ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА»

Институт «Информатики и кибернетики»

Специальность «Фотоника и оптоинформатика 6201-120303D»

Отчет по лабораторной работе № 5

Выполнил: студент Султанова А.М.,

группа 6201-120303D

Проверил: преподаватель Борисов Д.С.

Самара 2025

Задание 1

Переопределим в классе FunctionPoint следующие методы:

toString() возвращает текстовое описание точки (x; y).

equals(Object o) возвращает true, если другой объект тоже FunctionPoint и координаты совпадают. Необходимо корректно сравнивать double.

clone(): возвращает объект-копию, простое клонирование достаточно.

```
public String toString() {  new *
    return "(" + x + "; " + y + ")";
}

public boolean equals(Object o) {  new *
    if (o == this) return true;
    if (o == null) return false;
    if (!(o instanceof FunctionPoint)) return false;

    FunctionPoint p = (FunctionPoint) o;

    return Math.abs(this.x - p.x) < 1e-9 && Math.abs(this.y - p.y) < 1e-9;
}

public int hashCode() {  new *
    long xb = Double.doubleToLongBits(x);
    long yb = Double.doubleToLongBits(y);

    int x1 = (int)(xb >>> 32);
    int x2 = (int)(xb & 0xFFFFFFFF);
    int y1 = (int)(yb >>> 32);
    int y2 = (int)(yb & 0xFFFFFFFF);

    return x1 ^ x2 ^ y1 ^ y2;
}

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

Задание 2

В классе ArrayTabulatedFunction были переопределены методы, унаследованные от класса Object, для расширения функционала табулированной функции:

toString() - формирует текстовое представление функции в виде множества точек {(x0; y0), (x1; y1), …}. Для эффективного создания строки использовался класс StringBuilder и метод append(), что позволило последовательно добавлять координаты каждой точки и разделять их запятыми.

equals(Object o) - реализовано сравнение с любым объектом, реализующим интерфейс TabulatedFunction. Метод возвращает true, если количество точек совпадает и все координаты совпадают с точностью до малой погрешности (EPS). Для объектов класса ArrayTabulatedFunction использовано прямое обращение к массиву точек для ускорения работы.

hashCode() - рассчитан на основе количества точек и хэш-кодов всех точек функции. Используется операция XOR для объединения хэш-кодов, что обеспечивает различие хэш-кодов функций с разным набором точек.

clone() - выполнено глубокое клонирование. Создаётся новый массив точек, каждая точка клонируется с помощью метода clone(), и создаётся новый объект ArrayTabulatedFunction, содержащий копии всех точек. Это обеспечивает независимость клона от исходного объекта.

```java
public String toString() {  new *
    StringBuilder sb = new StringBuilder();
    sb.append("{");
    for (int i = 0; i < size; i++) {
        sb.append("(").append(points[i].getX()).append("; ").append(points[i].getY()).append(")");
        if (i != size - 1) sb.append(", ");
    }
    sb.append("}");
    return sb.toString();
}

public boolean equals(Object o) {  new *
    if (o == this) return true;
    if (o == null) return false;
    if (!(o instanceof TabulatedFunction)) return false;

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

    if (o instanceof ArrayTabulatedFunction) {
        ArrayTabulatedFunction af = (ArrayTabulatedFunction) o;
        for (int i = 0; i < size; i++) {
            if (Math.abs(this.points[i].getX() - af.points[i].getX()) > 1e-9 || Math.abs(this.points[i].getY() - af.points[i].getY()) > 1e-9)
                return false;
        }
    } else {
        for (int i = 0; i < size; i++) {
            if (Math.abs(this.points[i].getX() - f.getPointX(i)) > 1e-9 || Math.abs(this.points[i].getY() - f.getPointY(i)) > 1e-9)
                return false;
        }
    }
    return true;
}

public int hashCode() {  new *
    int hash = size;
    for (int i = 0; i < size; i++) {
        hash ^= points[i].hashCode();
    }
    return hash;
}

public Object clone() throws CloneNotSupportedException {  new *
    FunctionPoint[] newPoints = new FunctionPoint[size];
    for (int i = 0; i < size; i++) {
        newPoints[i] = (FunctionPoint) points[i].clone(); // глубокое клонирование
    }
    return new ArrayTabulatedFunction(newPoints);
}
```

Задание 3

Расширяем класс LinkedListTabulatedFunction, переопределив методы toString(), equals(), hashCode() и clone().

Метод toString() - Создан для текстового представления табулированной функции.Используется StringBuilder для формирования строки вида {(x0; y0), (x1; y1), …}.Цикл проходит по всем узлам списка (FunctionNode) и добавляет координаты точек.

Метод equals(Object o) - Проверяет равенство двух табулированных функций.

Сначала сравнивается тип объекта и количество точек. Затем последовательно сравниваются координаты всех точек с использованием точности EPS для чисел с плавающей запятой. Реализовано так, что метод корректно работает при сравнении с любым объектом, реализующим интерфейс TabulatedFunction.

Метод hashCode() - Формирует уникальный хэш-код функции. Используется побитовое XOR между хэш-кодами всех точек и размером списка. Обеспечивает согласованность методов equals() и hashCode().

Метод clone() - Реализует глубокое клонирование списка. Создаётся новый объект LinkedListTabulatedFunction, поочерёдно добавляются копии всех точек из исходного списка. Гарантирует, что изменение исходного списка после клонирования не влияет на клон.

```
public String toString() {  new *
    StringBuilder sb = new StringBuilder();
    sb.append("{");
    FunctionNode current = head.next;
    while (current != head) {
        sb.append("(").append(current.point.getX()).append("; ").append(current.point.getY()).append(")");
        if (current.next != head) sb.append(", ");
        current = current.next;
    }
    sb.append("}");
    return sb.toString();
}
public boolean equals(Object o) {  new *
    if (this == o) return true;
    if (!(o instanceof TabulatedFunction)) return false;

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

    for (int i = 0; i < size; i++) {
        if (Math.abs(this.getPointX(i) - other.getPointX(i)) > EPS || Math.abs(this.getPointY(i) - other.getPointY(i)) > EPS) return false;
    }
    return true;
}
public int hashCode() {  new *
    int result = size;
    FunctionNode current = head.next;
    while (current != head) {
        result ^= current.point.hashCode();
        current = current.next;
    }
    return result;
}
public Object clone() throws CloneNotSupportedException {  new *
    // создаём массив для клонирования точек
    FunctionPoint[] clonedPoints = new FunctionPoint[this.size]; // используем size
    FunctionNode current = this.head.next;

    for (int i = 0; i < this.size; i++) {
        clonedPoints[i] = (FunctionPoint) current.point.clone(); // глубокое клонирование точки
        current = current.next;
    }

    // создаём новый объект LinkedListTabulatedFunction из массива точек
    return new LinkedListTabulatedFunction(clonedPoints);
}
```

Задание 4

В интерфейс TabulatedFunction добавлен метод Object clone() throws CloneNotSupportedException.

```java
package functions;

public interface TabulatedFunction extends Function {  22 usages  2 implementations  SultanovaArina *
    int getPointsCount();  6 usages  2 implementations  SultanovaArina
    double getPointX(int index);  12 usages  2 implementations  SultanovaArina
    double getPointY(int index);  8 usages  2 implementations  SultanovaArina
    void setPointX(int index, double x) throws InappropriateFunctionPointException;  no usages  2 implementations  SultanovaArina
    void setPointY(int index, double y);  no usages  2 implementations  SultanovaArina
    FunctionPoint getPoint(int index);  no usages  2 implementations  SultanovaArina
    void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException;  no usages  2 implementations  SultanovaArina
    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;  no usages  2 implementations  SultanovaArina
    void deletePoint(int index);  no usages  2 implementations  SultanovaArina

    Object clone() throws CloneNotSupportedException;  1 implementation  new *

}
```

Все классы, реализующие интерфейс (ArrayTabulatedFunction и LinkedListTabulatedFunction), реализуют этот метод с глубоким копированием точек, что делает объекты клонируемыми с точки зрения JVM.

```java
public Object clone() throws CloneNotSupportedException {  new *
    FunctionPoint[] newPoints = new FunctionPoint[size];
    for (int i = 0; i < size; i++) {
        newPoints[i] = (FunctionPoint) points[i].clone(); // глубокое клонирование
    }
    return new ArrayTabulatedFunction(newPoints);
}
```

```java
public Object clone() throws CloneNotSupportedException {  new *
    // создаём массив для клонирования точек
    FunctionPoint[] clonedPoints = new FunctionPoint[this.size]; // используем size
    FunctionNode current = this.head.next;

    for (int i = 0; i < this.size; i++) {
        clonedPoints[i] = (FunctionPoint) current.point.clone(); // глубокое клонирование точки
        current = current.next;
    }

    // создаём новый объект LinkedListTabulatedFunction из массива точек
    return new LinkedListTabulatedFunction(clonedPoints);
}
```

Задание 5

Проверим написанные методы:

```java
import functions.*;

public class Main {  SultanovaArina *
    public static void main(String[] args) throws CloneNotSupportedException {  new *
        
        // Создание тестовых объектов
        double[] values1 = {1.0, 2.0, 3.0, 4.0};
        double[] values2 = {1.0, 2.0, 3.1, 4.0}; // немного изменённая точка для проверки equals и hashCode

        ArrayTabulatedFunction arrayFunc1 = new ArrayTabulatedFunction( leftX: 0.0,  rightX: 3.0, values1);
        ArrayTabulatedFunction arrayFunc2 = new ArrayTabulatedFunction( leftX: 0.0,  rightX: 3.0, values1);
        ArrayTabulatedFunction arrayFunc3 = new ArrayTabulatedFunction( leftX: 0.0,  rightX: 3.0, values2);

        LinkedListTabulatedFunction listFunc1 = new LinkedListTabulatedFunction( leftX: 0.0,  rightX: 3.0, values1);
        LinkedListTabulatedFunction listFunc2 = new LinkedListTabulatedFunction( leftX: 0.0,  rightX: 3.0, values1);
        LinkedListTabulatedFunction listFunc3 = new LinkedListTabulatedFunction( leftX: 0.0,  rightX: 3.0, values2);

        // Проверка toString()
        System.out.println("\ntoString() ");
        System.out.println("ArrayTabulatedFunction: " + arrayFunc1);
        System.out.println("LinkedListTabulatedFunction: " + listFunc1);

        // Проверка equals()
        System.out.println("\nequals()");
        System.out.println("arrayFunc1.equals(arrayFunc2): " + arrayFunc1.equals(arrayFunc2)); // true
        System.out.println("arrayFunc1.equals(arrayFunc3): " + arrayFunc1.equals(arrayFunc3)); // false
        System.out.println("arrayFunc1.equals(listFunc1): " + arrayFunc1.equals(listFunc1));   // true
        System.out.println("listFunc1.equals(listFunc3): " + listFunc1.equals(listFunc3));     // false

        // Проверка hashCode()
        System.out.println("\nhashCode()");
        System.out.println("arrayFunc1.hashCode(): " + arrayFunc1.hashCode());
        System.out.println("arrayFunc2.hashCode(): " + arrayFunc2.hashCode());
        System.out.println("arrayFunc3.hashCode(): " + arrayFunc3.hashCode());
        System.out.println("listFunc1.hashCode(): " + listFunc1.hashCode());
        System.out.println("listFunc2.hashCode(): " + listFunc2.hashCode());
        System.out.println("listFunc3.hashCode(): " + listFunc3.hashCode());

        // Проверка clone()
        System.out.println("\n clone() и проверка глубокого клонирования");
        ArrayTabulatedFunction arrayClone = (ArrayTabulatedFunction) arrayFunc1.clone();
        LinkedListTabulatedFunction listClone = (LinkedListTabulatedFunction) listFunc1.clone();

        System.out.println("Клоны до изменения исходных объектов:");
        System.out.println("arrayClone: " + arrayClone);
        System.out.println("listClone: " + listClone);

        // Изменяем исходные объекты
        arrayFunc1.setPointY( index: 0,  y: 10.0);
        listFunc1.setPointY( index: 0,  y: 10.0);

        System.out.println("\nПосле изменения исходных объектов:");
        System.out.println("Исходный arrayFunc1: " + arrayFunc1);
        System.out.println("Клон arrayClone (должен остаться прежним): " + arrayClone);

        System.out.println("Исходный listFunc1: " + listFunc1);
```

```java
        System.out.println("Исходный listFunc1: " + listFunc1);
        System.out.println("Клон listClone (должен остаться прежним): " + listClone);

        // Проверка equals() между клоном и исходным после изменения
        System.out.println("\nПроверка equals() после изменения исходных объектов:");
        System.out.println("arrayFunc1.equals(arrayClone): " + arrayFunc1.equals(arrayClone)); // false
        System.out.println("listFunc1.equals(listClone): " + listFunc1.equals(listClone));     // false
    }
}
```

```
toString()
ArrayTabulatedFunction: {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0), (3.0; 4.0)}
LinkedListTabulatedFunction: {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0), (3.0; 4.0)}

equals()
arrayFunc1.equals(arrayFunc2): true
arrayFunc1.equals(arrayFunc3): false
arrayFunc1.equals(listFunc1): true
listFunc1.equals(listFunc3): false

hashCode()
arrayFunc1.hashCode(): 1074790404
arrayFunc2.hashCode(): 1074790404
arrayFunc3.hashCode(): -1931739131
listFunc1.hashCode(): 1074790404
listFunc2.hashCode(): 1074790404
listFunc3.hashCode(): -1931739131

 clone() и проверка глубокого клонирования
Клоны до изменения исходных объектов:
arrayClone: {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0), (3.0; 4.0)}
listClone: {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0), (3.0; 4.0)}

После изменения исходных объектов:
Исходный arrayFunc1: {(0.0; 10.0), (1.0; 2.0), (2.0; 3.0), (3.0; 4.0)}
Клон arrayClone (должен остаться прежним): {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0), (3.0; 4.0)}
Исходный listFunc1: {(0.0; 10.0), (1.0; 2.0), (2.0; 3.0), (3.0; 4.0)}
Клон listClone (должен остаться прежним): {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0), (3.0; 4.0)}

Проверка equals() после изменения исходных объектов:
arrayFunc1.equals(arrayClone): false
listFunc1.equals(listClone): false

Process finished with exit code 0
```