

Лабораторная работа №2

Автор: Ежова Екатерина

Группа: 6203-010302D

Задание 2

В пакете `functions` создаем публичный класс `FunctionPoint`, в нем у нас есть два `private` поля, которые хранят значения `x` и `y`.

Первый конструктор `FunctionPoint(double x, double y)` создаёт объект точки с заданными координатами, в конструкторе используем `this` и присваиваем значения `x` и `y`.

Второй конструктор `FunctionPoint(FunctionPoint point)` создаёт объект точки с теми же координатами, что у указанной точки.

Третий конструктор `FunctionPoint()` создаёт точку с координатами `(0; 0)`. Также в этот класс я добавила методы `getX()` и `getY()`, которые возвращают значения `x` и `y`. Они нужны для того, чтобы получить доступ к значениям `private` полей.

Задание 3

В пакете `functions` создаем публичный класс `TabulatedFunction`, в нем у нас есть два приватных поля: массив типа `FunctionPoint` `arrayPoints`, который хранит данные о точках табулированной функции и длина этого массива `arrayLen`. В классе я реализовала 2 конструктора.

Конструктор `TabulatedFunction(double leftX, double rightX, int pointsCount)` создаёт объект табулированной функции по заданным левой и правой границе области определения, а также количеству точек для табулирования. В нем я создаю переменную `intervalLength` для определения интервала между точками x . С помощью цикла `for` инициализирую массив `arrayPoints`. Для каждой точки вызываем конструктор класса `FunctionPoint` и передаем в него значение $x = \text{leftX}$ и $y=0$, далее смещаем левую границу на интервал. Если точка является последней, то присваиваем x значение правой границы.

Второй конструктор `TabulatedFunction(double leftX, double rightX, double[] values)` вместо количества точек получает значения функции в виде массива. Находим длину массива с помощью метода `length`, аналогично заполняем массив только вместо $y=0$, передаем в конструктор значение y из массива `values`.

Задание 4

Методы `getLeftDomainBorder()` и `getRightDomainBorder()` возвращают значение левой и правой границы функции. В первом методе мы обращаемся к первой точке с индексом 0 и применяем метод `getX()`, который возвращает значение x для этой точки. Аналогично второй метод, только берем индекс последней точки.

Метод `getFunctionValue(double x)` возвращает значение функции в точке x . В начале проверяем принадлежит ли точка исходному интервалу, если нет возвращаем NaN. Если точка совпадает с левой или правой границей возвращаем значение функции для данной точки x . Находим интервал, которому принадлежит точка x , то есть с помощью цикла сравниваем нашу точку с уже заданными значениями x . Переменная `ind` будет хранить индекс точки X_2 , которая следует после x . С помощью этого индекса и метода `getPointX()` находим значения X_1 и X_2 – точек между, которыми лежит исходная точка, а также находим значения функции в этих точках - Y_1 и Y_2 . Возвращаем значение функции в исходной точке с помощью уравнения прямой, проходящей через заданные 2 точки.

Задание 5

Метод `getPointsCount()` возвращает длину массива `arrayLen`.

Метод `getPoint(int index)` возвращает копию точки. С помощью оператора `new` создаем новую точку, в конструктор передадим значения координат исходной точки, а их получим, из массива точек по заданному индексу и методов `getX()` и `getY()`.

Методы `getPointX(int index)` и `getPointY(int index)` возвращают значение `x` и `y` по индексу соответственно. Берем точку по индексу из массива `arrayPoints` и используем методы `getX()` и `getY()` для получения значений.

Метод `setPoint(int index, FunctionPoint point)` заменяет точку на новую. Находим значения абсцисс `leftPoint` и `rightPoint`, находящихся слева и справа от начальной точки. Если точка первая или последняя, то присваиваем значения для `leftPoint` или `rightPoint` такое же как у новой точки, так как в таком случае вставка новой точки на эти места не нарушает порядок `x`. Далее если точка, входит в интервал, то есть она находится между соседними точками `leftPoint` и `rightPoint`, меняем ее значение в массиве.

Метод `setPointX(int index, double x)` меняет значение абсциссы точки с указанным номером. В этом методе мы вызываем метод `setPoint()`, в который передаем индекс и новую точку с таким же `y`, но с новым `x`. Аналогично метод `setPointY()` только меняем уже `y`.

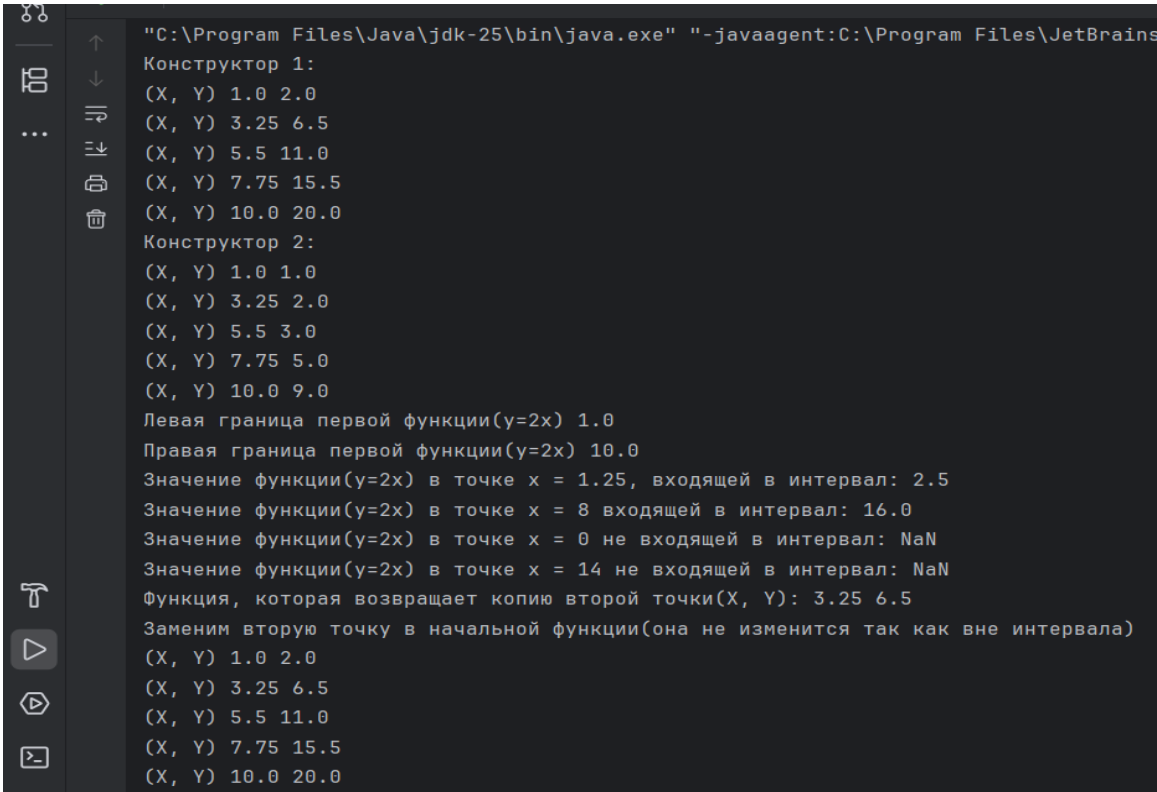
Задание 6

Метод `deletePoint(int index)` удаляет точку по указанному индексу. Если индекс принадлежит интервалу, то с помощью цикла сдвигаем все элементы влево. Затем уменьшаем длину массива.

Метод `addPoint(FunctionPoint point)` добавляет точку в массив. С помощью метода `getX()` находим абсциссу X . С помощью цикла находим индекс, где будет стоять новая точка, если X больше правой границы, `ind` будет равен длине массива. Далее создаем новый массив `arrayPoints1` с длиной на единицу больше. С помощью метода `arraycopy()` копируем в новый массив все элементы, стоящие до нового, добавляем в массив новый элемент, и копируем остальные, присваиваем исходному массиву значение нового и увеличиваем длину.

Задание 7

Импортируем классы `FunctionPoint` и `TabulatedFunction`. В классе `Main` в методе `main()` создаем экземпляр класса `TabulatedFunction` `fun` на интервале (1 10), для каждого `x` устанавливаем значение $y=2x$. С помощью цикла и метода `setPointY` устанавливаем новые значения `y`. Далее выводим результат. Потом проверяем работу второго конструктора, где создаем массив значений функции `val`, для каждого `x` устанавливаем значение из массива и выводим результат. Далее в классе я проверяла работу различных методов и выводила результат.



```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains
Конструктор 1:
(X, Y) 1.0 2.0
(X, Y) 3.25 6.5
(X, Y) 5.5 11.0
(X, Y) 7.75 15.5
(X, Y) 10.0 20.0
Конструктор 2:
(X, Y) 1.0 1.0
(X, Y) 3.25 2.0
(X, Y) 5.5 3.0
(X, Y) 7.75 5.0
(X, Y) 10.0 9.0
Левая граница первой функции(y=2x) 1.0
Правая граница первой функции(y=2x) 10.0
Значение функции(y=2x) в точке x = 1.25, входящей в интервал: 2.5
Значение функции(y=2x) в точке x = 8 входящей в интервал: 16.0
Значение функции(y=2x) в точке x = 0 не входящей в интервал: NaN
Значение функции(y=2x) в точке x = 14 не входящей в интервал: NaN
Функция, которая возвращает копию второй точки(X, Y): 3.25 6.5
Заменим вторую точку в начальной функции(она не изменится так как вне интервала)
(X, Y) 1.0 2.0
(X, Y) 3.25 6.5
(X, Y) 5.5 11.0
(X, Y) 7.75 15.5
(X, Y) 10.0 20.0
```

(X, Y) 10.0 20.0

Заменяем вторую точку в начальной функции

(X, Y) 1.0 2.0

(X, Y) 2.0 7.0

(X, Y) 5.5 11.0

(X, Y) 7.75 15.5

(X, Y) 10.0 20.0

Заменяем у третьей точки абсциссу

(X, Y) 1.0 2.0

(X, Y) 2.0 7.0

(X, Y) 6.6 11.0

(X, Y) 7.75 15.5

(X, Y) 10.0 20.0

Удалим третью точку

(X, Y) 1.0 2.0

(X, Y) 2.0 7.0

(X, Y) 7.75 15.5

(X, Y) 10.0 20.0

Добавим три новых точки

(X, Y) 0.0 0.0

(X, Y) 1.0 2.0

(X, Y) 2.0 7.0

(X, Y) 4.0 8.0

(X, Y) 7.75 15.5

(X, Y) 10.0 20.0

(X, Y) 11.0 9.0