

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 «Расширение возможности классов, связанных с табулированными
функциями»

Выполнил: Куликов Степан
Дмитриевич

Группа: 6201-120303D

Самара, 2025

Задание на лабораторную работу

Задание 1

Переопределите в классе FunctionPoint следующие методы.

Ход выполнения задания: public

```
String toString() {  
    return "(" + x + ";" + y + ")";  
}  
  
public boolean equals(Object o) {  
  
    if (this == o) return true;  
    if (o == null || getClass() != o.getClass()) return false;  
  
    FunctionPoint point = (FunctionPoint) o;  
  
    return Math.abs(point.x - x) < EPSILON &&  
        Math.abs(point.y - y) < EPSILON;  
}  
  
public int hashCode() {  
  
    long xBits = Double.doubleToLongBits(x);    long  
    yBits = Double.doubleToLongBits(y);  
  
    int xLow = (int) (xBits & 0xffffffffL);  
    int xHigh = (int) (xBits >> 32);  
  
    int yLow = (int) (yBits & 0xffffffffL);  
    int yHigh = (int) (yBits >> 32);
```

```
    return xLow ^ xHigh ^ yLow ^ yHigh;  
}
```

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

Задание 2

Переопределите в классе ArrayTabulatedFunction следующие методы.

Ход выполнения задания: public String toString() {

```
    StringBuilder sb = new StringBuilder("{}");
```

```
    for (int i = 0; i < pointsCount; i++) {  
        sb.append(point[i].toString());        if  
(i < pointsCount - 1)            sb.append(", ");  
    }
```

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

```
public boolean equals(Object o) {
```

```
    if (this == o) return true;  
    if (o == null) 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 (!this.getPoint(i).equals(other.getPoint(i))) {
            return false;
        }
    }
    return true;
}

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

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

    for (int i = 0; i < pointsCount; i++) {
        if (!this.getPoint(i).equals(other.getPoint(i)))
            return false;
    }
    return true;
}

return false;
}

public int hashCode()
{
    int hash = pointsCount;

    for (int i = 0; i < pointsCount; i++) {
hash ^= point[i].hashCode();
    }
}

```

```

        return hash;
    }

    public Object clone() {

        FunctionPoint[] newPoints = new FunctionPoint[pointsCount];

        for (int i = 0; i < pointsCount; i++) {
            newPoints[i] = (FunctionPoint) point[i].clone();
        }

        return new ArrayTabulatedFunction(newPoints);
    }

```

Задание 3

Аналогично, переопределить методы `toString()`, `equals()`, `hashCode()` и `clone()` в классе `LinkedListTabulatedFunction`.

Ход выполнения задания:

```

public String toString() {
    StringBuilder sb = new
    StringBuilder();    sb.append("{");

    FunctionNode current = head.next;
    for (int i = 0; i < pointsCount; i++) {
        sb.append("(").append(current.data.getX()).append(";");
        ).append(current.data.getY()).append(")");

        if (i < pointsCount - 1) sb.append(", ");
        current = current.next;
    }
}

```

```

    sb.append("}");    return
    sb.toString();
}

public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null) return false;

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

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

        FunctionNode a = this.head.next;
        FunctionNode b = other.head.next;

        while (a != this.head) {
            if (!a.data.equals(b.data)) return false;

            a = a.next;
            b = b.next;
        }

        return true;
    }

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

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

```

```

FunctionNode a = this.head.next;

for (int i = 0; i < pointsCount; i++) {
    if (!a.data.equals(other.getPoint(i)))
        return false;
    a = a.next;
}

return true;
}

return false;
}

public int hashCode() {
    int hash = pointsCount;

    FunctionNode current = head.next;

    for (int i = 0; i < pointsCount; i++) {
        long x = Double.doubleToLongBits(current.data.getX());      long
        y = Double.doubleToLongBits(current.data.getY());

        hash ^= (int)(x ^ (x >>> 32));
        hash ^= (int)(y ^ (y >>> 32));

        current = current.next;
    }
}

```

```

        return hash;
    }

    public Object clone() {
        LinkedListTabulatedFunction clone = new LinkedListTabulatedFunction(); // пустой конструктор

        FunctionNode current = this.head.next;
        while (current != this.head) { // идём по оригинальному списку
            clone.addPointInternal(new FunctionPoint(current.data.getX(),
                current.data.getY()));      current = current.next;
        }

        return clone;
    }

```

Задание 4

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

Ход выполнения задания:

```

public interface TabulatedFunction extends Cloneable {

    ...
}

Object clone() throws CloneNotSupportedException;
}

```

Задание 5

Проверьте работу написанных методов.

Ход выполнения задания:

```
System.out.println("\nПРОВЕРКА toString()");
```

```
TabulatedFunction arr1 = new ArrayTabulatedFunction(0, 4, new double[] {0, 1, 4,
9, 16});

TabulatedFunction list1 = new LinkedListTabulatedFunction(0, 4, new double[] {0,
1, 4, 9, 16});

System.out.println("ArrayTabulatedFunction:");
System.out.println(arr1.toString());

System.out.println("\nLinkedListTabulatedFunction:");
System.out.println(list1.toString());

System.out.println("\nПРОВЕРКА equals()");

TabulatedFunction arr2 = new ArrayTabulatedFunction(0, 4, new double[] {0, 1, 4,
9, 16});

TabulatedFunction list2 = new LinkedListTabulatedFunction(0, 4, new double[] {0,
1, 4, 9, 16});

System.out.println("arr1.equals(arr2): " + arr1.equals(arr2));
System.out.println("arr1.equals(list1): " + arr1.equals(list1));
System.out.println("list1.equals(list2): " + list1.equals(list2));

arr2.setPointY(2, arr2.getPoint(2).getY() + 0.001);

System.out.println("arr1.equals(arr2) после изменения: " + arr1.equals(arr2));
System.out.println("\nПРОВЕРКА hashCode()");

System.out.println("hashCode arr1: " + arr1.hashCode());
System.out.println("hashCode arr2: " + arr2.hashCode());
```

```
System.out.println("hashCode list1: " + list1.hashCode());
System.out.println("hashCode list2: " + list2.hashCode());

System.out.println("\nПРОВЕРКА clone()");

try {
    // Клонируем Array и LinkedList
    ArrayTabulatedFunction arrClone = (ArrayTabulatedFunction) arr1.clone();
    LinkedListTabulatedFunction listClone = (LinkedListTabulatedFunction)
        list1.clone();

    System.out.println("\nКлон Array:");
    System.out.println(arrClone);

    System.out.println("\nКлон LinkedList:");
    System.out.println(listClone);

    // Изменяем оригиналы    arr1.setPointY(1,
    999);    list1.setPointY(1,
    777);

    System.out.println("\nПосле изменения оригиналов:");
    System.out.println("\nОригинал Array:");
    System.out.println(arr1);

    System.out.println("\nКлон Array (НЕ ДОЛЖЕН ПОМЕНЯТЬСЯ):");
    System.out.println(arrClone);

    System.out.println("\nОригинал LinkedList:");
}
```

```
System.out.println(list1);

System.out.println("\nКлон LinkedList (НЕ ДОЛЖЕН ПОМЕНЯТЬСЯ):");
System.out.println(listClone);

} catch (CloneNotSupportedException e) {
    e.printStackTrace();
}

}
```

```
System.out.println("\nКОНЕЦ ПРОВЕРОК");
```

Команды в консоли:

```
C:\Users\fael7\Desktop\лр5\Lab-5-2025>javac Main.java
```

```
C:\Users\fael7\Desktop\лр5\Lab-5-2025>java Main
```

```
...
```

ПРОВЕРКА `toString()`

`ArrayTabulatedFunction:`

```
{(0.0; 0.0), (1.0; 1.0), (2.0; 4.0), (3.0; 9.0), (4.0; 16.0)}
```

`LinkedListTabulatedFunction:`

```
{(0.0; 0.0), (1.0; 1.0), (2.0; 4.0), (3.0; 9.0), (4.0; 16.0)}
```

ПРОВЕРКА `equals()`

`arr1.equals(arr2): true arr1.equals(list1):`

`true list1.equals(list2): true`

`arr1.equals(arr2) после изменения: false`

ПРОВЕРКА `hashCode()`

`hashCode arr1: 1703941`

`hashCode arr2: 617033240`

hashCode list1: 1703941

hashCode list2: 1703941

ПРОВЕРКА clone()

Клон Array:

{(0.0; 0.0), (1.0; 1.0), (2.0; 4.0), (3.0; 9.0), (4.0; 16.0)}

Клон LinkedList:

{(0.0; 0.0), (1.0; 1.0), (2.0; 4.0), (3.0; 9.0), (4.0; 16.0)}

После изменения оригиналов:

Оригинал Array:

{(0.0; 0.0), (1.0; 999.0), (2.0; 4.0), (3.0; 9.0), (4.0; 16.0)}

Клон Array (НЕ ДОЛЖЕН ПОМЕНЯТЬСЯ):

{(0.0; 0.0), (1.0; 1.0), (2.0; 4.0), (3.0; 9.0), (4.0; 16.0)}

Оригинал LinkedList:

{(0.0; 0.0), (1.0; 777.0), (2.0; 4.0), (3.0; 9.0), (4.0; 16.0)} Клон LinkedList
(НЕ ДОЛЖЕН ПОМЕНЯТЬСЯ):

{(0.0; 0.0), (1.0; 1.0), (2.0; 4.0), (3.0; 9.0), (4.0; 16.0)}

КОНЕЦ ПРОВЕРОК

Работа программы завершена.