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 Чячяков А.И.

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка

САМАРА 2025

Задание 1

Переопределил в классе FunctionPoint следующие методы.

• **String toString():** Должен возвращать текстовое описание точки. Например: (1.1; -7.5), где 1.1 и -7.5 – абсцисса и ордината точки соответственно.

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

• **int hashCode():** Должен возвращать значение хэш-кода для объекта точки. Можно выбрать реализацию хэш-функции или воспользоваться простейшей реализацией, основанной на применении операции исключающего ИЛИ (XOR). В этом случае хэш-код рассчитывается как побитовое XOR для набора значений типа int. Этот набор должен включать в себя всю информацию, описывающую состояние объекта, т.е. два значения координат. Поскольку они имеют тип double, необходимо без потерь перевести эту информацию к типу int, например, представив одно значение типа double (8 байт) как два значения типа int (4 байта и 4 байта). Сделать это можно с помощью метода Double.doubleToLongBits(), оператора побитового И (&) (для выделения младших четырёх байтов) и оператора битового логического сдвига (>>) (для выделения старших четырёх байтов).

• **Object clone():** Должен возвращать объект-копию для объекта точки. Достаточно простого клонирования, так как точка не имеет ссылок на другие объекты.

Код 1

FunctionPoint.java

```java
public class FunctionPoint implements Serializable {  41 usages

    @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 that = (FunctionPoint) o;

        double eps = 1e-9;
        return Math.abs(this.x - that.x) < eps &&
                Math.abs(this.y - that.y) < eps;
    }

    @Override
    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;
    }

    @Override
    public Object clone() {
        try {
            return super.clone();
        } catch (CloneNotSupportedException e) {
            return new FunctionPoint(this.x, this.y);
```

Задание 2

Переопределил в классе ArrayTabulatedFunction следующие методы.

• String toString(): Должен возвращать описание табулированной функции. Например: {(0.0; 1.2), (1.0; 3.8), (2.0; 15.2)}, где в круглых скобках указываются координаты точек.

• boolean equals(Object o): Должен возвращать 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, время работы метода должно быть сокращено за счёт прямого обращения к элементам состояния переданного объекта.

• int hashCode(): Должен возвращать значение хэш-кода для объекта табулированной функции. Можно выбрать реализацию хэш-функции или воспользоваться простейшей реализацией, основанной на применении операции исключающего ИЛИ (XOR). В этом случае хэш-код рассчитывается как побитовое XOR для набора значений типа int. В данный набор входят хэш-коды всех точек табулированной функции, а также количество точек в функции. Последнее нужно для того, чтобы значения хэш-кода были различны для функций, отличающихся наличием нулевой точки (например, {(-1; 1), (0; 0), (1, 1)} и {(-1; 1), (1, 1)}).

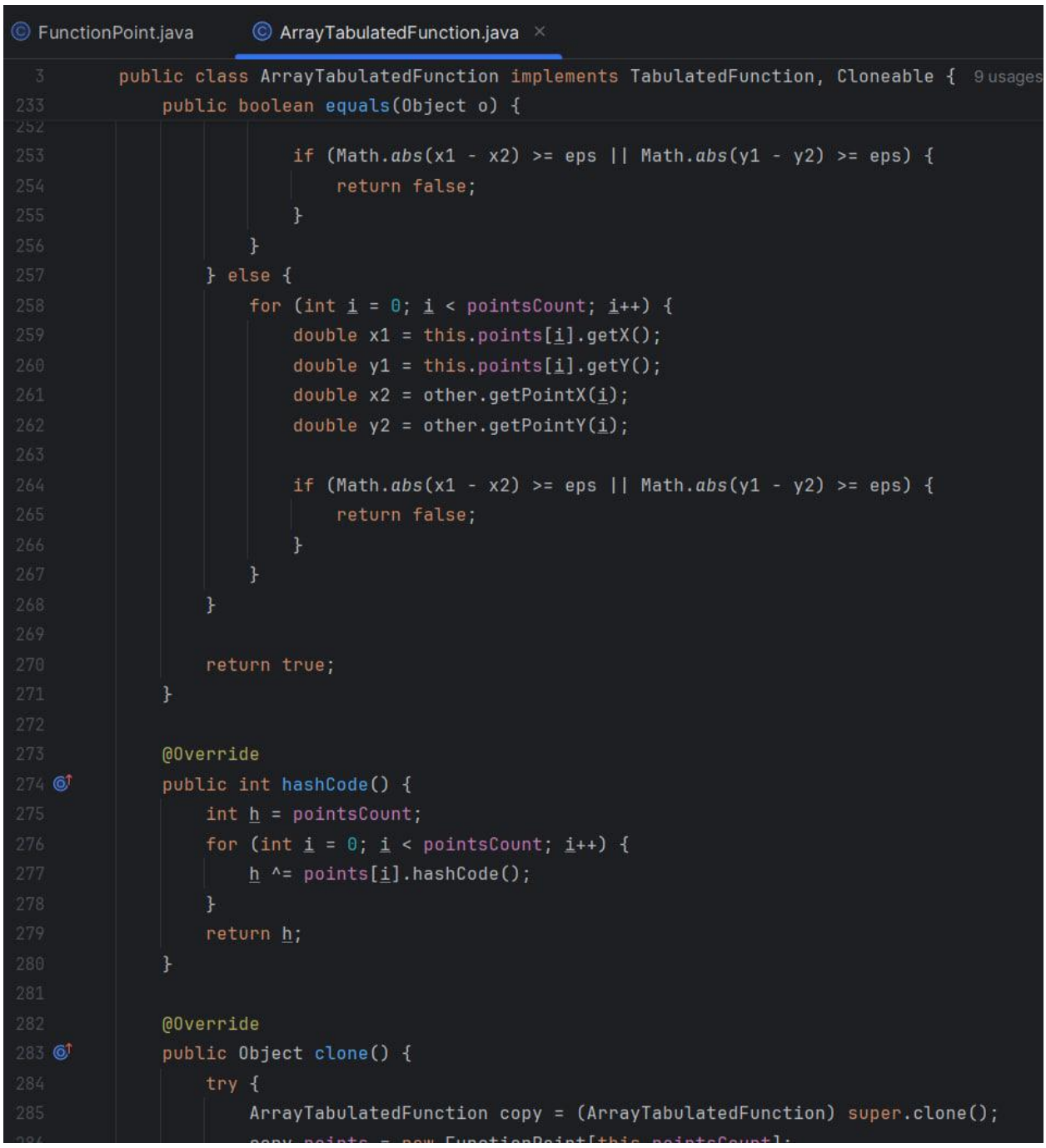
• Object clone(): Должен возвращать объект-копию для объекта табулированной функции. Поскольку табулированная функция ссылается на другие объекты, клонирование должно быть глубоким.

Код 2.1

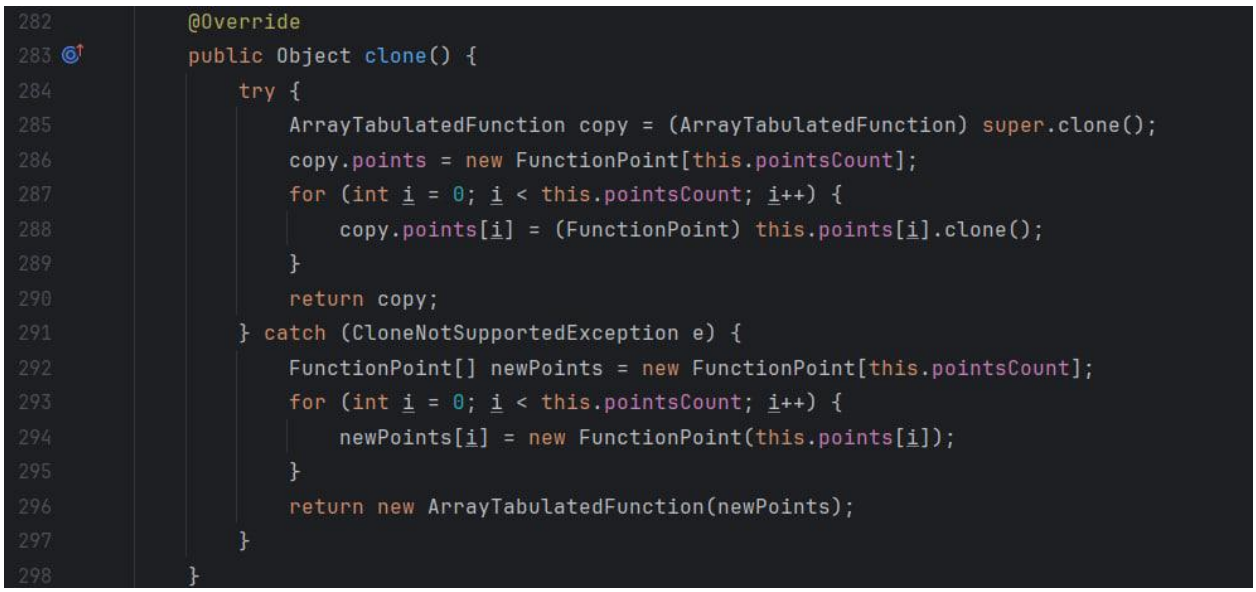
FunctionPoint.java | ArrayTabulatedFunction.java

```
3   public class ArrayTabulatedFunction implements TabulatedFunction, Cloneable {  9 usages
218     @Override
219     public String toString() {
220         StringBuilder sb = new StringBuilder();
221         sb.append('{');
222         for (int i = 0; i < pointsCount; i++) {
223             sb.append(points[i].toString());
224             if (i != pointsCount - 1) {
225                 sb.append(", ");
226             }
227         }
228         sb.append('}');
229         return sb.toString();
230     }
231
232     @Override
233     public boolean equals(Object o) {
234         if (this == o) return true;
235
236         if (!(o instanceof TabulatedFunction)) return false;
237         TabulatedFunction other = (TabulatedFunction) o;
238
239         if (this.getPointsCount() != other.getPointsCount()) {
240             return false;
241         }
242
243         double eps = 1e-9;
244
245         if (o instanceof ArrayTabulatedFunction) {
246             ArrayTabulatedFunction that = (ArrayTabulatedFunction) o;
247             for (int i = 0; i < pointsCount; i++) {
248                 double x1 = this.points[i].getX();
249                 double y1 = this.points[i].getY();
250                 double x2 = that.points[i].getX();
251                 double y2 = that.points[i].getY();
252
```

Код 2.2


FunctionPoint.java
ArrayTabulatedFunction.java
3 public class ArrayTabulatedFunction implements TabulatedFunction, Cloneable { 9 usages
233 public boolean equals(Object o) {
253 if (Math.abs(x1 - x2) >= eps || Math.abs(y1 - y2) >= eps) {
254 return false;
255 }
256 }
257 } else {
258 for (int i = 0; i < pointsCount; i++) {
259 double x1 = this.points[i].getX();
260 double y1 = this.points[i].getY();
261 double x2 = other.getPointX(i);
262 double y2 = other.getPointY(i);
263
264 if (Math.abs(x1 - x2) >= eps || Math.abs(y1 - y2) >= eps) {
265 return false;
266 }
267 }
268 }
269
270 return true;
271 }
272
273 @Override
274 public int hashCode() {
275 int h = pointsCount;
276 for (int i = 0; i < pointsCount; i++) {
277 h ^= points[i].hashCode();
278 }
279 return h;
280 }
281
282 @Override
283 public Object clone() {
284 try {
285 ArrayTabulatedFunction copy = (ArrayTabulatedFunction) super.clone();

Код 2.3



```
@Override
public Object clone() {
    try {
        ArrayTabulatedFunction copy = (ArrayTabulatedFunction) super.clone();
        copy.points = new FunctionPoint[this.pointsCount];
        for (int i = 0; i < this.pointsCount; i++) {
            copy.points[i] = (FunctionPoint) this.points[i].clone();
        }
        return copy;
    } catch (CloneNotSupportedException e) {
        FunctionPoint[] newPoints = new FunctionPoint[this.pointsCount];
        for (int i = 0; i < this.pointsCount; i++) {
            newPoints[i] = new FunctionPoint(this.points[i]);
        }
        return new ArrayTabulatedFunction(newPoints);
    }
}
```

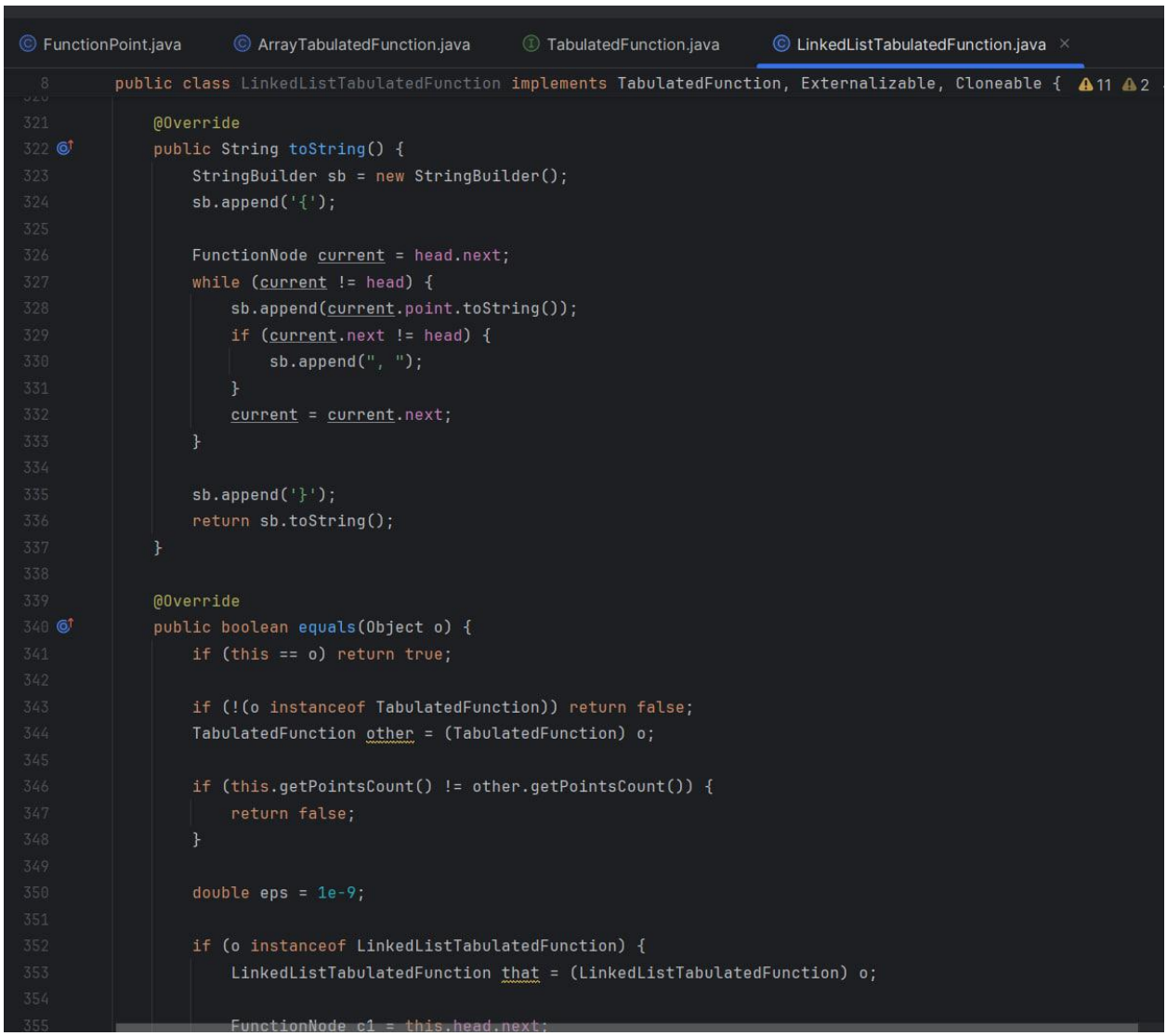
Задание 3

Аналогично, переопределил методы toString(), equals(), hashCode() и clone() в классе LinkedListTabulatedFunction. При написании методов учтел следующие особенности.

• Метод equals() также должен корректно работать при сравнении с любым объектом типа TabulatedFunction, а при сравнении с объектом типа LinkedListTabulatedFunction время работы метода должно быть сокращено за счёт возможности прямого обращения к полям переданного объекта.

• Клонирование в методе clone() тоже должно быть глубоким, однако классическое глубокое клонирование в данном случае не совсем разумно. Если сделать объекты класса FunctionNode клонируемыми, после их клонирования значения полей ссылок придётся изменить (т.к. они будут ссылаться на объекты из исходного списка), и значение ссылающегося на объект точки поля тоже придётся изменить (т.к. его нужно будет заменить клоном объекта точки).

Код 3.1


FunctionPoint.java
ArrayTabulatedFunction.java
TabulatedFunction.java
LinkedListTabulatedFunction.java
public class LinkedListTabulatedFunction implements TabulatedFunction, Externalizable, Cloneable {
@Override
public String toString() {
StringBuilder sb = new StringBuilder();
sb.append('{');
FunctionNode current = head.next;
while (current != head) {
sb.append(current.point.toString());
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.getPointsCount() != other.getPointsCount()) {
return false;
}
double eps = 1e-9;
if (o instanceof LinkedListTabulatedFunction) {
LinkedListTabulatedFunction that = (LinkedListTabulatedFunction) o;
FunctionNode c1 = this.head.next;

Код 3.2

```
340         public boolean equals(Object o) {
355             FunctionNode c1 = this.head.next;
356             FunctionNode c2 = that.head.next;
357
358             while (c1 != this.head && c2 != that.head) {
359                 double x1 = c1.point.getX();
360                 double y1 = c1.point.getY();
361                 double x2 = c2.point.getX();
362                 double y2 = c2.point.getY();
363
364                 if (Math.abs(x1 - x2) >= eps || Math.abs(y1 - y2) >= eps) {
365                     return false;
366                 }
367
368                 c1 = c1.next;
369                 c2 = c2.next;
370             }
371
372             return c1 == this.head && c2 == that.head;
373         } else {
374             FunctionNode current = this.head.next;
375             int i = 0;
376             while (current != this.head) {
377                 double x1 = current.point.getX();
378                 double y1 = current.point.getY();
379                 double x2 = other.getPointX(i);
380                 double y2 = other.getPointY(i);
381
382                 if (Math.abs(x1 - x2) >= eps || Math.abs(y1 - y2) >= eps) {
383                     return false;
384                 }
385
386                 current = current.next;
387                 i++;
388             }
```

Код 3.3

```
    @Override
    public int hashCode() {
        int h = pointsCount;

        FunctionNode current = head.next;
        while (current != head) {
            h ^= current.point.hashCode();
            current = current.next;
        }

        return h;
    }

    @Override
    public Object clone() {
        try {
            LinkedListTabulatedFunction copy = (LinkedListTabulatedFunction) super.clone();
            copy.initEmptyList();

            FunctionNode current = this.head.next;
            while (current != this.head) {
                FunctionPoint clonedPoint = (FunctionPoint) current.point.clone();
                FunctionNode node = copy.addNodeToTail();
                node.point = clonedPoint;
                current = current.next;
            }

            return copy;
        } catch (CloneNotSupportedException e) {
            LinkedListTabulatedFunction copy = new LinkedListTabulatedFunction();
            FunctionNode current = this.head.next;
            while (current != this.head) {
                FunctionNode node = copy.addNodeToTail();
                node.point = new FunctionPoint(current.point);
```

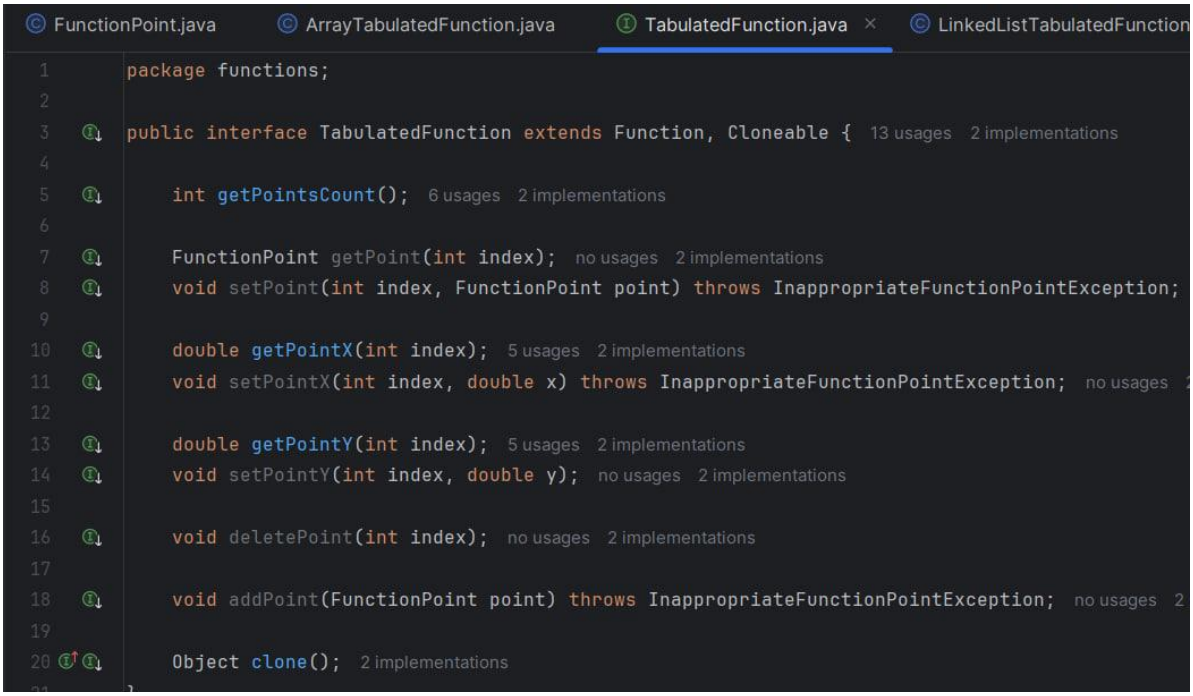
Код 3.4

```java
@Override
public Object clone() {
    try {
        LinkedListTabulatedFunction copy = (LinkedListTabulatedFunction) super.clone();
        copy.initEmptyList();

        FunctionNode current = this.head.next;
        while (current != this.head) {
            FunctionPoint clonedPoint = (FunctionPoint) current.point.clone();
            FunctionNode node = copy.addNodeToTail();
            node.point = clonedPoint;
            current = current.next;
        }

        return copy;
    } catch (CloneNotSupportedException e) {
        LinkedListTabulatedFunction copy = new LinkedListTabulatedFunction();
        FunctionNode current = this.head.next;
        while (current != this.head) {
            FunctionNode node = copy.addNodeToTail();
            node.point = new FunctionPoint(current.point);
            current = current.next;
        }
        return copy;
    }
}
```

Задание 4

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

Код 4



Задание 5

Проверил работу написанных методов.

• Проверил работу метода toString() для объектов типов ArrayTabulatedFunction и LinkedListTabulatedFunction, выведя строковое представление объектов в консоль.

• Проверил работу метода equals(), вызывая его для одинаковых и различающихся объектов одинаковых и различающихся классов.

• Проверил работу метода hashCode(), выведя в консоль его значения для всех использованных объектов. Убедился в согласованности работы методов equals() и hashCode(). Также попробовал незначительно изменить один из объектов (например, изменить одну из координат одной из точек на несколько тысячных) и проверьте, как изменится значение хэш-кода объекта.

• Проверил работу метода clone() для объектов обоих классов табулированных функций. Убедился, что произведено именно глубокое клонирование: для этого после клонирования измените исходные объекты и проверил, что объекты-клоны не изменились.

Код 5:

```
import functions.*;

import functions.basic.*;


public class Main {
```

```
public static void main(String[] args) throws Exception {
                Function sin = new Sin();

TabulatedFunction arr1 = TabulatedFunctions.tabulate(sin, 0.0,
                    Math.PI, 5);
TabulatedFunction arr2 = TabulatedFunctions.tabulate(sin, 0.0,
                    Math.PI, 5);
                TabulatedFunction list1;
                        {
                  FunctionPoint[] pts = new
        FunctionPoint[arr1.getPointsCount()];
             for (int i = 0; i < pts.length; i++) {
                    pts[i] = arr1.getPoint(i);
                          }
          list1 = new LinkedListTabulatedFunction(pts);
                        }

   System.out.println("toString ArrayTabulatedFunction:");
            System.out.println(arr1.toString());
 System.out.println("toString LinkedListTabulatedFunction:");
           System.out.println(list1.toString());

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

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

```
System.out.println("arr2.hashCode() = " + arr2.hashCode());
System.out.println("list1.hashCode() = " + list1.hashCode());

arr2.setPointY(0, arr2.getPointY(0) + 0.001);
System.out.println("После небольшого изменения arr2:");
System.out.println("arr1.equals(arr2) = " +
        arr1.equals(arr2));
System.out.println("arr2.hashCode() = " + arr2.hashCode());

TabulatedFunction arrClone = (TabulatedFunction) arr1.clone();
TabulatedFunction listClone = (TabulatedFunction)
        list1.clone();

System.out.println("clone:");
System.out.println("arr1.equals(arrClone) = " +
        arr1.equals(arrClone));
System.out.println("list1.equals(listClone) = " +
        list1.equals(listClone));

arr1.setPointY(1, arr1.getPointY(1) + 1.0);
list1.setPointY(1, list1.getPointY(1) + 1.0);

System.out.println("После изменения исходных объектов:");
System.out.println("arr1.equals(arrClone) = " +
        arr1.equals(arrClone));
System.out.println("list1.equals(listClone) = " +
        list1.equals(listClone));
    }
}
```

Вывод:

```
toString ArrayTabulatedFunction:
{(0.0; 0.0), (0.7853981633974483; 0.7071067811865475), (1.5707963267948966; 1.0), (2.356194490192345; 0.7071067811865476), (3.141592653589793; 1.2246467991473532E-16)}
toString LinkedListTabulatedFunction:
{(0.0; 0.0), (0.7853981633974483; 0.7071067811865475), (1.5707963267948966; 1.0), (2.356194490192345; 0.7071067811865476), (3.141592653589793; 1.2246467991473532E-16)}

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

hashCode:
arr1.hashCode() = 455675496
arr2.hashCode() = 455675496
list1.hashCode() = 455675496

После небольшого изменения arr2:
arr1.equals(arr2) = false
arr2.hashCode() = -158808615

clone:
arr1.equals(arrClone) = true
list1.equals(listClone) = true

После изменения исходных объектов:
arr1.equals(arrClone) = false
list1.equals(listClone) = false

Process finished with exit code 0
```