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 Луковникова Е.Д

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

Задание 1

Переопределила в классе FunctionPoint следующие методы.

• **String toString():** Возвращает текстовое описание точки.

• **boolean equals(Object o):** Возвращает 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():** Возвращать значение хэш-кода для объекта точки.

• **Object clone():** Возвращает объект-копию для объекта точки.

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


@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }
    if (o == null || getClass() != o.getClass()) {
        return false;
    }

    FunctionPoint that = (FunctionPoint) o;

    return Double.compare(that.x, x) == 0 && Double.compare(that.y, y) == 0;
}


@Override
public int hashCode() {
    long xBits = Double.doubleToLongBits(x);
    long yBits = Double.doubleToLongBits(y);

    int xHigh = (int)(xBits >>> 32);
    int xLow = (int)(xBits & 0xFFFFFFFFL);
    int yHigh = (int)(yBits >>> 32);
    int yLow = (int)(yBits & 0xFFFFFFFFL);

    return xHigh ^ xLow ^ yHigh ^ yLow;
}


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

Скрин 1

Задание 2

Переопределила в классе ArrayTabulatedFunction следующие методы.

• **String toString():** Возвращает описание табулированной функции.

• **boolean equals(Object o):** Возвращает 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 сокращено за счёт прямого обращения к элементам состояния переданного объекта.

• **int hashCode():** Возвращает значение хэш-кода для объекта табулированной функции.

• **Object clone():** Возвращает объект-копию для объекта табулированной функции. Поскольку табулированная функция ссылается на другие объекты, клонирование является глубоким.

```java
@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append(str: "{");

    for (int i = 0; i < pointsCount; i++) {
        if (i > 0) {
            sb.append(str: ", ");
        }
        sb.append(str: "(");
        sb.append(points[i].getX());
        sb.append(str: "; ");
        sb.append(points[i].getY());
        sb.append(str: ")");
    }

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

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

    if (!(o instanceof TabulatedFunction)) return false;

    TabulatedFunction otherFunc = (TabulatedFunction) o;

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

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

        for (int i = 0; i < pointsCount; i++) {
            if (!this.points[i].equals(otherArrayFunc.points[i])) {
                return false;
            }
        }
    } else {
        for (int i = 0; i < pointsCount; i++) {
            FunctionPoint thisPoint = this.getPoint(i);
            FunctionPoint otherPoint = otherFunc.getPoint(i);

            if (!thisPoint.equals(otherPoint)) {
                return false;
            }
        }
    }

    return true;
}
```

Скрин 2

```java
@Override
public int hashCode() {
    int hash = 0;

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

    hash ^= pointsCount;

    return hash;
}


@Override
public Object clone() throws CloneNotSupportedException {
ArrayTabulatedFunction cloned = (ArrayTabulatedFunction) super.clone();

cloned.points = new FunctionPoint[this.points.length];
cloned.pointsCount = this.pointsCount;

for (int i = 0; i < this.pointsCount; i++) {
    if (this.points[i] != null) {
        cloned.points[i] = new FunctionPoint(this.points[i]);
    }
}

for (int i = cloned.pointsCount; i < cloned.points.length; i++) {
    cloned.points[i] = null;
}

return cloned;
}
```

Скрин 3

Задание 3

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

```
@Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append(str: "{");

        for (int i = 0; i < pointsCount; i++) {
            if (i > 0) {
                sb.append(str: ", ");
            }
            FunctionPoint point = getPoint(i);
            sb.append(str: "(");
            sb.append(point.getX());
            sb.append(str: "; ");
            sb.append(point.getY());
            sb.append(str: ")");
        }

        sb.append(str: "}");
        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.pointsCount != other.getPointsCount()) return false;

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

            FunctionNode node1 = this.head.getNext();
            FunctionNode node2 = otherList.head.getNext();
            int count = 0;

            while (node1 != this.head && node2 != otherList.head && count < pointsCount) {
                if ((!node1.getPoint().equals(node2.getPoint()))) {
                    return false;
                }
                node1 = node1.getNext();
                node2 = node2.getNext();
                count++;
            }
            return true;
        } else {
            for (int i = 0; i < pointsCount; i++) {
                if (!this.getPoint(i).equals(other.getPoint(i))) {
                    return false;
                }
            }
            return true;
        }
    }
```

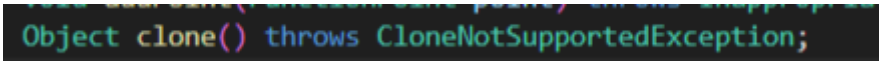
Скрин 4

```java
@Override
public int hashCode() {
    int hash = 0;

    FunctionNode current = head.getNext();
    int count = 0;

    while (current != head && count < pointsCount) {
        hash ^= current.getPoint().hashCode();
        current = current.getNext();
        count++;
    }

    hash ^= pointsCount;
    return hash;
}

@Override
public Object clone() throws CloneNotSupportedException {
LinkedListTabulatedFunction cloned = (LinkedListTabulatedFunction) super.clone();

FunctionPoint[] pointsArray = new FunctionPoint[pointsCount];

FunctionNode current = head.getNext();
int index = 0;

while (current != head && index < pointsCount) {
    pointsArray[index] = new FunctionPoint(current.getPoint());
    current = current.getNext();
    index++;
}

cloned.pointsCount = pointsCount;
cloned.head = new FunctionNode(point: null, pred: null, next: null);
cloned.head.setNext(cloned.head);
cloned.head.setPred(cloned.head);

for (FunctionPoint point : pointsArray) {
    cloned.addNodeToTail(new FunctionPoint(point));
}

return cloned;
}
```

Скрин 5

Задание 4

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



Скрин 6

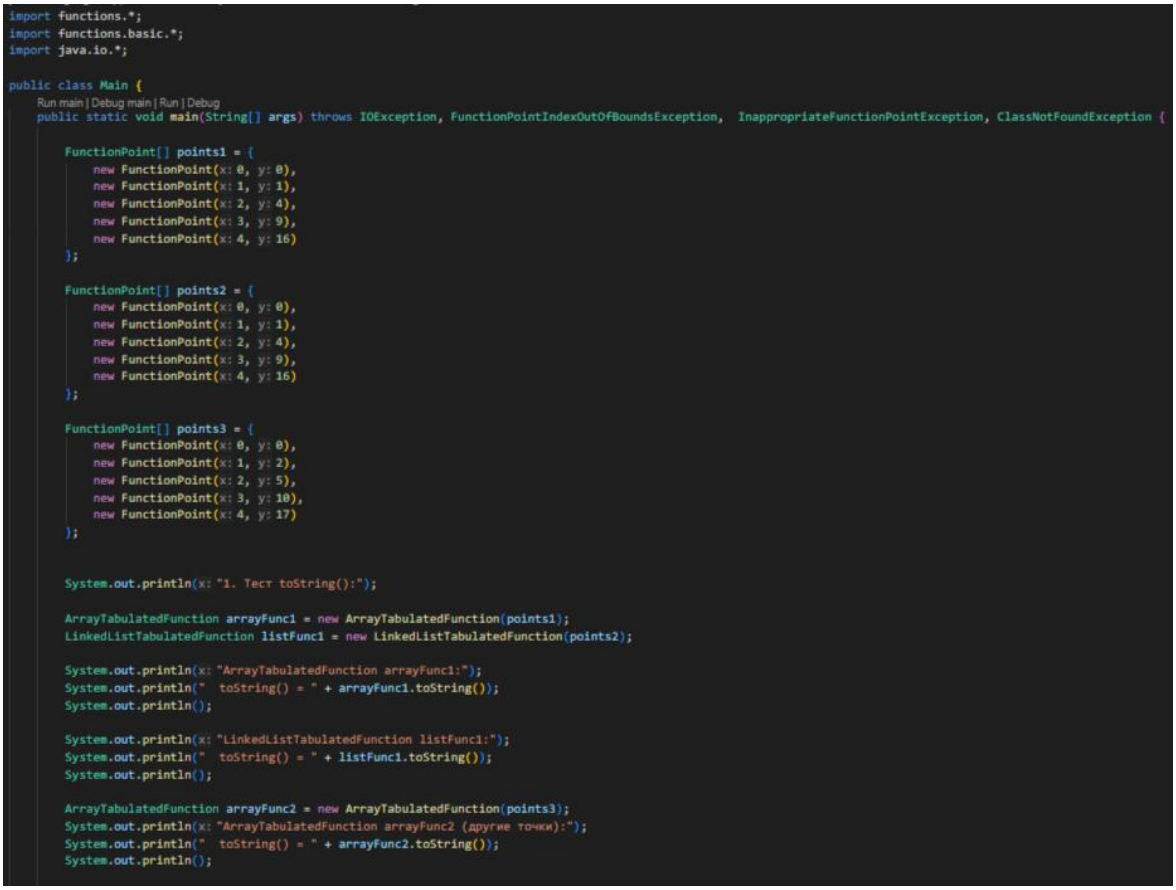
Задание 5

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

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

• Проверила работу метода hashCode(), выведя в консоль его значения для всех использованных объектов. Убедилась в согласованности работы методов equals() и hashCode(). Также незначительно изменила один из объектов и проверила, как изменится значение хэш-кода объекта.

• Проверила работу метода clone() для объектов обоих классов табулированных функций. Убедилась, что произведено именно глубокое клонирование: для этого после клонирования изменила исходные объекты и проверила, что объекты-клоны не изменились.



```
import functions.*;
import functions.basic.*;
import java.io.*;

public class Main {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) throws IOException, FunctionPointIndexOutOfBoundsException,  InappropriateFunctionPointException, ClassNotFoundException {

        FunctionPoint[] points1 = {
            new FunctionPoint(x: 0, y: 0),
            new FunctionPoint(x: 1, y: 1),
            new FunctionPoint(x: 2, y: 4),
            new FunctionPoint(x: 3, y: 9),
            new FunctionPoint(x: 4, y: 16)
        };

        FunctionPoint[] points2 = {
            new FunctionPoint(x: 0, y: 0),
            new FunctionPoint(x: 1, y: 1),
            new FunctionPoint(x: 2, y: 4),
            new FunctionPoint(x: 3, y: 9),
            new FunctionPoint(x: 4, y: 16)
        };

        FunctionPoint[] points3 = {
            new FunctionPoint(x: 0, y: 0),
            new FunctionPoint(x: 1, y: 2),
            new FunctionPoint(x: 2, y: 5),
            new FunctionPoint(x: 3, y: 10),
            new FunctionPoint(x: 4, y: 17)
        };


        System.out.println(x: "1. Тест toString():");

        ArrayTabulatedFunction arrayFunc1 = new ArrayTabulatedFunction(points1);
        LinkedListTabulatedFunction listFunc1 = new LinkedListTabulatedFunction(points2);

        System.out.println(x: "ArrayTabulatedFunction arrayFunc1:");
        System.out.println("  toString() = " + arrayFunc1.toString());
        System.out.println();

        System.out.println(x: "LinkedListTabulatedFunction listFunc1:");
        System.out.println("  toString() = " + listFunc1.toString());
        System.out.println();

        ArrayTabulatedFunction arrayFunc2 = new ArrayTabulatedFunction(points3);
        System.out.println(x: "ArrayTabulatedFunction arrayFunc2 (другие точки):");
        System.out.println("  toString() = " + arrayFunc2.toString());
        System.out.println();
```

Скрин 7

```
System.out.println(x: "\n2. Тест equals():");

System.out.println("  arrayFunc1.equals(listFunc1) = " + arrayFunc1.equals(listFunc1));
System.out.println("  listFunc1.equals(arrayFunc1) = " + listFunc1.equals(arrayFunc1));
System.out.println();

System.out.println("  arrayFunc1.equals(arrayFunc2) = " + arrayFunc1.equals(arrayFunc2));
System.out.println("  arrayFunc2.equals(arrayFunc1) = " + arrayFunc2.equals(arrayFunc1));
System.out.println();

System.out.println("  arrayFunc1.equals(arrayFunc1) = " + arrayFunc1.equals(arrayFunc1));
System.out.println("  listFunc1.equals(listFunc1) = " + listFunc1.equals(listFunc1));
System.out.println();

System.out.println(x: "\n3. Тест hashCode():");


System.out.println("  arrayFunc1.hashCode() = " + arrayFunc1.hashCode());
System.out.println("  listFunc1.hashCode() = " + listFunc1.hashCode());
System.out.println("  arrayFunc2.hashCode() = " + arrayFunc2.hashCode());
System.out.println();

System.out.println("  arrayFunc1.equals(listFunc1) = " + arrayFunc1.equals(listFunc1));
System.out.println("  arrayFunc1.hashCode() == listFunc1.hashCode() = " + (arrayFunc1.hashCode() == listFunc1.hashCode()));
System.out.println();

System.out.println(x: "Изменение функции и проверка изменения hashCode:");
ArrayTabulatedFunction testFunc = new ArrayTabulatedFunction(points1);
System.out.println("  Исходный hashCode = " + testFunc.hashCode());

testFunc.setPointY(index: 2, y: 4.001);
System.out.println(x: "  После изменения точки (2, 4) на (2, 4.001):");
System.out.println("  Новый hashCode = " + testFunc.hashCode());
System.out.println("  Сравнение с исходным hashCode " + (testFunc.hashCode() == arrayFunc1.hashCode()));
System.out.println("  Результат сравнения с исходным объектом " + testFunc.equals(arrayFunc1));
System.out.println();
```

Скрин 8

```
System.out.println(x: "\n4. Тест clone():");

ArrayTabulatedFunction arrayOriginal = new ArrayTabulatedFunction(points1);
ArrayTabulatedFunction arrayClone = null;
try {
    arrayClone = (ArrayTabulatedFunction) arrayOriginal.clone();
} catch (CloneNotSupportedException e) {
    System.out.println("  Ошибка клонирования: " + e.getMessage());
}

if (arrayClone != null) {
    System.out.println("  Исходная функция: " + arrayOriginal);
    System.out.println("  Клонированная функция: " + arrayClone);
    System.out.println("  Одинаковые после клонирования? = " + arrayOriginal.equals(arrayClone));
    System.out.println("  Это один и тот же объект? " + (arrayOriginal == arrayClone));
    System.out.println();

    System.out.println(x: "Проверка глубокого клонирования ArrayTabulatedFunction:");
    arrayOriginal.setPointY(index: 3, y: 100);
    System.out.println("  Исходная после изменения: " + arrayOriginal);
    System.out.println("  Клон после изменения: " + arrayClone);
    System.out.println("  Одинаковые ли после изменения = " + arrayOriginal.equals(arrayClone));
    System.out.println();
}

System.out.println(x: "Тестирование clone() для LinkedListTabulatedFunction:");
LinkedListTabulatedFunction listOriginal = new LinkedListTabulatedFunction(points2);
LinkedListTabulatedFunction listClone = null;
try {
    listClone = (LinkedListTabulatedFunction) listOriginal.clone();
} catch (CloneNotSupportedException e) {
    System.out.println("  Ошибка клонирования: " + e.getMessage());
}

if (listClone != null) {
    System.out.println("  Исходная функция: " + listOriginal);
    System.out.println("  Клонированная функция: " + listClone);
    System.out.println("  Одинаковые после клонирования? " + listOriginal.equals(listClone));
    System.out.println("  Это один и тот же объект? " + (listOriginal == listClone));
    System.out.println();

    System.out.println(x: "Проверка глубокого клонирования LinkedListTabulatedFunction:");
    listOriginal.setPointY(index: 1, y: 200);
    System.out.println("  Исходная после изменения: " + listOriginal);
    System.out.println("  Клон после изменения: " + listClone);
    System.out.println("  Одинаковые ли после изменения = " + listOriginal.equals(listClone));
    System.out.println();
}
```

Скрин 9

```
System.out.println(x: "\n5. Тест с реальными функциями:");

Function sinFunc = new Sin();
Function cosFunc = new Cos();
double pi = Math.PI;

TabulatedFunction tabulatedSin = TabulatedFunctions.tabulate(sinFunc, leftX: 0, pi, pointsCount: 20);
TabulatedFunction tabulatedCos = TabulatedFunctions.tabulate(cosFunc, leftX: 0, pi, pointsCount: 20);

System.out.println(x: "Табулированные sin и cos:");
System.out.println("  Табулированный sin: " + tabulatedSin);
System.out.println("  Табулированный cos: " + tabulatedCos);
System.out.println();

TabulatedFunction clonedSin = null;
try {
    clonedSin = (TabulatedFunction) tabulatedSin.clone();
} catch (CloneNotSupportedException e) {
    System.out.println("  Ошибка клонирования sin: " + e.getMessage());
}

if (clonedSin != null) {
    System.out.println("Клонированный sin: " + clonedSin);
    System.out.println("  Равенство с исходным? " + tabulatedSin.equals(clonedSin));


    if (tabulatedSin instanceof ArrayTabulatedFunction) {
        ((ArrayTabulatedFunction)tabulatedSin).setPointY(index: 10, y: 999);
    } else if (tabulatedSin instanceof LinkedListTabulatedFunction) {
        ((LinkedListTabulatedFunction)tabulatedSin).setPointY(index: 10, y: 999);
    }

    System.out.println(x: "  После изменения исходного в точке 10:");
    System.out.println("    Исходный sin[10] = " + tabulatedSin.getPointY(index: 10));
    System.out.println("    Клон sin[10] = " + clonedSin.getPointY(index: 10));
    System.out.println("    Равны? " + tabulatedSin.equals(clonedSin));
}
```

Скрин 10

```
System.out.println(x: "\n6. Тест с разным количеством точек:");

TabulatedFunction sin10 = TabulatedFunctions.tabulate(sinFunc, leftX: 0, pi, pointsCount: 10);
TabulatedFunction sin20 = TabulatedFunctions.tabulate(sinFunc, leftX: 0, pi, pointsCount: 20);
TabulatedFunction sin10Clone = null;

try {
    sin10Clone = (TabulatedFunction) sin10.clone();
} catch (CloneNotSupportedException e) {
    System.out.println("  Ошибка клонирования sin10: " + e.getMessage());
}

if (sin10Clone != null) {
    System.out.println("sin с 10 точками: " + sin10);
    System.out.println("sin с 20 точками: " + sin20);
    System.out.println("Клон sin с 10 точками: " + sin10Clone);
    System.out.println();

    System.out.println("  sin10.equals(sin20) = " + sin10.equals(sin20));
    System.out.println("  sin10.hashCode() == sin20.hashCode() = " + (sin10.hashCode() == sin20.hashCode()));
}
```

Скрин 11

Результаты работы

1. Тест toString():

ArrayTabulatedFunction arrayFunc1:

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

LinkedListTabulatedFunction listFunc1:

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

ArrayTabulatedFunction arrayFunc2 (другие точки):

toString() = {(0.0; 0.0), (1.0; 2.0), (2.0; 5.0), (3.0; 10.0), (4.0; 17.0)}

2. Тест equals():

arrayFunc1.equals(listFunc1) = true

listFunc1.equals(arrayFunc1) = true

arrayFunc1.equals(arrayFunc2) = false

arrayFunc2.equals(arrayFunc1) = false

arrayFunc1.equals(arrayFunc1) = true

listFunc1.equals(listFunc1) = true

3. Тест hashCode():

arrayFunc1.hashCode() = 1703941

listFunc1.hashCode() = 1703941

arrayFunc2.hashCode() = 2145976325

arrayFunc1.equals(listFunc1) = true

arrayFunc1.hashCode() == listFunc1.hashCode() = true

Изменение функции и проверка изменения hashCode:

Исходный hashCode = 1703941

После изменения точки (2, 4) на (2, 4.001):

Новый hashCode = 617033240

Сравнение с исходным hashCode false

Результат сравнения с исходным объектом false

4. Тест clone():

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

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

Одинаковые после клонирования? = true

Это один и тот же объект? false

Проверка глубокого клонирования ArrayTabulatedFunction:

Исходная после изменения: {(0.0; 0.0), (1.0; 1.0), (2.0; 4.0), (3.0; 100.0), (4.0; 16.0)}

Клон после изменения: {(0.0; 0.0), (1.0; 1.0), (2.0; 4.0), (3.0; 9.0), (4.0; 16.0)}

Одинаковые ли после изменения = false

Тестирование clone() для LinkedListTabulatedFunction:

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

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

Одинаковые после клонирования? true

Это один и тот же объект? false

Проверка глубокого клонирования LinkedListTabulatedFunction:

Исходная после изменения: {(0.0; 0.0), (1.0; 200.0), (2.0; 4.0), (3.0; 9.0), (4.0; 16.0)}

Клон после изменения: {(0.0; 0.0), (1.0; 1.0), (2.0; 4.0), (3.0; 9.0), (4.0; 16.0)}

Одинаковые ли после изменения = false

5. Тест с реальными функциями:

Табулированные sin и cos:

Табулированный sin: {(0.0; 0.0), (0.16534698176788384; 0.1645945902807339), (0.3306939635357677; 0.32469946920468346), (0.4960409453036515; 0.4759473930370735), (0.6613879270715354; 0.6142127126896678), (0.8267349088394192; 0.7357239106731316), (0.992081890607303; 0.8371664782625285), (1.1574288723751869; 0.9157733266550574), (1.3227758541430708; 0.9694002659393304), (1.4881228359109546; 0.9965844930066698), (1.6534698176788385; 0.9965844930066698), (1.8188167994467224; 0.9694002659393304), (1.984163781214606; 0.9157733266550575), (2.14951076298249; 0.8371664782625287), (2.3148577447503738; 0.7357239106731318), (2.4802047265182576; 0.6142127126896679), (2.6455517082861415; 0.4759473930370737), (2.8108986900540254; 0.3246994692046836), (2.9762456718219092; 0.16459459028073403), (3.141592653589793; 1.2246467991473532E-16)}

Табулированный cos: {(0.0; 1.0), (0.16534698176788384; 0.9863613034027223), (0.3306939635357677; 0.9458172417006346), (0.4960409453036515; 0.8794737512064891), (0.6613879270715354; 0.7891405093963936), (0.8267349088394192; 0.6772815716257411), (0.992081890607303;

0.546948158122427), (1.1574288723751869; 0.40169542465296953), (1.3227758541430708; 0.24548548714079924), (1.4881228359109546; 0.0825793454723324), (1.6534698176788385; -0.08257934547233227), (1.8188167994467224; -0.24548548714079912), (1.984163781214606; -0.40169542465296926), (2.14951076298249; -0.5469481581224267), (2.3148577447503738; -0.6772815716257409), (2.4802047265182576; -0.7891405093963935), (2.6455517082861415; -0.879473751206489), (2.8108986900540254; -0.9458172417006346), (2.9762456718219092; -0.9863613034027223), (3.141592653589793; -1.0)}

Клонированный sin: {(0.0; 0.0), (0.16534698176788384; 0.1645945902807339), (0.3306939635357677; 0.32469946920468346), (0.4960409453036515; 0.4759473930370735), (0.6613879270715354; 0.6142127126896678), (0.8267349088394192; 0.7357239106731316), (0.992081890607303; 0.8371664782625285), (1.1574288723751869; 0.9157733266550574), (1.3227758541430708; 0.9694002659393304), (1.4881228359109546; 0.9965844930066698), (1.6534698176788385; 0.9965844930066698), (1.8188167994467224; 0.9694002659393304), (1.984163781214606; 0.9157733266550575), (2.14951076298249; 0.8371664782625287), (2.3148577447503738; 0.7357239106731318), (2.4802047265182576; 0.6142127126896679), (2.6455517082861415; 0.4759473930370737), (2.8108986900540254; 0.3246994692046836), (2.9762456718219092; 0.16459459028073403), (3.141592653589793; 1.2246467991473532E-16)}

Равенство с исходным? true

После изменения исходного в точке 10:

Исходный sin[10] = 999.0

Клон sin[10] = 0.9965844930066698

Равны? false

6. Тест с разным количеством точек:

sin с 10 точками: {(0.0; 0.0), (0.3490658503988659; 0.3420201433256687), (0.6981317007977318; 0.6427876096865393), (1.0471975511965976; 0.8660254037844386), (1.3962634015954636; 0.984807753012208), (1.7453292519943295; 0.984807753012208), (2.0943951023931953; 0.8660254037844387), (2.443460952792061; 0.6427876096865395), (2.792526803190927; 0.3420201433256689), (3.141592653589793; 1.2246467991473532E-16)}

sin с 20 точками: {(0.0; 0.0), (0.16534698176788384; 0.1645945902807339), (0.3306939635357677; 0.32469946920468346), (0.4960409453036515; 0.4759473930370735), (0.6613879270715354; 0.6142127126896678), (0.8267349088394192; 0.7357239106731316), (0.992081890607303; 0.8371664782625285), (1.1574288723751869; 0.9157733266550574), (1.3227758541430708; 0.9694002659393304), (1.4881228359109546; 0.9965844930066698), (1.6534698176788385; 0.9965844930066698), (1.8188167994467224; 0.9694002659393304), (1.984163781214606; 0.9157733266550575), (2.14951076298249; 0.8371664782625287), (2.3148577447503738; 0.7357239106731318), (2.4802047265182576; 0.6142127126896679), (2.6455517082861415; 0.4759473930370737), (2.8108986900540254; 0.3246994692046836), (2.9762456718219092; 0.16459459028073403), (3.141592653589793; 1.2246467991473532E-16)}

Клон sin с 10 точками: {(0.0; 0.0), (0.3490658503988659; 0.3420201433256687), (0.6981317007977318; 0.6427876096865393), (1.0471975511965976; 0.8660254037844386), (1.3962634015954636; 0.984807753012208), (1.7453292519943295; 0.984807753012208), (2.0943951023931953; 0.8660254037844387), (2.443460952792061; 0.6427876096865395), (2.792526803190927; 0.3420201433256689), (3.141592653589793; 1.2246467991473532E-16)}

sin10.equals(sin20) = false

sin10.hashCode() == sin20.hashCode() = false