Вкладка 1

Отчет о лабораторной работе №2

«Разработка классов для работы с табулированными функциями»

Выполнил:
Петров Евгений Станиславович
Группа: 6204-010302D

Вкладка 2

Постановка задачи

Разработать пакет **`functions`** с классами **`FunctionPoint`** и **`TabulatedFunction`**.

Требования:

- Обеспечить хранение точек табулированной функции **без использования `java.util`**.
- Реализовать операции:
    - получения значения функции;
    - доступа к точкам;
    - изменения координат точек;
    - изменения количества точек.
- Продемонстрировать работу классов в `Main`.

Теория и используемые подходы

- Табулированная функция хранится как массив точек `(x, y)`, **упорядоченных по возрастанию `x`**.
- Значения внутри области определения вычисляются **линейной интерполяцией** между соседними точками.
- Добавление и удаление точек выполняется через **`System.arraycopy`**.

- Для минимизации перераспределений массив хранится **с запасом ёмкости**.

Реализация

Класс `FunctionPoint`

- Инкапсулирует координаты точки.
- Имеет три конструктора:
  - по координатам;
  - копирующий;
  - по умолчанию.
- Предоставляет стандартные **геттеры и сеттеры**.

Класс `TabulatedFunction`

Поля:

- `FunctionPoint[] points` — массив точек
- `int size` — текущее число точек
- константы `EPSILON`, `DEFAULT_CAPACITY`

Конструкторы:

1. `TabulatedFunction(double leftX, double rightX, int pointsCount)`

— создаёт равномерную сетку с нулевыми значениями.

2. `TabulatedFunction(double leftX, double rightX, double[] values)`
   — задаёт значения из массива.

Ключевые методы:

- `getLeftDomainBorder()` / `getRightDomainBorder()`
- `getFunctionValue(double x)`
  — внутри диапазона использует интерполяцию, вне возвращает `Double.NaN`
- `getPointsCount`, `getPoint`, `setPoint`
- `get/setPointX`, `get/setPointY`
- `deletePoint(int index)`
  — запрещено удалять точку, если останется < 2
- `addPoint(FunctionPoint point)`
  — вставляет точку по `x`, запрещает дубликаты

Внутренние утилиты:

- `ensureCapacity`
- `ensureCorrectOrder`
- `checkIndex`

Класс `Main`

- Создаёт табулированную функцию синуса на диапазоне `[0; π]`
- Считывает и изменяет значения
- Добавляет и удаляет точки
- Выводит результаты, включая точки вне области определения

Тестирование

Компиляция:

```
javac src\functions\FunctionPoint.java
src\functions\TabulatedFunction.java src\Main.java
```

Запуск:

```
java -cp src Main
```

Проверка производится:

- ручным анализом вывода,
- обработкой исключений,
- набором сценариев.

Сценарии проверки

1. **Базовая инициализация**
   Создание сетки `[0; π]` (6 точек). Проверяется шаг и

корректность крайних точек.

2. **Интерполяция внутри диапазона**
   Сравнение `getFunctionValue(π/6)`, `π/3`, `π/2` с `Math.sin(x)`
   Отклонение ≤ `EPSILON`.

3. **Добавление точки**
   Вставка `x = 0.75π`. Проверка сортировки и расширения массива.

4. **Удаление точки + обработка ошибок**
   `deletePoint(0)` уменьшает размер.
   При попытке удалить при двух оставшихся — `IllegalStateException`.

5. **Запрос вне области**
   `getFunctionValue(-0.5)` и `getFunctionValue(π+0.5)` возвращают `NaN`.

6. **Дубликаты `x`**
   Добавление точки с существующим `x` вызывает `IllegalArgumentException`.

Фрагмент консольного вывода

Initial sine approximation
f(-0,5000) = NaN
f(0,0000) = 0,000000
f(0,5236) = 0,489821
f(1,0472) = 0,829966
f(1,5708) = 0,951057
f(3,1416) = 0,000000

f(3,6416) = NaN

After editing points
f(0,2000) = NaN
f(1,0472) = 1,029966
f(1,5708) = 1,101057
f(2,3562) = 0,707107
f(3,3916) = NaN

Выводы

- Созданы классы для хранения и обработки табулированной функции **без стандартных коллекций**.
- Реализованы операции доступа, изменения и структурных преобразований точек.
- Продемонстрирована работа на примере функции синуса.
- Все требования задания выполнены.
