Лабораторная работа №3

Коновалов Сергей Сергеевич 6204-010302D

Содержание

Задание 1

Изучил классы исключений Java API:

- java.lang.Exception - базовый класс для проверяемых исключений

- java.lang.IndexOutOfBoundsException - выход за границы индекса

- java.lang.ArrayIndexOutOfBoundsException - выход за границы массива

- java.lang.IllegalArgumentException - недопустимый аргумент

- java.lang.IllegalStateException - недопустимое состояние объекта

Задание 2

Создал классы исключений FunctionPointIndexOutOfBoundsException и InappropriateFunctionPointException в пакете functions:

```java
public class FunctionPointIndexOutOfBoundsException extends IndexOutOfBoundsException {
    public FunctionPointIndexOutOfBoundsException() {
        super();
    }

    public FunctionPointIndexOutOfBoundsException(String message) {
        super(message);
    }

    public FunctionPointIndexOutOfBoundsException(String message, Throwable cause) {
        super(message);
        this.initCause(cause);
    }
}
```

```java
public class InappropriateFunctionPointException extends Exception {

    public InappropriateFunctionPointException() {
        super();
    }

    public InappropriateFunctionPointException(String message) {
        super(message);
    }

    public InappropriateFunctionPointException(String message, Throwable cause) {
        super(message, cause);
    }

    public InappropriateFunctionPointException(Throwable cause) {
        super(cause);
    }
}
```

Задание 3

В класс TabulatedFunction добавил проверки и исключения согласно требованиям:

```
if (leftX >= rightX) {
    throw new IllegalArgumentException("Левая граница области определения не может быть больше или равна правой");
}
if (pointsCount < 2) {
    throw new IllegalArgumentException("Количество точек не может быть меньше двух");
}
```

```
if (index < 0 || index >= pointsCount) {
    throw new FunctionPointIndexOutOfBoundsException(
        "Индекс " + index + " вне границ [0, " + (pointsCount - 1) + "]"
    );
}
```

```
if (index == 0) {
    if (newX >= points[1].getX() && !doubleEquals(newX, points[1].getX())) {
        throw new InappropriateFunctionPointException(
            "Новая координата x=" + newX + " должна быть меньше " + points[1].getX()
        );
    }
}
else if (index == pointsCount - 1) {
    if (newX <= points[pointsCount - 2].getX() && !doubleEquals(newX, points[pointsCount - 2].getX())) {
        throw new InappropriateFunctionPointException(
            "Новая координата x=" + newX + " должна быть больше " + points[pointsCount - 2].getX()
        );
    }
}
else {
    if ((newX <= points[index - 1].getX() && !doubleEquals(newX, points[index - 1].getX())) ||
        (newX >= points[index + 1].getX() && !doubleEquals(newX, points[index + 1].getX()))) {
        throw new InappropriateFunctionPointException(
            "Новая координата x=" + newX + " должна быть в интервале (" +
            points[index - 1].getX() + ", " + points[index + 1].getX() + ")"
        );
    }
}
```

```
if (pointsCount < 3) {
    throw new IllegalStateException("Невозможно удалить точку: количество точек не может быть меньше двух");
}
```

Задание 4

В пакете functions создал класс LinkedListTabulatedFunction и реализовал методы, требующие по заданию:

```
public class LinkedListTabulatedFunction implements TabulatedFunction {

    private FunctionNode head;

    private int pointsCount;

    private FunctionNode lastAccessedNode;
    private int lastAccessedIndex;

    private static final double EPSILON = 1e-10;

    private boolean doubleEquals(double a, double b) {
        return Math.abs(a - b) < EPSILON;
    }

    protected static class FunctionNode {
        private FunctionPoint point;
        private FunctionNode prev;
        private FunctionNode next;

        public FunctionNode(FunctionPoint point, FunctionNode prev, FunctionNode next) {
            this.point = point;
            this.prev = prev;
            this.next = next;
        }

        public FunctionNode(FunctionNode node) {
            this.point = new FunctionPoint(node.point);
            this.prev = node.prev;
            this.next = node.next;
        }

        FunctionPoint getPoint() {
            return point;
        }

        void setPoint(FunctionPoint point) {
            this.point = point;
        }

        FunctionNode getPrev() {
            return prev;
        }
```

```
    void setPoint(FunctionPoint point) {
        this.point = point;
    }

    FunctionNode getPrev() {
        return prev;
    }

    void setPrev(FunctionNode prev) {
        this.prev = prev;
    }

    FunctionNode getNext() {
        return next;
    }

    void setNext(FunctionNode next) {
        this.next = next;
    }
}
```

Задание 5

Сделал методы табулированной функции для LinkedListTabulatedFunction:

```
public LinkedListTabulatedFunction(double leftX, double rightX, int pointsCount) {
    if (leftX >= rightX) {
        throw new IllegalArgumentException("Левая граница области определения не может быть больше или равна правой");
    }
    if (pointsCount < 2) {
        throw new IllegalArgumentException("Количество точек не может быть меньше двух");
    }

    initializeList();

    double step = (rightX - leftX) / (pointsCount - 1);
    for (int i = 0; i < pointsCount; i++) {
        double x = leftX + i * step;
        addNodeToTail().setPoint(new FunctionPoint(x, 0.0));
    }
}

public LinkedListTabulatedFunction(double leftX, double rightX, double[] values) {
    if (leftX >= rightX) {
        throw new IllegalArgumentException("Левая граница области определения не может быть больше или равна правой");
    }
    if (values.length < 2) {
        throw new IllegalArgumentException("Количество точек не может быть меньше двух");
    }

    initializeList();

    double step = (rightX - leftX) / (values.length - 1);
    for (int i = 0; i < values.length; i++) {
        double x = leftX + i * step;
        addNodeToTail().setPoint(new FunctionPoint(x, values[i]));
    }
}
```

```
public double getLeftDomainBorder() {
    if (pointsCount == 0) {
        throw new IllegalStateException("Функция не содержит точек");
    }
    return head.getNext().getPoint().getX();
}

public double getRightDomainBorder() {
    if (pointsCount == 0) {
        throw new IllegalStateException("Функция не содержит точек");
    }
    return head.getPrev().getPoint().getX();
}
```

```java
public double getFunctionValue(double x) {
    if (pointsCount == 0) return Double.NaN;

    FunctionNode startNode = (lastAccessedIndex != -1) ? lastAccessedNode : head.getNext();
    int startIndex = (lastAccessedIndex != -1) ? lastAccessedIndex : 0;

    FunctionNode currentNode = startNode;
    int checkedNodes = 0;

    while (checkedNodes < pointsCount) {
        FunctionPoint point1 = currentNode.getPoint();
        FunctionPoint point2 = currentNode.getNext().getPoint();

        if (point1 == null || point2 == null) {
            currentNode = currentNode.getNext();
            checkedNodes++;
            continue;
        }

        double x1 = point1.getX();
        double x2 = point2.getX();

        if (x >= x1 && x <= x2) {
            double y1 = point1.getY();
            double y2 = point2.getY();

            lastAccessedNode = currentNode;
            lastAccessedIndex = findNodeIndex(currentNode);

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

        currentNode = currentNode.getNext();
        checkedNodes++;

        if (currentNode == head) {
            currentNode = head.getNext();
        }
    }

    return Double.NaN;
}
```

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

```
public void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException {
    FunctionNode node = getNodeByIndex(index);
    double newX = point.getX();

    FunctionNode prevNode = node.getPrev();
    FunctionNode nextNode = node.getNext();

    if (prevNode == head) {
        if (newX >= nextNode.getPoint().getX() && !doubleEquals(newX, nextNode.getPoint().getX())) {
            throw new InappropriateFunctionPointException(
                "Новая координата x=" + newX + " должна быть меньше " + nextNode.getPoint().getX()
            );
        }
    } else if (nextNode == head) {
        if (newX <= prevNode.getPoint().getX() && !doubleEquals(newX, prevNode.getPoint().getX())) {
            throw new InappropriateFunctionPointException(
                "Новая координата x=" + newX + " должна быть больше " + prevNode.getPoint().getX()
            );
        }
    } else {
        if ((newX <= prevNode.getPoint().getX() && !doubleEquals(newX, prevNode.getPoint().getX())) ||
            (newX >= nextNode.getPoint().getX() && !doubleEquals(newX, nextNode.getPoint().getX()))) {
            throw new InappropriateFunctionPointException(
                "Новая координата x=" + newX + " должна быть в интервале (" +
                prevNode.getPoint().getX() + ", " + nextNode.getPoint().getX() + ")"
            );
        }
    }

    node.setPoint(new FunctionPoint(point));
}

public double getPointX(int index) {
    return getNodeByIndex(index).getPoint().getX();
}

public double getPointY(int index) {
    return getNodeByIndex(index).getPoint().getY();
}
```

```java
public void setPointX(int index, double x) throws InappropriateFunctionPointException {
    FunctionPoint currentPoint = getNodeByIndex(index).getPoint();
    setPoint(index, new FunctionPoint(x, currentPoint.getY()));
}

public void setPointY(int index, double y) {
    FunctionNode node = getNodeByIndex(index);
    FunctionPoint currentPoint = node.getPoint();
    node.setPoint(new FunctionPoint(currentPoint.getX(), y));
}

public void deletePoint(int index) {
    deleteNodeByIndex(index);
}

public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException {
    FunctionNode currentNode = head.getNext();
    for (int i = 0; i < pointsCount; i++) {
        if (doubleEquals(point.getX(), currentNode.getPoint().getX())) {
            throw new InappropriateFunctionPointException(
                "Точка с x=" + point.getX() + " уже существует"
            );
        }
        currentNode = currentNode.getNext();
    }

    int insertIndex = 0;
    currentNode = head.getNext();
    while (insertIndex < pointsCount && point.getX() > currentNode.getPoint().getX()) {
        currentNode = currentNode.getNext();
        insertIndex++;
    }

    FunctionNode newNode = addNodeByIndex(insertIndex);
    newNode.setPoint(new FunctionPoint(point));

    FunctionNode prevNode = newNode.getPrev();
    FunctionNode nextNode = newNode.getNext();

    if ((prevNode != head && newNode.getPoint().getX() <= prevNode.getPoint().getX()) ||
        (nextNode != head && newNode.getPoint().getX() >= nextNode.getPoint().getX())) {
        deleteNodeByIndex(insertIndex);
        throw new InappropriateFunctionPointException(
            "Новая точка нарушает упорядоченность функции"
        );
    }
}
```

Задание 6

Класс TabulatedFunction переименовал в класс ArrayTabulatedFunction.

Создал интерфейс TabulatedFunction, содержащий объявления общих методов классов ArrayTabulatedFunction и LinkedListTabulatedFunction:

```
public interface TabulatedFunction {

    double getLeftDomainBorder();

    double getRightDomainBorder();

    double getFunctionValue(double x);

    int getPointsCount();

    FunctionPoint getPoint(int index);

    void setPoint(int index, FunctionPoint point) throws
InappropriateFunctionPointException;

    double getPointX(int index);

    double getPointY(int index);

    void setPointX(int index, double x) throws InappropriateFunctionPointException;

    void setPointY(int index, double y);

    void deletePoint(int index);

    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;
}
```

Задание 7

Создал Main.java для тестирования функций и проверок,

которая выводит в консоль информацию, изображенную на

следующих рисунках:

```
PS D:\Other\Study\GitHubLab3> javac functions\*.java
PS D:\Other\Study\GitHubLab3> javac Main.java
PS D:\Other\Study\GitHubLab3> java Main
||| Тестирование TabulatedFunction |||

Тестирование: ArrayTabulatedFunction
Область определения: [0.0, 10.0]
Количество точек: 5
Точки функции:
  [0] (0,00, 0,00)
  [1] (2,50, 0,00)
  [2] (5,00, 0,00)
  [3] (7,50, 0,00)
  [4] (10,00, 0,00)
Значения функции:
  f(0,00) = 0,00
  f(2,50) = 0,00
  f(5,00) = 0,00
  f(7,50) = 0,00
  f(10,00) = 0,00

Тестирование исключений:
FunctionPointIndexOutOfBoundsException: Индекс -1 вне границ [0, 4]
FunctionPointIndexOutOfBoundsException: Индекс 100 вне границ [0, 4]
InappropriateFunctionPointException: Новая координата x=-1.0 должна быть в интервале (2.5, 7.5)
InappropriateFunctionPointException: Точка с x=2.5 уже существует
IllegalStateException: Невозможно удалить точку: количество точек не может быть меньше двух
ОШИБКА: Точка с x=2.5 уже существует

================================================

Тестирование: LinkedListTabulatedFunction
Область определения: [0.0, 10.0]
Количество точек: 5
Точки функции:
  [0] (0,00, 1,00)
  [1] (2,50, 2,00)
  [2] (5,00, 3,00)
  [3] (7,50, 4,00)
  [4] (10,00, 5,00)
Значения функции:
  f(0,00) = 1,00
  f(2,50) = 2,00
  f(5,00) = 3,00
  f(7,50) = 4,00
  f(10,00) = 5,00

Тестирование исключений:
FunctionPointIndexOutOfBoundsException: Индекс -1 вне границ [0, 4]
FunctionPointIndexOutOfBoundsException: Индекс 100 вне границ [0, 4]
InappropriateFunctionPointException: Новая координата x=-1.0 должна быть в интервале (2.5, 7.5)
InappropriateFunctionPointException: Точка с x=2.5 уже существует
IllegalStateException: Невозможно удалить точку: количество точек не может быть меньше двух
ОШИБКА: Точка с x=2.5 уже существует

================================================

Тестирование исключений в конструкторах:
IllegalArgumentException: Левая граница области определения не может быть больше или равна правой
IllegalArgumentException: Левая граница области определения не может быть больше или равна правой
IllegalArgumentException: Количество точек не может быть меньше двух
IllegalArgumentException: Количество точек не может быть меньше двух
ArrayTabulatedFunction создан успешно
LinkedListTabulatedFunction создан успешно

================================================

Тестирование операций с точками:
Исходная функция:
  [0] (0,00, 0,00)
  [1] (2,00, 0,00)
  [2] (4,00, 0,00)

Добавление новой точки:
  [0] (0,00, 0,00)
  [1] (2,00, 0,00)
  [2] (2,50, 10,00)
  [3] (4,00, 0,00)

Добавление новой точки:
  [0] (0,00, 0,00)
  [1] (1,50, 5,00)
  [2] (2,00, 0,00)
  [3] (2,50, 10,00)
  [4] (4,00, 0,00)

Удаление точки:
  [0] (0,00, 0,00)
  [1] (1,50, 5,00)
  [2] (2,50, 10,00)
  [3] (4,00, 0,00)

Изменение точки:
  [0] (0,00, 0,00)
  [1] (1,80, 8,00)
  [2] (2,50, 10,00)
  [3] (4,00, 0,00)
PS D:\Other\Study\GitHubLab3>
```