Лабораторная работа по ООП №2

Студент: Тарасова Полина

Группа: 6204-010302D

Задание 1

Я создала пакет functions. Потом в нём будут создаваться классы программы.

Задание 2

Я создала класс FunctionPoint в пакете functions. Он описывает одну точку табулированной функции. Когда я писала класс, я учитывала особенности инкапсуляции. Состояние объектов содержит координаты точки по оси абсцисс и оси ординат. В классе описаны конструкторы для создания объекта точки с заданными координатами (FunctionPoint(double x, double y)), для создания объекта точки с теми же координатами, что у указанной точки (FunctionPoint(FunctionPoint point)), для создания точки с координатами (0;0) (FunctionPoint()).

```
package functions;

// класс описывает одну точку табулированной функции
public class FunctionPoint {
    private double x;
    private double y;

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

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

    // точка (0, 0) по умолчанию
    public FunctionPoint() {
        x = 0;
        y = 0;
    }

    public double getX() {
        return x;
    }

    public void setX(double x) {
        this.x = x;
    }

    public double getY() {
        return y;
    }

    public void setY(double y) {
        this.y = y;
```

```
    }
}
```

Задание 3

В пакете functions я создала класс TabulatedFunction. Его объект описывает табулированную функцию. В нём используется массив типа FunctionPoint для хранения данных о точках. Точки в нём упорядочены по значению координаты x. В этом классе описываются такие конструкторы как TabulatedFunction(double leftX, double rightY, int pointCount) для создания объекта табулированной функции по заданным левой и правой границе области определения, а также количеству точек для табулирования, TabulatedFunction(double leftX, double rightY, double[] values) аналогично предыдущему конструктору, но вместо количества точек получает значения функции в виде массива.

```
package functions;

public class TabulatedFunction {
    private FunctionPoint[] points;
    private int pointsCount;

    // создаёт табулированную функцию с нулевыми значениями
    public TabulatedFunction(double leftX, double rightX, int pointsCount) {
        this.pointsCount = pointsCount;
        points = new FunctionPoint[pointsCount];
        double step = (rightX - leftX) / (pointsCount - 1);
        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step;
            points[i] = new FunctionPoint(x, 0);
        }
    }

    // создаёт табулированную функцию с готовыми значениями
    public TabulatedFunction(double leftX, double rightX, double[] values) {
        pointsCount = values.length;
        points = new FunctionPoint[pointsCount];
        double step = (rightX - leftX) / (pointsCount - 1);
        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step;
            points[i] = new FunctionPoint(x, values[i]);
        }
    }
```

Задание 4

В классе TabulatedFunction я описала методы, необходимые для работы с функцией. Метод double getLeftDomainBorder() возвращает значение левой границы области определения табулированной функции. Оно совпадает с абсциссой самой левой точки в описывающей функцию таблице. Метод double getRightDomainBorder() возвращает значение правой границы области определения табулированной функции. Метод double getFunctionValue(double x) возвращает значение функции в точке x, если эта точка лежит в области определения функции. В противном случае метод возвращает значение неопределённости. При расчёте значения функции используются линейная интерполяция, т.е. считается, что на интервале между заданными в таблице точками функция является прямой линией.

```
public double getRightDomainBorder() {
    return points[pointsCount - 1].getX();
}

public double getFunctionValue(double x) {
    if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
        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 && x <= x2) {
            double y1 = points[i].getY();
            double y2 = points[i + 1].getY();
            return y1 + (y2 - y1) * (x - x1) / (x2 - x1);
        }
    }
    return Double.NaN;
}
```

Задание 5

В классе TabulatedFunction я описала методы, необходимые для работы с точками табулированной функции. Метод int getPointsCount() возвращает количество точек. Метод FunctionPoint getPoint(int index) возвращает копию точки, соответствующей переданному индексу. Метод void setPoint(int index, FunctionPoint point) заменяет указанную точку табулированной функции на

переданную. Метод double getPointX(int index) возвращает значение абсциссы точки с указанным номером. Метод void setPointX(int index, double x) изменяет значение абсциссы точки с указанным номером. Аналогично метод setPoint(). Метод double getPointY(int index) возвращает значение ординаты точки с указанным номером. Метод void setPointY(int index, double y) изменяет значение ординаты точки с указанным номером.

```
public int getPointsCount() {
    return pointsCount;
}

public FunctionPoint getPoint(int index) {
    return new FunctionPoint(points[index]);
}

public void setPoint(int index, FunctionPoint point) {
    if (index > 0 && index < pointsCount - 1) {
        if (point.getX() > points[index - 1].getX() && point.getX() <
points[index + 1].getX()) {
            points[index] = new FunctionPoint(point);
        }
    } else if (index == 0 && point.getX() < points[1].getX()) {
        points[index] = new FunctionPoint(point);
    } else if (index == pointsCount - 1 && point.getX() > points[index -
1].getX()) {
        points[index] = new FunctionPoint(point);
    }
}

public double getPointX(int index) {
    return points[index].getX();
}

public void setPointX(int index, double x) {
    if (index > 0 && index < pointsCount - 1) {
        if (x > points[index - 1].getX() && x < points[index + 1].getX()) {
            points[index].setX(x);
        }
    }
}

public double getPointY(int index) {
    return points[index].getY();
}

public void setPointY(int index, double y) {
    points[index].setY(y);
}
```

Задание 6

В классе TabulatedFunction я описала методы, изменяющие количество точек табулированной функции. Метод void deletePoint(int index) удаляет заданную точку табулированной функции. Метод void

addPoint(FunctionPoint point) добавляет новую точку табулированной функции. Для копирования участков массивов я воспользовалась методом arraycopy() класса System.

```
    public void deletePoint(int index) {
        if (pointsCount <= 2) return;
        FunctionPoint[] newPoints = new FunctionPoint[pointsCount - 1];
        System.arraycopy(points, 0, newPoints, 0, index);
        System.arraycopy(points, index + 1, newPoints, index, pointsCount -
index - 1);
        points = newPoints;
        pointsCount--;
    }

    public void addPoint(FunctionPoint point) {
        for (int i = 0; i < pointsCount; i++) {
            if (points[i].getX() == point.getX()) {
                return;
            }
        }

        FunctionPoint[] newPoints = new FunctionPoint[pointsCount + 1];
        int i = 0;
        while (i < pointsCount && points[i].getX() < point.getX()) {
            newPoints[i] = points[i];
            i++;
        }
        newPoints[i] = new FunctionPoint(point);
        System.arraycopy(points, i, newPoints, i + 1, pointsCount - i);
        points = newPoints;
        pointsCount++;
    }
}
```

Задание 7

Я создала класс Main, содержащий точку входа программы.

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

public class Main {
    public static void main(String[] args) {
        double[] values = {0, 1, 4, 9, 16};
        TabulatedFunction func = new TabulatedFunction(0, 4, values);

        System.out.println("Левая граница: " + func.getLeftDomainBorder());
        System.out.println("Правая граница: " + func.getRightDomainBorder());
        System.out.println("Исходные точки функции:");
        printPoints(func);

        double x = 2.5;
        System.out.println("Значение функции в точке " + x + ": " +
func.getFunctionValue(x));
        System.out.println();

        func.setPointY(2, 10);
        System.out.println("После изменения Y в точке 2:");
        printPoints(func);
```

```
        func.deletePoint(1);
        System.out.println("После удаления точки 1:");
        printPoints(func);

        func.addPoint(new FunctionPoint(2.2, 5));
        System.out.println("После добавления точки (2.2, 5):");
        printPoints(func);
    }
    private static void printPoints(TabulatedFunction func) {
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.println("Point " + i + ": (" + func.getPointX(i) + ", " +
func.getPointY(i) + ")");
        }
        System.out.println();
    }

}
```

В методе main() я создала экземпляр класса TabulatedFunction и задала для него табулированные значения. Я вывела в консоль значения функции на разных рядах точек.
Пример вывода 1:

Левая граница: 0.0

Правая граница: 4.0

Исходные точки функции:

Point 0: (0.0, 0.0)

Point 1: (1.0, 1.0)

Point 2: (2.0, 4.0)

Point 3: (3.0, 9.0)

Point 4: (4.0, 16.0)

Значение функции в точке 2.5: 6.5

После изменения Y в точке 2:

Point 0: (0.0, 0.0)

Point 1: (1.0, 1.0)

Point 2: (2.0, 10.0)

Point 3: (3.0, 9.0)

Point 4: (4.0, 16.0)

После удаления точки 1:

Point 0: (0.0, 0.0)

Point 1: (2.0, 10.0)

Point 2: (3.0, 9.0)

Point 3: (4.0, 16.0)

После добавления точки (2.2, 5):

Point 0: (0.0, 0.0)

Point 1: (2.0, 10.0)

Point 2: (2.2, 5.0)

Point 3: (3.0, 9.0)

Point 4: (4.0, 16.0)

Пример вывода 2:

Левая граница: 0.0

Правая граница: 4.0

Исходные точки функции:

Point 0: (0.0, 0.0)

Point 1: (0.8, 3.0)

Point 2: (1.6, 23.0)

Point 3: (2.4000000000000004, 36.0)

Point 4: (3.2, 111.0)

Point 5: (4.0, 99.0)

Значение функции в точке 3.5: 106.5

После изменения Y в точке 2:

Point 0: (0.0, 0.0)

Point 1: (0.8, 3.0)

Point 2: (1.6, 23.0)

Point 3: (2.4000000000000004, 36.0)

Point 4: (3.2, 11.0)

Point 5: (4.0, 99.0)

После удаления точки 1:

Point 0: (0.0, 0.0)

Point 1: (0.8, 3.0)

Point 2: (2.4000000000000004, 36.0)

Point 3: (3.2, 11.0)

Point 4: (4.0, 99.0)

После добавления точки (2.2, 5):

Point 0: (0.0, 0.0)

Point 1: (0.8, 3.0)

Point 2: (2.4000000000000004, 36.0)

Point 3: (3.2, 11.0)

Point 4: (4.0, 99.0)

Point 5: (4.2, 6.0)