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 Ветров И.Р.

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

Задание №1

String toString() возвращает текстовое описание точки

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

boolean equals(Object 0) возвращает true, когда переданный объект является точкой и его координаты совпадают с координатами объекта, у которого вызывается метод

```
@Override
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

```
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

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++) {
        sb.append(str: "(")
          .append(String.format(format: "%.3f", points[i].getX()))
          .append(str: "; ")
          .append(String.format(format: "%.3f", points[i].getY()))
          .append(str: ")");
        if (i < pointsCount - 1) {
            sb.append(str: ", ");
        }
    }
    sb.append(str: "}");
    return sb.toString();
}
```

boolean equals(Object 0) возвращает true, когда переданный объекь также является табулированной функцией и её набор точек в точности совпадает с набором точек функции, у которой вызывается метод.

```java
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof TabulatedFunction)) return false;

    if (o instanceof ArrayTabulatedFunction) {
        ArrayTabulatedFunction other = (ArrayTabulatedFunction) o;

        if (this.pointsCount != other.pointsCount) return false;

        for (int i = 0; i < pointsCount; i++) {
            if (Math.abs(this.points[i].getX() - other.points[i].getX()) > EPSILON ||
                Math.abs(this.points[i].getY() - other.points[i].getY()) > EPSILON) {
                return false;
            }
        }
        return true;
    } else {
        TabulatedFunction other = (TabulatedFunction) o;

        if (this.pointsCount != other.getPointsCount()) return false;

        for (int i = 0; i < pointsCount; i++) {
            try {
                FunctionPoint thisPoint = this.getPoint(i);
                FunctionPoint otherPoint = other.getPoint(i);

                if (Math.abs(thisPoint.getX() - otherPoint.getX()) > EPSILON ||
                    Math.abs(thisPoint.getY() - otherPoint.getY()) > EPSILON) {
                    return false;
                }
            } catch (FunctionPointIndexOutOfBoundsException e) {
                return false;
            }
        }
        return true;
    }
}
```

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

```java
@Override
public TabulatedFunction clone() {
    FunctionPoint[] pointsCopy = new FunctionPoint[pointsCount];
    for (int i = 0; i < pointsCount; i++) {
        pointsCopy[i] = new FunctionPoint(points[i]);
    }

    try {
        return new ArrayTabulatedFunction(pointsCopy);
    } catch (IllegalArgumentException e) {
        return null;
    }
}
```

Задание №3

Аналогично всё в LinkedListTabulatedFunction.

toString()

```java
@Override
public String toString() {
    if (pointsCount == 0) return "{}";

    StringBuilder sb = new StringBuilder();
    sb.append(str: "{");

    FunctionNode current = head.getNext();
    while (current != head) {
        FunctionPoint point = current.getPoint();
        sb.append(String.format(format: "(%.3f; %.3f)", point.getX(), point.getY()));

        current = current.getNext();
        if (current != head) sb.append(str: ", ");
    }

    sb.append(str: "}");
    return sb.toString();
}
```

equals()

```java
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof TabulatedFunction)) return false;

    TabulatedFunction other = (TabulatedFunction) o;
    if (this.pointsCount != other.getPointsCount()) return false;

    if (o instanceof LinkedListTabulatedFunction) {
        LinkedListTabulatedFunction otherList = (LinkedListTabulatedFunction) o;

        FunctionNode current1 = this.head.getNext();
        FunctionNode current2 = otherList.head.getNext();

        while (current1 != this.head && current2 != otherList.head) {
            FunctionPoint p1 = current1.getPoint();
            FunctionPoint p2 = current2.getPoint();

            if (!p1.equals(p2)) return false;

            current1 = current1.getNext();
            current2 = current2.getNext();
        }
        return true;
    } else {
        for (int i = 0; i < pointsCount; i++) {
            FunctionPoint p1 = this.getPoint(i);
            FunctionPoint p2 = other.getPoint(i);
            if (!p1.equals(p2)) return false;
        }
        return true;
    }
}
```

hashCode()

```java
@Override
public int hashCode() {
    int result = pointsCount;
    FunctionNode current = head.getNext();
    while (current != head) {
        result ^= current.getPoint().hashCode();
        current = current.getNext();
    }
    return result;
}
```

clone()

```java
@Override
public TabulatedFunction clone() {
    FunctionPoint[] pointsArray = new FunctionPoint[pointsCount];
    FunctionNode current = head.getNext();
    for (int i = 0; i < pointsCount; i++) {
        pointsArray[i] = new FunctionPoint(current.getPoint());
        current = current.getNext();
    }
    return new LinkedListTabulatedFunction(pointsArray);
}
```

Задание №4

```java
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

```
1. МЕТОД toString():
   Array функции:
   - array1: {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)}
   - array2: {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)}
   - array3: {(0,000; 1,000), (1,000; 3,500), (2,000; 5,000)}

   LinkedList функции:
   - linked1: {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)}
   - linked2: {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)}
   - linked3: {(0,000; 1,000), (1,000; 3,500), (2,000; 5,000)}

2. МЕТОД equals():
   Сравнение Array функций:
   - array1.equals(array2): true (должно быть true)
   - array1.equals(array3): false (должно быть false)
   - array1.equals(null):   false (должно быть false)

   Сравнение LinkedList функций:
   - linked1.equals(linked2): true (должно быть true)
   - linked1.equals(linked3): false (должно быть false)
   - linked1.equals(null):    false (должно быть false)

   Сравнение Array и LinkedList:
   - array1.equals(linked2):  true (должно быть true)
   - array1.equals(linked3):  false (должно быть false)

3. МЕТОД hashCode():
   Array функции:
   - array1.hashCode(): -1074790400
   - array2.hashCode(): -1074790400
   - array3.hashCode(): 1075314691
   - array1.hashCode() == array2.hashCode(): true (должно быть true)

   LinkedList функции:
   - linked1.hashCode(): 1043334087
   - linked2.hashCode(): 1043334087
   - linked3.hashCode(): -1040710718
   - linked1.hashCode() == linked2.hashCode(): true (должно быть true)

   Согласованность Array и LinkedList:
   - array1.hashCode() == linked2.hashCode(): false (должно быть true)

4. ИЗМЕНЕНИЕ ОБЪЕКТА И ПРОВЕРКА hashCode():
   До изменения:
   - array1.hashCode():  -1074790400
   - linked1.hashCode(): 1043334087

   После изменения:
   - array1.hashCode():  642121731
   - linked1.hashCode(): 1045169095
   - Хэш изменился для array1:  true
   - Хэш изменился для linked1: true
```

```
5. МЕТОД clone() (глубокое копирование):
   До клонирования:
   - array1:  {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)}
   - linked1: {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)}

   После клонирования:
   - array1:  {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)}
   - arrayClone: {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)}
   - linked1: {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)}
   - linkedClone: {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)}

   Проверка равенства до изменений:
   - array1.equals(arrayClone):  true (должно быть true)
   - linked1.equals(linkedClone): true (должно быть true)

   После изменения оригиналов:
   - array1 (изменен):  {(0,000; 1,400), (1,000; 6,000), (2,000; 100,000)}
   - arrayClone: {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)} (не должен измениться)
   - linked1 (изменен): {(0,000; 1,400), (1,000; 200,000), (2,000; 4,400)}
   - linkedClone: {(0,000; 1,400), (1,000; 6,000), (2,000; 4,400)} (не должен измениться)

   Проверка равенства после изменений:
   - array1.equals(arrayClone):  false (должно быть false)
   - linked1.equals(linkedClone): false (должно быть false)

6. ТЕСТИРОВАНИЕ FunctionPoint:
   - point1: (10.0; 30.0)
   - point2: (12.0; 31.0)
   - pointClone: (10.0; 30.0)

   Методы FunctionPoint:
   - point1.equals(pointClone): true (должно быть true)
   - point1.equals(point2):     false (должно быть false)
   - point1.hashCode() == pointClone.hashCode(): true (должно быть true)
   - point1.hashCode() == point2.hashCode():     false (должно быть false)
```