ОТЧЕТ ПО
ЛАБОРАТОРНОЙ РАБОТЕ №2

Выполнил:
Батуров Максим Дмитриевич
Группа: 6203-010302D

Оглавление

Задание №1

Я создал пакет functions. В этом пакете будут размещаться все классы, разработанные в рамках данной программы.

Задание №2

В пакете functions я разработал класс FunctionPoint. Этот класс представляет одну точку табулированной функции. При проектировании класса я уделил внимание принципам инкапсуляции. Объекты класса содержат два поля: координату точки по оси X и координату по оси Y. В классе реализованы три конструктора: конструктор с параметрами координат (FunctionPoint(double x, double y)), конструктор копирования (FunctionPoint(FunctionPoint point)) для создания объекта с такими же координатами, как у существующей точки, и конструктор по умолчанию (FunctionPoint()), создающий точку с координатами (0, 0).

```java
package functions;

public class FunctionPoint {  19 usages
    private double x;  6 usages
    private double y;  6 usages

    // конструктор с заданными координатами
    public FunctionPoint(double x, double y) {  4 usages
        this.x = x;
        this.y = y;
    }

    // конструктор копирования существующей точки
    public FunctionPoint(FunctionPoint point) {  5 usages
        this.x = point.x;
        this.y = point.y;
    }

    // конструктор по умолчанию, создает точку (0, 0)
    public FunctionPoint() { x = 0; y = 0; }  no usages

    // возвращает координату x точки
    public double getX() { return x; }  17 usages

    // возвращает координату y точки
    public double getY() { return y; }  4 usages

    // устанавливает новое значение координаты x
    public void setX(double x) { this.x = x; }  no usages

    // устанавливает новое значение координаты y
    public void setY(double y) { this.y = y; }  1 usage
}
```

Задание №3

В пакете functions я создал класс TabulatedFunction. Объекты этого класса описывают табулированную функцию. Для хранения информации о точках функции используется массив элементов типа FunctionPoint. Особенностью реализации является поддержание упорядоченности точек по значению координаты X. Класс содержит два конструктора: TabulatedFunction(double leftX, double rightX, int pointsCount) создает объект функции с заданными границами области определения и количеством точек, а TabulatedFunction(double leftX, double rightX, double[] values) создает функцию с указанными значениями в точках. В обоих случаях точки распределяются равномерно по заданному интервалу.

```
package functions;

public class TabulatedFunction {  4 usages
    private FunctionPoint[] points;  33 usages
    private int pointsCount;  19 usages
    private static final double EPSILON = 1e-10;  10 usages

    // создает табулированную ф-цию с нулевыми значениями
    public TabulatedFunction(double leftX, double rightX, int pointsCount) {
        this.pointsCount = pointsCount;
        points = new FunctionPoint[pointsCount + 3];

        double step = (rightX - leftX) / (pointsCount - 1);

        for (int i = 0; i < pointsCount; i++) {
            points[i] = new FunctionPoint( x: leftX + step * i,  y: 0);
        }
    }

    // создает табулированную ф-цию с заданными значениями в точках
    public TabulatedFunction(double leftX, double rightX, double[] values) {
        pointsCount = values.length;
        points = new FunctionPoint[pointsCount + 3];

        double step = (rightX - leftX) / (pointsCount - 1);

        for (int i = 0; i < pointsCount; i++) {
            points[i] = new FunctionPoint( x: leftX + step * i, values[i]);
        }
    }
```

Задание №4

В классе TabulatedFunction я реализовал методы для работы с функцией. Метод getLeftDomainBorder() возвращает значение левой границы области определения, которое соответствует X-координате первой точки. Метод getRightDomainBorder() возвращает правую границу области определения, равную X-координате последней точки. Метод getFunctionValue(double x) вычисляет значение функции в произвольной точке X. Если точка находится вне области определения, метод возвращает специальное значение Double.NaN. Для вычисления значений между известными точками применяется линейная интерполяция, предполагающая линейное поведение функции на каждом интервале между соседними точками.

```java
// возвращает левую границу обл. опред. ф-ции
public double getLeftDomainBorder() {  2 usages
    return points[0].getX();
}

// возвращает правую границу обл. опред. ф-ции
public double getRightDomainBorder() {  2 usages
    return points[pointsCount - 1].getX();
}

// вычисляет значение ф-ции в заданной точке
public double getFunctionValue(double x) {  2 usages
    if (x < getLeftDomainBorder() - EPSILON || x > getRightDomainBorder() + EPSILON) {
        return Double.NaN;
    }

    for (int i = 0; i < pointsCount - 1; i++) {
        double x1 = points[i].getX();
        double x2 = points[i + 1].getX();

        if (x >= x1 - EPSILON && x <= x2 + EPSILON) {
            double y1 = points[i].getY();
            double y2 = points[i + 1].getY();

            return ((x - x1) * (y2 - y1)) / (x2 - x1) + y1;
        }
    }

    return Double.NaN;
}
```

Задание №5

В классе TabulatedFunction я добавил методы для работы с отдельными точками функции. Метод getPointsCount() возвращает общее количество точек. Метод getPoint(int index) предоставляет копию точки по указанному индексу, что обеспечивает защиту исходных данных. Метод setPoint(int index, FunctionPoint point) заменяет существующую точку на новую, при этом проверяется сохранение упорядоченности по координате X. Методы getPointX(int index) и getPointY(int index) возвращают соответствующие координаты точки, а setPointX(int index, double x) и setPointY(int index, double y) позволяют изменять эти координаты с соблюдением необходимых проверок.

```java
// возвращает количество точек в ф-ции
public int getPointsCount() { return pointsCount; } 1 usage

// возвращает копию точки по указанному индексу
public FunctionPoint getPoint(int index) { return new FunctionPoint(points[index]); } no usages

// заменяет точку по указанному индексу
public void setPoint(int index, FunctionPoint point){ 1 usage
    double newX = point.getX();

    if ((index != 0) && (index != pointsCount-1)){
        double prevX = points[index-1].getX();
        double nextX = points[index+1].getX();
        if((newX > prevX + EPSILON) && (newX < nextX - EPSILON)){
            points[index] = new FunctionPoint(point);
        }
    }
    else if ((index == 0) && (newX < points[1].getX() - EPSILON)) {
        points[index] = new FunctionPoint(point);
    }
    else if ((index == pointsCount-1) && (newX > points[index-1].getX() + EPSILON)) {
        points[index] = new FunctionPoint(point);
    }
    else return;
}

// возвращает координату x точки по индексу
public double getPointX(int index) { return points[index].getX(); } 1 usage

// возвращает координату y точки по индексу
public double getPointY(int index) { return points[index].getY(); } 1 usage

// изменяет координату x точки по индексу
public void setPointX(int index, double x){ setPoint(index, new FunctionPoint(x, points[index].getY())); }

// изменяет координату y точки по индексу
public void setPointY(int index, double y) { points[index].setY(y); } 1 usage
```

Задание №6

В классе TabulatedFunction я реализовал методы, изменяющие количество точек функции. Метод deletePoint(int index) удаляет точку с указанным индексом, при этом минимальное количество точек не может быть меньше двух. Метод addPoint(FunctionPoint point) добавляет новую точку в функцию, автоматически размещая ее в правильной позиции для сохранения упорядоченности по X. При добавлении проверяется отсутствие точек с таким же значением X. Для эффективного копирования элементов массива используется метод System.arraycopy(). Массив точек создается с запасом места, что позволяет избежать частого перевыделения памяти.

```
    // удаляет точку по указанному индексу
    public void deletePoint(int index) {  1 usage
        if (pointsCount <= 2) return;
        System.arraycopy(points,  srcPos: index + 1, points, index,  length: pointsCount - index - 1);
        pointsCount--;
    }

    // добавляет новую точку в ф-цию
    public void addPoint(FunctionPoint point) {  1 usage
        double newX = point.getX();
        for (int i = 0; i < pointsCount; i++) {
            if (Math.abs(points[i].getX() - newX) < EPSILON) {
                return;
            }
        }

        if (pointsCount == points.length) {
            FunctionPoint[] newPoints = new FunctionPoint[points.length * 2];
            System.arraycopy(points,  srcPos: 0, newPoints,  destPos: 0, pointsCount);
            points = newPoints;
        }

        int insInd = 0;
        while (insInd < pointsCount && points[insInd].getX() < newX - EPSILON) {
            insInd++;
        }

        System.arraycopy(points, insInd, points,  destPos: insInd + 1,  length: pointsCount - insInd);

        points[insInd] = new FunctionPoint(point);
        pointsCount++;
    }
}
```

Задание №7

Я создал класс Main, содержащий основную точку входа в программу. В методе main() продемонстрирована работа всех разработанных классов. Создается экземпляр табулированной функции с тестовыми значениями, после чего проверяются различные операции: вычисление значений функции в разных точках, изменение координат существующих точек, удаление и добавление новых точек. Результаты каждой операции выводятся в консоль для визуальной проверки корректности работы программы.

```java
import functions.FunctionPoint;
import functions.TabulatedFunction;

public class Main {
    public static void main(String[] args) {
        double[] data = {2, 5, 10, 17, 26};
        TabulatedFunction func = new TabulatedFunction( leftX: 1,  rightX: 5, data);

        System.out.println("Область определения: " + func.getLeftDomainBorder() + " - " + func.getRightDomainBorder());
        showPoints(func);

        System.out.println("f(3,2) = " + func.getFunctionValue( x: 3,2));
        System.out.println("f(0) = " + func.getFunctionValue( x: 0));

        func.setPointY( index: 1,  y: 7);
        System.out.println("После смены Y:");
        showPoints(func);

        func.deletePoint( index: 2);
        System.out.println("После удаления:");
        showPoints(func);

        func.addPoint(new FunctionPoint( x: 2.8,  y: 15));
        System.out.println("После добавления:");
        showPoints(func);
    }

    private static void showPoints(TabulatedFunction f) {  4 usages
        System.out.println("  Список точек:");
        for (int i = 0; i < f.getPointsCount(); i++) {
            double x = f.getPointX(i);
            double y = f.getPointY(i);
            System.out.println("    " + i + ": x=" + x + " y=" + y);
        }
    }
}
```

Пример работы:

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Intell
Область определения: 1.0 - 5.0
  Список точек:
    0: x=1.0 y=2.0
    1: x=2.0 y=5.0
    2: x=3.0 y=10.0
    3: x=4.0 y=17.0
    4: x=5.0 y=26.0
f(3.2) = 11.400000000000002
f(0) = NaN
После смены Y:
  Список точек:
    0: x=1.0 y=2.0
    1: x=2.0 y=7.0
    2: x=3.0 y=10.0
    3: x=4.0 y=17.0
    4: x=5.0 y=26.0
После удаления:
  Список точек:
    0: x=1.0 y=2.0
    1: x=2.0 y=7.0
    2: x=4.0 y=17.0
    3: x=5.0 y=26.0
После добавления:
  Список точек:
    0: x=1.0 y=2.0
    1: x=2.0 y=7.0
    2: x=2.8 y=15.0
    3: x=4.0 y=17.0
    4: x=5.0 y=26.0

Process finished with exit code 0
```