

Министерство науки и высшего образования Российской Федерации  
федеральное государственное автономное  
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет имени академика  
С.П. Королева»  
Институт информатики и кибернетики  
Кафедра технической кибернетики

Отчет по лабораторной работе №5

Дисциплина: «ООП»

Тема «Расширить возможности классов, связанных с табулированными функциями, переопределив в них методы, унаследованные из класса Object.»

Выполнил: Зорин Дмитрий  
Сергеевич

Группа: 6201-120303D

Самара, 2025

## Задание на лабораторную работу

### Задание 1

В классе `FunctionPoint` переопределил методы: `toString()`, `equals(Object o)`, `hashCode()`, `clone()`. Пример кода:

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

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof FunctionPoint)) return false;
    FunctionPoint p = (FunctionPoint) o;
    return Double.compare(x, p.x) == 0 && Double.compare(y, p.y) == 0;
}

@Override
public int hashCode() {
    long bitsX = Double.doubleToLongBits(x);
    long bitsY = Double.doubleToLongBits(y);
    int hashX = (int)(bitsX ^ (bitsX >>> 32));
    int hashY = (int)(bitsY ^ (bitsY >>> 32));
    return hashX ^ hashY;
}

@Override
public Object clone() {
    return new FunctionPoint(this);
}
```

Создаем точки и ее копии и сравниваем через `equals()`. Проверяем через `hashCode()`

## Задание 2

Здесь необходимо реализовать методы `toString()`, `equals()`, `hashCode()` и `clone()` для массива точек

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

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

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

    if (other instanceof ArrayTabulatedFunction arr) {
        FunctionPoint[] otherPoints = arr.getPoints();
        for (int i = 0; i < points.length; i++) {
            if (Double.compare(points[i].getX(), otherPoints[i].getX()) != 0 ||
                Double.compare(points[i].getY(), otherPoints[i].getY()) != 0) {
                return false;
            }
        }
        return true;
    }
}
```

```
        for (int i = 0; i < points.length; i++) {
            FunctionPoint p = other.getPoint(i);
            if (Double.compare(points[i].getX(), p.getX()) != 0 ||
                Double.compare(points[i].getY(), p.getY()) != 0) {
                return false;
            }
        }
        return true;
    }

    @Override
    public int hashCode() {
        int hash = points.length;
        for (FunctionPoint p : points) {
            long xBits = Double.doubleToLongBits(p.getX());
            long yBits = Double.doubleToLongBits(p.getY());
            int xHash = (int)(xBits ^ (xBits >>> 32));
            int yHash = (int)(yBits ^ (yBits >>> 32));
            hash ^= xHash ^ yHash;
        }
        return hash;
    }

    @Override
    public TabulatedFunction clone() {
        FunctionPoint[] newPoints = new FunctionPoint[points.length];
        for (int i = 0; i < points.length; i++) {
            newPoints[i] = new FunctionPoint(points[i]);
        }
        return new ArrayTabulatedFunction(newPoints);
    }
}
```

Теперь все правильно отображается, сравнивается и копируется

### Задание 3

Задание 3 аналогично заданию 2, здесь код необходимо для linked, только глубокое клонирование реализовать через пересборку списка

Экспонента:

```
@Override
public String toString() {
    StringBuilder sb = new StringBuilder("{");
    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();
}

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

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

    if (other instanceof LinkedListTabulatedFunction ll) {
        FunctionNode thisNode = this.head.next;
        FunctionNode otherNode = ll.head.next;
        while (thisNode != head) {
            if (Double.compare(thisNode.point.getX(), otherNode.point.getX()) != 0 ||
                Double.compare(thisNode.point.getY(), otherNode.point.getY()) != 0)
                return false;
            thisNode = thisNode.next;
            otherNode = otherNode.next;
        }
        return true;
    }

    FunctionNode node = head.next;
    for (int i = 0; i < size; i++) {
        FunctionPoint p = other.getPoint(i);
        if (Double.compare(node.point.getX(), p.getX()) != 0 ||
            Double.compare(node.point.getY(), p.getY()) != 0)
            return false;
        node = node.next;
    }
    return true;
}

@Override
public int hashCode() {
    int hash = size;
    FunctionNode node = head.next;
    while (node != head) {
        long xBits = Double.doubleToLongBits(node.point.getX());
        long yBits = Double.doubleToLongBits(node.point.getY());
        int xHash = (int) (xBits ^ (xBits >>> 32));
        int yHash = (int) (yBits ^ (yBits >>> 32));
        hash ^= xHash ^ yHash;
        node = node.next;
    }
    return hash;
}
```

```

@Override
public TabulatedFunction clone() {
    LinkedListTabulatedFunction copy = new LinkedListTabulatedFunction();
    FunctionNode current = this.head.next;
    while (current != this.head) {
        copy.addNodeToTail(new FunctionPoint(current.point));
        current = current.next;
    }
    return copy;
}

```

Как итог методы корректно работают для связного списка

#### Задание 4

Все объекты типа TabulatedFunction должны быть клонируемыми с точки зрения JVM и добавить метод clone() в интерфейс

Пример кода:

```

public interface TabulatedFunction extends Function, Serializable, Cloneable {

```

любой объект, реализующий TabulatedFunction, можно клонировать через JVM

## Задание 5

В этом задании мы проверяем работу написанных методов, для этого я создал отдельный класс Test:

```
package functions;

public class Test {
    public static void main(String[] args) throws InappropriateFunctionPointException {
        double[] yValues = {1, 4, 9};
        ArrayTabulatedFunction arrayFunc = new ArrayTabulatedFunction(0, 2, yValues);

        FunctionPoint[] points = {
            new FunctionPoint(0, 1),
            new FunctionPoint(1, 4),
            new FunctionPoint(2, 9)
        };
        LinkedListTabulatedFunction listFunc = new LinkedListTabulatedFunction(points);

        //Проверка toString()
        System.out.println("ArrayTabulatedFunction: " + arrayFunc);
        System.out.println("LinkedListTabulatedFunction: " + listFunc);

        //Проверка equals()
        ArrayTabulatedFunction arrayCopy = (ArrayTabulatedFunction) arrayFunc.clone();
        LinkedListTabulatedFunction listCopy = (LinkedListTabulatedFunction) listFunc.clone();

        System.out.println("arrayFunc.equals(arrayCopy): " + arrayFunc.equals(arrayCopy)); // true
        System.out.println("listFunc.equals(listCopy): " + listFunc.equals(listCopy)); // true
        System.out.println("arrayFunc.equals(listFunc): " + arrayFunc.equals(listFunc)); // false

        //Проверка hashCode()
        System.out.println("arrayFunc.hashCode(): " + arrayFunc.hashCode());
        System.out.println("arrayCopy.hashCode(): " + arrayCopy.hashCode());
        System.out.println("listFunc.hashCode(): " + listFunc.hashCode());
        System.out.println("listCopy.hashCode(): " + listCopy.hashCode());
    }
}
```

Пример работы кода:

ArrayTabulatedFunction: {(0.0; 1.0), (1.0; 4.0), (2.0; 9.0)}

LinkedListTabulatedFunction: {(0.0; 1.0), (1.0; 4.0), (2.0; 9.0)}

arrayFunc.equals(arrayCopy): true

listFunc.equals(listCopy): true

arrayFunc.equals(listFunc): true

arrayFunc.hashCode(): 1077018627

arrayCopy.hashCode(): 1077018627

listFunc.hashCode(): 1077018627

listCopy.hashCode(): 1077018627

После изменения arrayFunc.hashCode(): 159931450

Исходный arrayFunc: {(0.0; 1.0), (1.0; 4.002), (2.0; 9.0)}

Клон arrayCopy: {(0.0; 10.0), (1.0; 4.0), (2.0; 9.0)}

Исходный listFunc: {(0.0; 1.0), (1.0; 4.0), (2.0; 9.0)}

Клон listCopy: {(0.0; 20.0), (1.0; 4.0), (2.0; 9.0)}