Лабораторная работа №3

Заболотнов Николай Михайлович

6204-010302D

Task 1

Изучены классы исключений: Exception, IndexOutOfBoundsException, ArrayIndexOutOfBoundsException, IllegalArgumentException, IllegalStateException

Задание выполнено.

Task 2

В пакете functions создаем два класса исключений: FunctionPointIndexOutOfBoundsException, наследуетмый от класса IndexOutOfBoundsException (рис 1) и InappropriateFunctionPointException наследуемый от класса Exception (рис 2).

```
package functions;

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

Рис 1

```
package functions;

public class InappropriateFunctionPointException extends Exception {
    public InappropriateFunctionPointException(String message) {
        super(message);
    }
}
```

Рис 2

Задание выполнено.

Task 3

В класс TabulatedFunction добавим обработку исключений:

- Конструкторы выбрасывают исключение IllegalArgumentException (рис 3)
- getPoint(), setPoint(), getPointX(), setPointX(), getPointY(), setPointY() и deletePoint() исключение FunctionPointIndexOutOfBoundsException (рис 4 и 5)
- setPoint(), setPointX() и addPoint() исключение InappropriateFunctionPointException (рис 4 и 6)

- deletePoint() исключение IllegalStateException (рис 5)

```
public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount) {
    if (leftX >= rightX)
        throw new IllegalArgumentException("Левая граница больше или равна правой границе");
    if (pointsCount < 2)
        throw new IllegalArgumentException("Точек меньше двух");
    points = new FunctionPoint[pointsCount];
    double step = (rightX - leftX) / (pointsCount - 1);
    for (int i = 0; i < pointsCount; ++i, leftX += step)
        points[i] = new FunctionPoint(leftX, 0);
    PointsCount = pointsCount;
}

public ArrayTabulatedFunction(double leftX, double rightX, double[] values) {
    if (leftX >= rightX)
        throw new IllegalArgumentException("Левая граница больше или равна правой границе");
    if (values.length < 2)
        throw new IllegalArgumentException("Точек меньше двух");
    int pointsCount =  values.length;
    points = new FunctionPoint[pointsCount];
    double step = (rightX - leftX) / (pointsCount - 1);
    for (int i = 0; i < pointsCount; ++i, leftX += step)
        points[i] = new FunctionPoint(leftX, values[i]);
    PointsCount = pointsCount;
}
```

Рис 3

```
public FunctionPoint getPoint(int index) {
    if (index < 0 || index >= PointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Индекс выходит за границы набора точек");
    return new FunctionPoint(points[index]);
}

public void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException {
    if (index < 0 || index >= PointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Индекс выходит за границы набора точек");
    if ((index == 0 || index == PointsCount - 1)) {
        if (points[index].getX() != point.getX())
            throw new InappropriateFunctionPointException("x выходит за границы определения функции");
        points[index] = new FunctionPoint(point);
    }
    else
        if (point.getX() > points[index - 1].getX() && point.getX() < points[index + 1].getX())
            points[index] = new FunctionPoint(point);
        else
            throw new InappropriateFunctionPointException("x выходит за границы определения соседних точек");
}

public double getPointX(int index) {
    if (index < 0 || index >= PointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Индекс выходит за границы набора точек");
    return points[index].getX();
}

public void setPointX(int index, double x) throws InappropriateFunctionPointException {
    if (index < 0 || index >= PointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Индекс выходит за границы набора точек");
    if (index == 0 || index == PointsCount - 1 || x <= points[index - 1].getX() || x >= points[index + 1].getX())
        throw new InappropriateFunctionPointException("x выходит за границы определения соседних точек");
    points[index].setX(x);
}
```

Рис 4

```
public double getPointY(int index) {
    if (index < 0 || index >= PointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Индекс выходит за границы набора точек");
    return points[index].getY();
}

public void setPointY(int index, double y) {
    if (index < 0 || index >= PointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Индекс выходит за границы набора точек");
    points[index].setY(y);
}

public void deletePoint(int index) {
    if (index < 0 || index >= PointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Индекс выходит за границы набора точек");
    if (PointsCount < 3)
        throw new IllegalStateException("Кол-во элементов меньше трех");
    System.arraycopy(points, index + 1, points, index, PointsCount - index - 1);
    --PointsCount;
}
```

Рис 5

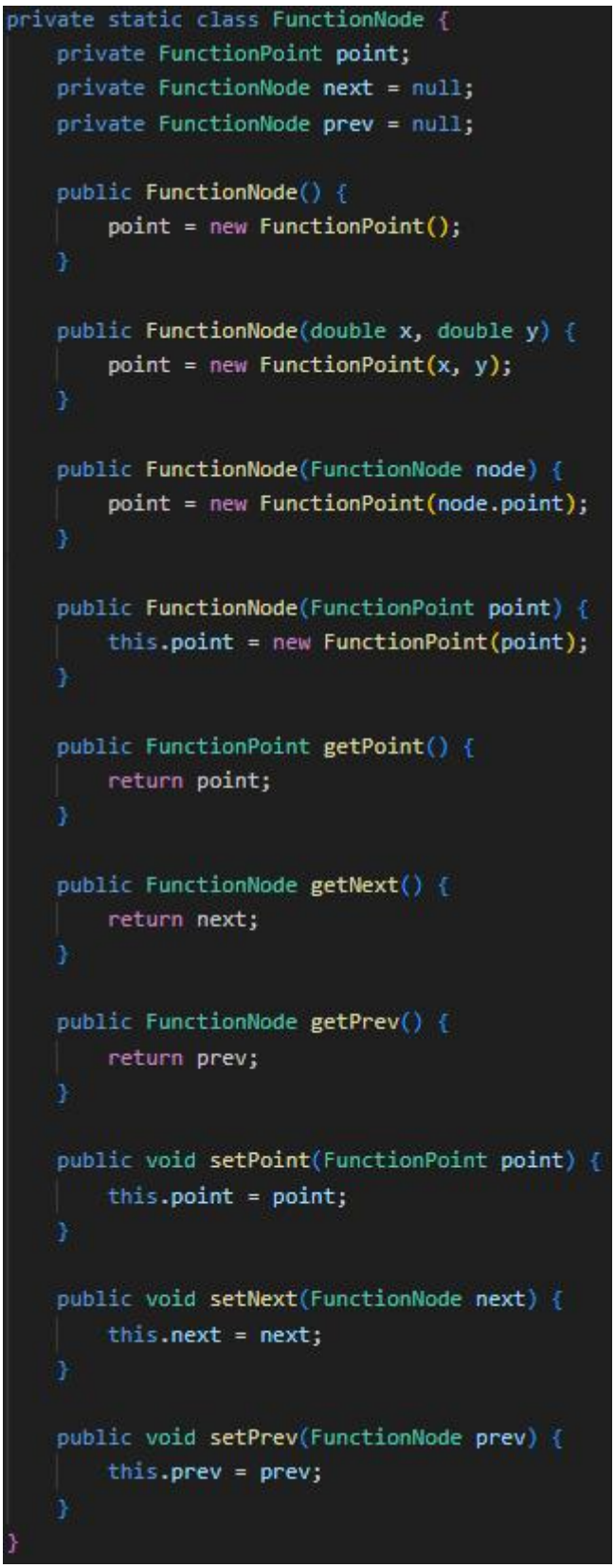
```java
public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException {
    int index = 0;
    if (point.getX() > points[PointsCount - 1].getX())
        index = PointsCount;
    else {
        if (point.getX() == points[index].getX())
                throw new InappropriateFunctionPointException("Точка с такой координатой  x определена");
        while (point.getX() >= points[index].getX()) {
            if (point.getX() == points[index].getX())
                throw new InappropriateFunctionPointException("Точка с такой координатой  x определена");
            ++index;
        }
    }
    if (PointsCount + 1 > points.length) {
        FunctionPoint[] newPoints = new FunctionPoint[PointsCount * 2 + 1];
        System.arraycopy(points, 0, newPoints, 0, index);
        newPoints[index] = new FunctionPoint(point);
        System.arraycopy(points, index, newPoints, index + 1, PointsCount - index);
        points = newPoints;
    }
    else {
        System.arraycopy(points, index, points, index + 1, PointsCount - index);
        points[index] = new FunctionPoint(point);
    }
    ++PointsCount;
}
```

Рис 6

Задание выполнено.

Task 4

Напишем класс LinkedListTabulatedFunction (двухсвязный циклический список). Для начала напишем класс элементов списка FunctionNode (рис 7).



```
private static class FunctionNode {
    private FunctionPoint point;
    private FunctionNode next = null;
    private FunctionNode prev = null;

    public FunctionNode() {
        point = new FunctionPoint();
    }

    public FunctionNode(double x, double y) {
        point = new FunctionPoint(x, y);
    }

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

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

    public FunctionPoint getPoint() {
        return point;
    }

    public FunctionNode getNext() {
        return next;
    }

    public FunctionNode getPrev() {
        return prev;
    }

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

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

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

Рис 7

Класс LinkedListTabulatedFunction будет содержать поля: head (первый элемент списка), size (длина списка), lastNode (последний элемент, к которому обращались), lastindex (последний индекс, к которому обращались). Реализуем приватные методы, которые нужны для написание следующих методов и

оптимизацию работы со списком: getNodeByIndex(), addNodeToTail(), addNodeByIndex() и deleteNodeByIndex() (рис 8 и 9).

```
private FunctionNode getNodeByIndex(int index) {
    FunctionNode node;
    if (index <= lastindex / 2 || index > (size + lastindex) / 2) {
        node = head;
        if (index > (size + lastindex) / 2) {
            node = node.prev;
            for (int i = size - 1; i != index; --i)
                node = node.prev;
        }
        else
            for (int i = 0; i != index; ++i)
                node = node.next;
    }
    else {
        node = lastNode;
        if (index < lastindex)
            for (int i = lastindex; i != index; --i)
                node = node.prev;
        else
            for (int i = lastindex; i != index; ++i)
                node = node.next;
    }
    lastindex = index;
    lastNode = node;
    return node;
}

private FunctionNode addNodeToTail(FunctionNode node) {
    FunctionNode tail = head.prev;
    head.prev = new FunctionNode(node);
    tail.next = head.prev;
    tail.next.prev = tail;
    tail.next.next = head;
    ++size;
    return tail.next;
}
```

Рис 8

```java
private FunctionNode addNodeByIndex(int index, FunctionNode node) {
    if (index == size)
        addNodeToTail(node);
    FunctionNode nextnode = getNodeByIndex(index);
    nextnode.prev.next = new FunctionNode(node);
    nextnode.prev.next.next = nextnode;
    nextnode.prev.next.prev = nextnode.prev;
    nextnode.prev = nextnode.prev.next;
    if (index == 0)
        head = head.prev;
    ++size;
    return nextnode.prev;
}

private FunctionNode deleteNodeByIndex(int index) {
    FunctionNode node = getNodeByIndex(index);
    node.next.prev = node.prev;
    node.prev.next = node.next;
    if (index == 0)
        head = head.next;
    --size;
    lastindex = 0;
    lastNode = head;
    return node;
}
```

Рис 9

Задание выполнено.

Task 5

В классе LinkedListTabulatedFunction реализуем те же конструкторы (рис 10) и методы, что и в классе TabulatedFunction.

```java
public LinkedListTabulatedFunction(double leftX, double rightX, int pointsCount) {
    if (leftX >= rightX)
        throw new IllegalArgumentException("Левая граница больше или равна правой границе");
    if (pointsCount < 2)
        throw new IllegalArgumentException("Точек меньше двух");
    double step = (rightX - leftX) / (pointsCount - 1);
    head = new FunctionNode(leftX, 0);
    head.next = head;
    head.prev = head;
    ++size;
    leftX += step;
    for(int i = 1; i != pointsCount; ++i, leftX += step) {
        addNodeToTail(new FunctionNode(leftX, 0));
    }
}

public LinkedListTabulatedFunction(double leftX, double rightX, double[] values) {
    if (leftX >= rightX)
        throw new IllegalArgumentException("Левая граница больше или равна правой границе");
    if (values.length < 2)
        throw new IllegalArgumentException("Точек меньше двух");
    int pointsCount = values.length;
    double step = (rightX - leftX) / (pointsCount - 1);
    head = new FunctionNode(leftX, values[0]);
    head.next = head;
    head.prev = head;
    ++size;
    leftX += step;
    for(int i = 1; i != pointsCount; ++i, leftX += step) {
        addNodeToTail(new FunctionNode(leftX, values[i]));
    }
}
```

Рис 10

Задание выполнено.

Task 6

Переименуем класс TabulatedFunction в класс ArrayTabulatedFunction. Создаем интерфейс TabulatedFunction (рис 11).

```java
public interface TabulatedFunction {
    double getLeftDomainBorder();
    double getRightDomainBorder();

    double getFunctionValue(double x);

    int getPointsCount();

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

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

    double getPointY(int index)
        throws FunctionPointIndexOutOfBoundsException;
    void setPointY(int index, double y)
        throws FunctionPointIndexOutOfBoundsException;

    void deletePoint(int index)
        throws FunctionPointIndexOutOfBoundsException, IllegalStateException;
    void addPoint(FunctionPoint point)
        throws InappropriateFunctionPointException;

    void outClass();
}
```

Рис 11

В оба класса добавим реализацию интерфейса (рис 12).

```java
implements TabulatedFunction
```

Рис 12

Задание выполнено.

Task 7

В классе Main сделаем точку входа программы и напишем метод test(), проверяющий работу исключений (рис 13).

```java
private static void test(TabulatedFunction f) {
    System.out.println("Точки:");
    f.outClass();
    try {
        f.getPoint(6);
    } catch (FunctionPointIndexOutOfBoundsException e) {
        System.out.println("FunctionPointIndexOutOfBoundsException: " + e.getMessage());
    }
    try {
        f.setPoint(8, new FunctionPoint(46, 1000));
        System.out.println("После изменения точки:");
        f.outClass();
    } catch (FunctionPointIndexOutOfBoundsException e) {
        System.out.println("FunctionPointIndexOutOfBoundsException: " + e.getMessage());
    } catch (InappropriateFunctionPointException e) {
        System.out.println("InappropriateFunctionPointException: " + e.getMessage());
    }
    try {
        System.out.printf("X = %.2f\n", f.getPointX(5));
    } catch (FunctionPointIndexOutOfBoundsException e) {
        System.out.println("FunctionPointIndexOutOfBoundsException: " + e.getMessage());
    }
    try {
        f.setPointX(5, 34);
    } catch (FunctionPointIndexOutOfBoundsException e) {
        System.out.println("FunctionPointIndexOutOfBoundsException: " + e.getMessage());
    } catch (InappropriateFunctionPointException e) {
        System.out.println("InappropriateFunctionPointException: " + e.getMessage());
    }
    try {
        System.out.printf("Y = %.2f\n", f.getPointY(4));
    } catch (FunctionPointIndexOutOfBoundsException e) {
        System.out.println("FunctionPointIndexOutOfBoundsException: " + e.getMessage());
    }
    try {
        f.setPointY(6, 15);
    } catch (FunctionPointIndexOutOfBoundsException e) {
        System.out.println("FunctionPointIndexOutOfBoundsException: " + e.getMessage());
    }
    try {
        f.deletePoint(1);
        System.out.println("После удаления точки:");
        f.outClass();
    } catch (FunctionPointIndexOutOfBoundsException e) {
        System.out.println("FunctionPointIndexOutOfBoundsException: " + e.getMessage());
    } catch (IllegalStateException e) {
        System.out.println("IllegalStateException: " + e.getMessage());
    }
    try {
        f.addPoint(new FunctionPoint(1,1));
        System.out.println("После добавления точки:");
        f.outClass();
    } catch (InappropriateFunctionPointException e) {
        System.out.println("IllegalStateException: " + e.getMessage());
    }
}
```

Рис 13

Задание выпонено.