МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева» (Самарский университет)

Институт информатики и кибернетики

Кафедра информационных систем и технологий

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

по курсу Объектно-ориентированное программирование

Выполнил студент:

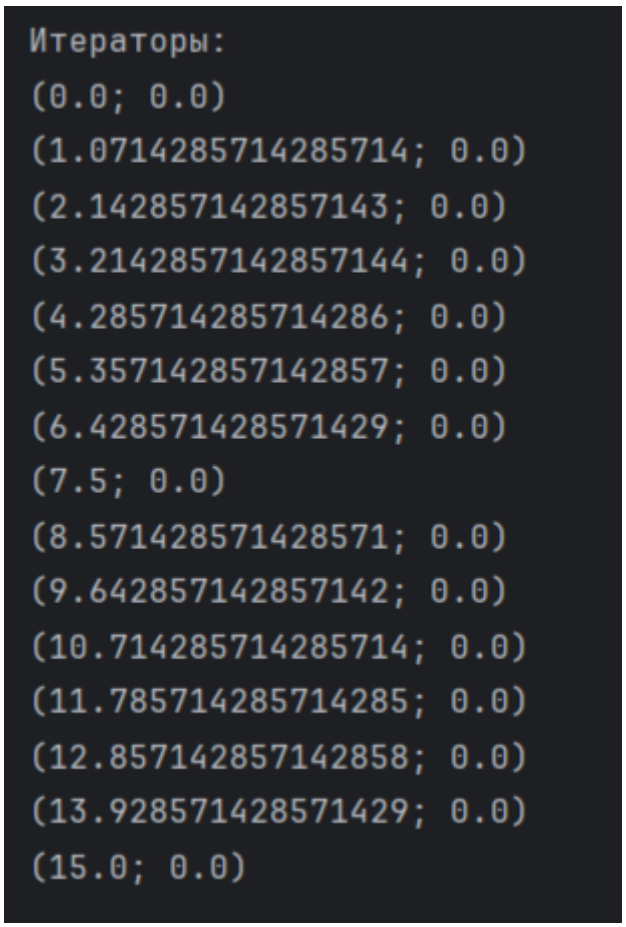
Мацюк А.А.

гр. 6203-010302D

Самара 2025

Задание №1

В рамках первого задания был расширен интерфейс TabulatedFunction путем наследования от Iterable<FunctionPoint>. Это позволяет использовать объекты табулированных функций в циклах for-each.
Для двух классов — ArrayTabulatedFunction и LinkedListTabulatedFunction — были разработаны анонимные классы-итераторы, которые работают непосредственно с внутренними структурами данных:
- В ArrayTabulatedFunction итератор использует индексацию массива точек.
- В LinkedListTabulatedFunction итератор осуществляет обход узлов связного списка.
Оба итератора возвращают копии точек, что обеспечивает сохранение инкапсуляции данных. Метод remove() в итераторах не поддерживается и вызывает исключение UnsupportedOperationException.



Задание №2

Во втором задании был введен интерфейс TabulatedFunctionFactory с тремя перегруженными методами createTabulatedFunction(), соответствующими конструкторам табулированных функций.
Для каждого класса табулированных функций создан вложенный класс-фабрика, реализующий этот интерфейс:
- ArrayTabulatedFunctionFactory
- LinkedListTabulatedFunctionFactory

В классе TabulatedFunctions было объявлено статическое поле типа TabulatedFunctionFactory, инициализированное фабрикой для ArrayTabulatedFunction. Также был добавлен

метод setTabulatedFunctionFactory() для динамической смены фабрики во время выполнения программы.

```
Фабрика:
class functions.ArrayTabulatedFunction
class functions.LinkedListTabulatedFunction
class functions.ArrayTabulatedFunction
```

Задание №3

Третье задание было посвящено применению механизма рефлексии в Java. В классе TabulatedFunctions были добавлены перегруженные
методы createTabulatedFunction(), которые:
Получают конструктор нужного класса с
помощью getConstructor() и создают экземпляр объекта через newInstance().
Также были реализованы методы для чтения табулированных функций из потоков с указанием целевого класса:
- inputTabulatedFunction(Class, InputStream)
- readTabulatedFunction(Class, Reader)
В случае возникновения исключений при работе с рефлексией они преобразуются в IllegalArgumentException.

```
Рефлексия:
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; 0.9510565162951
```