 7 были выполнены.

==== ТЕСТ ИТЕРАТОРОВ ===

1. Тестирование `ArrayTabulatedFunction`:

Исходная функция ($\sin(x)$ на $[0, \pi]$ с 5 точками):

(0.0; 0.0)

(0.7853981633974483; 0.7071067811865475)

(1.5707963267948966; 1.0)

(2.356194490192345; 0.7071067811865476)

(3.141592653589793; 1.2246467991473532E-16)

2. Тестирование `LinkedListTabulatedFunction`:

Исходная функция ($\cos(x)$ на $[0, \pi]$ с 5 точками):

(0.0; 1.0)

(0.7853981633974483; 0.7071067811865476)

(1.5707963267948966; 6.123233995736766E-17)

(2.356194490192345; -0.7071067811865475)

(3.141592653589793; -1.0)

Тест завершен успешно!

==== ТЕСТ ФАБРИК ===

По умолчанию (ArrayTabulatedFunction): ArrayTabulatedFunction

После установки LinkedList фабрики: LinkedListTabulatedFunction

После установки Array фабрики: ArrayTabulatedFunction

Тест завершен успешно!

==== ТЕСТ РЕФЛЕКСИИ ===

```
class functions.ArrayTabulatedFunction
{(0.0; 0.0), (5.0; 0.0), (10.0; 0.0)}
class functions.ArrayTabulatedFunction
{(0.0; 0.0), (10.0; 10.0)}
class functions.LinkedListTabulatedFunction
{(0.0; 0.0), (10.0; 10.0)}
class functions.LinkedListTabulatedFunction
{(0.0; 0.0), (0.3141592653589793; 0.3090169943749474),
(0.6283185307179586; 0.5877852522924731), (0.9424777960769379;
0.8090169943749475), (1.2566370614359172; 0.9510565162951535),
(1.5707963267948966; 1.0), (1.8849555921538759; 0.9510565162951536),
(2.199114857512855; 0.8090169943749475), (2.5132741228718345;
0.5877852522924732), (2.827433388230814; 0.3090169943749475),
(3.141592653589793; 1.2246467991473532E-16)}
```

Тест завершен успешно!