ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С.П. КОРОЛЁВА»

КАФЕДРА «ТЕХНИЧЕСКАЯ КИБЕРНЕТИКА»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ

«Объектно-ориентированное программирование»

ЛАБОРАТОРНАЯ РАБОТА №5

Студент Волков Д.Н.

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

Задание №1

В FunctionPoint добавил методы

String toString(): Возвращает текстовое описание точки.

```
@Override
public String toString() {
    return "(" + x + "; " + y + ")";
}
```

boolean equals(Object o):Возвращает true тогда и только тогда, когда переданный объект также является точкой и его координаты в точности совпадают с координатами объекта, у которого вызывается метод.

```
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }
    if (o == null || getClass() != o.getClass()) {
        return false;
    }
    FunctionPoint otherPoint = (FunctionPoint) o;
    return Math.abs(this.x - otherPoint.x) < 1e-9 && Math.abs(this.y - otherPoint.y) < 1e-9;
}
```

int hashCode(): Возвращает значение хэш-кода для объекта точки.

```
@Override
 public int hashCode() {
     return Objects.hash(x, y);
 }
```

Object clone(): Возвращает объект-копию для объекта точки.

```
@Override
public FunctionPoint clone(){
    return new FunctionPoint(this.x, this.y);
}
```

Задание №2

В ArrayTabulatedFunction добавил методы

String toString(): Возвращает описание табулированной функции.

```java
@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append(str: "{");
    for (int i = 0; i < pointsCount; i++) {
        FunctionPoint point = getPoint(i);
        if (point != null) {
            sb.append(str: "(").append(point.getX()).append(str: " ; ").append(point.getY()).append(str: ")");
        }
    }
    sb.append(str: "}");
    return sb.toString();
}
```

boolean equals(Object o):Возвращает true тогда и только тогда, когда переданный объект также является табулированной функцией (реализует интерфейс TabulatedFunction) и её набор точек в точности совпадает с набором точек функции, у которой вызывается метод. В случае если переданный объект является экземпляром класса ArrayTabulatedFunction, метод напрямую обращается к элементам переданного объекта.

```java
@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }

    if (o == null) {
        return false;
    }

    TabulatedFunction otherFunction = (TabulatedFunction) o;
    if (this.pointsCount != otherFunction.getPointsCount()) {
        return false;
    }

    ArrayTabulatedFunction otherArrayFunction = null;
    if (o instanceof ArrayTabulatedFunction) {
        otherArrayFunction = (ArrayTabulatedFunction) o;
    }
    double epsilon = 1e-9;

    if (otherArrayFunction != null) {
        for (int i = 0; i < this.pointsCount; i++) {
            FunctionPoint p1 = this.points[i];
            FunctionPoint p2 = otherArrayFunction.points[i];
            if (p1 == null || p2 == null) {
                return false;
            }
            if (Math.abs(p1.getX() - p2.getX()) >= epsilon || Math.abs(p1.getY() - p2.getY()) >= epsilon) {
                return false;
            }
        }
    } else {
        for (int i = 0; i < this.pointsCount; i++) {
            FunctionPoint p1 = this.points[i];
            FunctionPoint p2 = otherFunction.getPoint(i);
            if (p1 == null || p2 == null) {
                return false;
            }
            if (Math.abs(p1.getX() - p2.getX()) >= epsilon || Math.abs(p1.getY() - p2.getY()) >= epsilon) {
                return false;
            }
        }
    }
    return true;
}
```

int hashCode(): Возвращает значение хэш-кода для объекта табулированной функции.

```
@Override
public int hashCode() {
    int result = pointsCount;
    for (int i = 0; i < pointsCount; i++) {
        FunctionPoint point = points[i];
        result = point.hashCode();
    }
    return result;
}
```

Object clone(): Возвращает объект-копию для объекта табулированной функции.

```
@Override
@SuppressWarnings("CloneDeclaresCloneNotSupported")
public TabulatedFunction clone(){
    try {
        ArrayTabulatedFunction clonedFunction = (ArrayTabulatedFunction) super.clone();
        clonedFunction.points = new FunctionPoint[this.pointsCount];
        for (int i = 0; i < this.pointsCount; i++) {
            FunctionPoint originalPoint = this.points[i];
            clonedFunction.points[i] = (FunctionPoint) originalPoint.clone();
        }
        return clonedFunction;
    } catch (CloneNotSupportedException e) {
        throw new AssertionError(message: "Клонирование невозможно", e);
    }
}
```

Задание №3

Аналогичные методы создал в LinkedListTabulatedFunction

String toString(): Возвращает описание табулированной функции.

```java
@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    FunctionNode Node = head.getNext();
    sb.append(str: "{");
    for (int i = 0; i < pointsCount; i++) {
        if (Node != null) {
            sb.append(str: "(").append(Node.getPoint().getX()).append(str: " ; ").append(Node.getPoint().getY()).append(str: ")");
            Node = Node.getNext();
        }
    }
    sb.append(str: "}");
    return sb.toString();
}
```

boolean equals(Object o):Возвращает true тогда и только тогда, когда переданный объект также является табулированной функцией (реализует интерфейс TabulatedFunction) и её набор точек в точности совпадает с набором точек функции, у которой вызывается метод. В случае если переданный объект является экземпляром класса LinkedListTabulatedFunction, метод напрямую обращается к элементам переданного объекта.

```java
@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }

    if (o == null) {
        return false;
    }

    TabulatedFunction otherFunction = (TabulatedFunction) o;
    if (this.pointsCount != otherFunction.getPointsCount()) {
        return false;
    }

    LinkedListTabulatedFunction otherLinkedListFunction = null;
    if (o instanceof LinkedListTabulatedFunction) {
        otherLinkedListFunction = (LinkedListTabulatedFunction) o;
    }
    double epsilon = 1e-9;

    if (otherLinkedListFunction != null) {
        for (int i = 0; i < this.pointsCount; i++) {
            FunctionNode p1 = this.getNodeByIndex(i);
            FunctionNode p2 = otherLinkedListFunction.getNodeByIndex(i);
            if (p1 == null || p2 == null) {
                return false;
            }
            if (Math.abs(p1.getPoint().getX() - p2.getPoint().getX()) >= epsilon || Math.abs(p1.getPoint().getY() - p2.getPoint().getY()) >= epsilon) {
                return false;
            }
        }
    } else {
        for (int i = 0; i < this.pointsCount; i++) {
            FunctionNode p1 = this.getNodeByIndex(i);
            FunctionPoint p2 = otherFunction.getPoint(i);
            if (p1 == null || p2 == null) {
                return false;
            }
            if (Math.abs(p1.getPoint().getX() - p2.getX()) >= epsilon || Math.abs(p1.getPoint().getY() - p2.getY()) >= epsilon) {
                return false;
            }
        }
    }
    return true;
}
```

int hashCode(): Возвращает значение хэш-кода для объекта табулированной функции.

```java
@Override
public int hashCode() {
    int result = pointsCount;
    for (int i = 0; i < pointsCount; i++) {
        FunctionPoint point = getPoint(i);
        result = point.hashCode();
    }
    return result;
}
```

Object clone(): Возвращает объект-копию для объекта табулированной функции.

```java
@Override
@SuppressWarnings("CloneDeclaresCloneNotSupported")
public TabulatedFunction clone(){
    try {
        LinkedListTabulatedFunction clonedFunction = (LinkedListTabulatedFunction) super.clone();
        clonedFunction.head = new FunctionNode(point: null, prev: null, next: null);
        clonedFunction.head.setNext(clonedFunction.head);
        clonedFunction.head.setPrev(clonedFunction.head);

        FunctionNode current = this.head.getNext();
        for (int i = 0; i < this.pointsCount; i++) {
            FunctionPoint newPoint = new FunctionPoint(current.getPoint().getX(), current.getPoint().getY());
            clonedFunction.addNodeToTail(newPoint);
            current = current.getNext();
        }
        return clonedFunction;
    } catch (CloneNotSupportedException e) {
        throw new AssertionError(message: "Клонирование невозможно", e);
    }
}
}
```

Задание №4

Сделал так, чтобы все объекты типа TabulatedFunction были клонируемыми с точки зрения JVM и внес метод clone() в этот интерфейс.

```
package functions;

public interface TabulatedFunction extends Function, Cloneable{
    int getPointsCount();
    FunctionPoint getPoint(int index);
    void setPoint(int index,FunctionPoint point) throws InappropriateFunctionPointException ;
    double getPointX(int index);
    void setPointX(int index,double x) throws InappropriateFunctionPointException;
    double getPointY(int index);
    void setPointY(int index,double y);
    void deletePoint(int index) throws InappropriateFunctionPointException;
    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;
    TabulatedFunction clone();
}
```

Задание 5

```
                toString()
array1.toString(): {(0.0 ; 1.4)(1.0 ; 6.0)(10.0 ; 4.4)}
array2.toString(): {(0.0 ; 1.4)(1.0 ; 6.0)(10.0 ; 4.4)}
array3.toString(): {(0.0 ; 1.0)(1.0 ; 3.5)(2.0 ; 5.0)}

linked1.toString(): {(0.0 ; 1.4)(1.0 ; 6.0)(10.0 ; 4.4)}
linked2.toString(): {(0.0 ; 1.4)(1.0 ; 6.0)(10.0 ; 4.4)}
linked3.toString(): {(0.0 ; 1.0)(1.0 ; 3.5)(2.0 ; 5.0)}

                equals():
(сравнение одинаковых массивов) array1.equals(array2): true
(сравнение разных массивов) array1.equals(array3): false
array1(null): false
(сравнение одинаковых списков) linked1.equals(linked2): true
(сравнение разных списков) linked1.equals(linked3): false
linked1(null): false
(сравнение одинакового массива и списка) array1.equals(linked2): true
(сравнение разного массива и списка) array1.equals(linked3): false

                hashCode()
array1.hashCode(): -1646001212
array2.hashCode(): -1646001212
array3.hashCode(): 1311681
(одинаковые массивы) array1.hashCode() == array2.hashCode(): true
linked1.hashCode(): -1646001212
linked2.hashCode(): -1646001212
linked3.hashCode(): 1311681
(одинаковые списки) linked1.hashCode() == linked2.hashCode(): true
(одинаковые список и массив) array1.hashCode() == linked2.hashCode(): true

                hashCode() после изменения массива и списка
(до изменения) array1.hashCode(): -1.646001212E9
(после изменения array1) array1.hashCode(): 73139137
(после изменения array1) array1.hashCode() != array1.hashCode(): true

                Тестирование глубокого клонирования
Оригинальный массив array1 :{(0.0 ; 1.4)(1.0 ; 6.0)(9.0 ; 30.0)}
Клон масса array1 :{(0.0 ; 1.4)(1.0 ; 6.0)(9.0 ; 30.0)}
Оригинальный список linked1 :{(0.0 ; 1.4)(1.0 ; 6.0)(10.0 ; 4.4)}
Клон списка linked1 :{(0.0 ; 1.4)(1.0 ; 6.0)(10.0 ; 4.4)}
array1.equals(arrayClone): true
linked1.equals(linkedClone): true

Оригинальный массив array1(после изменения): {(0.0 ; 1.4)(1.0 ; 6.0)(9.0 ; 436.0)}
Копия после изменения array1: {(0.0 ; 1.4)(1.0 ; 6.0)(9.0 ; 30.0)}
Оригинальный список linked1(после изменения): {(0.0 ; 1.4)(1.0 ; 436.0)(10.0 ; 4.4)}
Клон списка после изменения linked1: {(0.0 ; 1.4)(1.0 ; 6.0)(10.0 ; 4.4)}
array1.equals(arrayClone): false
linked1.equals(linkedClone): false

                 Проверка FunctionPoint
point.toString:(10.0; 30.0)
point2.toString:(12.0; 31.0)

point.equals(pointClone): true
point.equals(point2): false

point.hashCode() == pointCopy.hashCode(): true
point.hashCode() != point2.hashCode(): true
```