

Лабораторная работа №2

Выполнила: Качкуркина Арина Валерьевна

6204-010302D

2025 г.

Задание 1

Я создала пакет functions для дальнейшего размещения классов программы.

Задание 2

В пакете functions я создала класс FunctionPoint для описания точек табулированной функции. Класс включает:

- приватные поля *x* и *y* для координат точки
- геттеры и сеттеры;
- три конструктора:
 - *FunctionPoint(double x, double y)* - точка с заданными координатами
 - *FunctionPoint(FunctionPoint point)* - копия существующей точки
 - *FunctionPoint()* – точка с координатами (0, 0)

При написании соблюдала принципы инкапсуляции - поля приватные, доступ через методы.

Задание 3

В пакете functions создала класс TabulatedFunction для работы с табулированными функциями. Я реализовала:

- массив *FunctionPoint[] points* для хранения точек;
- *pointsCount* для отслеживания количества точек;
- два конструктора:
 - ~*TabulatedFunction(double leftX, double rightX, int pointsCount)* - создает точки с *y = 0*;

**TabulatedFunction(double leftX, double rightX, double[] values)* - создает точки с заданными x (получает значения f -ий в виде массива);

Точки создаются через равные интервалы по x , сохраняя упорядоченность.

Задание 4

Добавила в класс TabulatedFunction методы для работы с функцией:

- Метод *double getLeftDomainBorder()* - возвращает левую границу области определения;
- Метод *double getRightDomainBorder()* - возвращает правую границу области определения ;
- Метод *double getFunctionValue(double x)* - вычисляет значение функции в точке x (если она лежит в области определения), иначе возвращает значение неопределенности (поле NaN класса Double).

Метод getFunctionValue использует линейную интерполяцию между точками.

Задание 5

Реализовала в классе TabulatedFunction методы для работы с точками функции:

- Метод *int getPointsCount()* – возвращает количество точек;
- Метод *FunctionPoint getPoint(int index)* - возвращает копию точки по индексу;
- Метод *void setPoint(int index, FunctionPoint point)* - заменяет точку с проверкой порядка по x ;

- Метод *double getPointX(int index), setPointX(int index, double x)* - работа с координатой x с указанным номером;
- Метод *double getPointY(int index), setPointY(int index, double y)* - работа с координатой y.

Для корректного сравнения чисел с плавающей точкой введена константа EPSILON = 1e-10. Она используется в методах setPoint() и setPointX() для проверки порядка точек по X, а также в addPoint() для сохранения упорядоченности массива. Это предотвращает ошибки сравнения.

Методы setPoint и setPointX проверяют, чтобы не нарушался порядок точек по x.

Задание 6

В классе TabulatedFunction добавила методы изменения количества точек:

- *deletePoint(int index)* - удаляет заданную точку по индексу;
- *addPoint(FunctionPoint point)* - добавляет новую точку таб.ф-ии;
- из рекомендаций использовала arraycopy() для копирования участков массивов.

При добавлении точки сохраняется упорядоченность по x. Массив увеличивается с запасом (+5) только при необходимости.

Задание 7

Проверяю работу написанных классов.

Создала класс Main вне пакета functions для тестирования:

- создала простую линейную функцию $f(x) = x$ с точками (0.0,0.0), (1.0,1.0), (2.0,2.0), (3.0,3.0), (4.0,4.0);

- проверила значение ф-й;
- вывела значения функции в точках внутри и вне области определения;
- проверила линейную интерполяцию в промежуточных точках;
- протестировала добавление и удаление точек;
- после каждой операции добавила вывод полного массива через гетеры, чтобы были видны изменения;
- убедилась, что порядок точек сохраняется после изменений.

Программа успешно вывела результаты до и после модификации точек, подтвердив корректность работы всех методов.

РЕЗУЛЬТАТ ПОСЛЕ ВЫПОЛНЕНИЯ РАБОТЫ:

Исходная функция

Левая граница: 0.0

Правая граница: 4.0

Кол-во точек: 5

Точки функции: [(0.0, 0.0), (1.0, 1.0), (2.0, 2.0), (3.0, 3.0), (4.0, 4.0)]

Проверка значений функции

$f(-1.0) = \text{NaN}$

$f(0.5) = 0.5$

$f(1.7) = 1.7$

$f(2.2) = 2.2$

$f(3.4) = 3.4$

$f(5.5) = \text{NaN}$

Проверка операций с точками

1. Добавляем точку (2.2, 2.2):

Точки функции: [(0.0, 0.0), (1.0, 1.0), (2.0, 2.0), (2.2, 2.2), (3.0, 3.0), (4.0, 4.0)]

$$f(2.2) = 2.2$$

2. Удаляем точку с индексом 1:

Точки функции: [(0.0, 0.0), (2.0, 2.0), (2.2, 2.2), (3.0, 3.0), (4.0, 4.0)]

3. Изменяем y точки с индексом 2 на 2.5:

Точки функции: [(0.0, 0.0), (2.0, 2.0), (2.2, 2.5), (3.0, 3.0), (4.0, 4.0)]

Итоговый результат

$$f(2.2) = 2.5$$

Точки функции: [(0.0, 0.0), (2.0, 2.0), (2.2, 2.5), (3.0, 3.0), (4.0, 4.0)]