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 НОМЕР 5

Студент Горейнов Д.В.

Группа 6301-030301D

Проверил Борисов.Д.С

Оценка _____

САМАРА 2025

Задание 1

Переопредели в классе FunctionPoint следующие методы.

String toString():Возвращать текстовое описание точки.

```
@Override

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

Фото 1

В данном методе используется конкатенация для вывода текстового описания 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.

```
@Override

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

    FunctionPoint that = (FunctionPoint) o;

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

}
```

Фото 2

Первое условие проверяет, указывают ли обе ссылки на один и тот же объект в памяти
Второе условие проверяет является ли переданный объект null и являются ли классы объектов разными

int hashCode():Возвращает значение хэш-кода для объекта точки.

```
    @Override
public int hashCode() {
    return Objects.hash(x, y);
}
```

Фото 3

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

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

Фото 4

Задание 2

Переопределил в классе ArrayTabulatedFunction следующие методы.

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

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

}
```

Фото 5

• **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 будет сокращено за счёт прямого обращения к элементам состояния переданного объекта.

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

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

        //ArrayTabulatedFunction
        if (o instanceof ArrayTabulatedFunction) {
            ArrayTabulatedFunction arrayOther = (ArrayTabulatedFunction) o;
            if (this.pointsCount != arrayOther.pointsCount) return false;
            for (int i = 0; i < pointsCount; i++) {
                if (!this.points[i].equals(arrayOther.points[i])) return false;
            }
            return true;
        }

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

        for (int i = 0; i < pointsCount; i++) {
            // Сравниваем координаты x и y через getX и getY
            if (this.getPointX(i) != other.getPointX(i) ||
                this.getPointY(i) != other.getPointY(i)) {
                return false;
            }
        }
        return true;
    }

    return false;
}
```

Фото 6

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

```java
@Override
public int hashCode() {
    int hash = pointsCount; // Включаем количество точек в хэш

    // Комбинируем хэш-коды всех точек
    for (int i = 0; i < pointsCount; i++) {
        hash ^= points[i].hashCode();
    }

    return hash;
}
```

Фото 7

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

```java
@Override
public TabulatedFunction clone()
{
    ArrayTabulatedFunction cloned= new ArrayTabulatedFunction();
    cloned.pointsCount=this.pointsCount;
    cloned.points=new FunctionPoint[this.points.length];

    for(int i=0;i<cloned.pointsCount;i++)
    {
        cloned.points[i]=(FunctionPoint)this.points[i].clone();
    }
    return cloned;

}
```

Фото 8

Задание 3

Аналогично, переопределил методы toString(), equals(), hashCode() и clone() в классе LinkedListTabulatedFunction:

toString()

```java
@Override
public String toString()
{
    StringBuilder list_num =new StringBuilder();
    list_num.append("{");
    FunctionNode current = head.getNext();
    while (current!=head)
    {
        list_num.append(current.getPoint().toString());
        if(current!=head)
            {
            list_num.append(",");
            }
            current=current.getNext();

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


}
```

Фото 9

equals()

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

    if(o==null)
    {
        return false;
    }

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

        if(this.pointsCount!=one.pointsCount)
        {
            return false;
        }

        FunctionNode current=this.head.getNext();
        FunctionNode one_current=one.head.getNext();

        while (current!=this.head && one_current != one.head)
        {
            if(!current.getPoint().equals(one_current.getPoint()))
            {
                return false;

            }
            current=current.getNext();
            one_current=current.getNext();
        }
     return true;
    }
```

Фото 10

```
else if (o instanceof TabulatedFunction)
{
    TabulatedFunction one = (TabulatedFunction) o;

    if (this.getPointsCount() != one.getPointsCount()) return false;

        // Сравниваем все точки по их координатам X и Y
    for (int i = 0; i < this.getPointsCount(); i++) {
    double thisX = this.getPointX(i);
    double thisY = this.getPointY(i);
    double oneX = one.getPointX(i);
    double oneY = one.getPointY(i);

            if (Math.abs(thisX - oneX) > 1e-9 || Math.abs(thisY - oneY) > 1e-9) {
                return false;
            }
        }

        return true;
    }

    return false;
}
```

Фото 11

hashCode()

```
@Override
public int hashCode() {
    int result = pointsCount;
    for (int i = 0; i < pointsCount; ++i) {
        result ^= getPoint(i).hashCode();
    }
    return result;
}
```

Фото 12

clone()

```java
@Override
public TabulatedFunction clone() {
    try {
        LinkedListTabulatedFunction cloned = (LinkedListTabulatedFunction) super.clone();

        // Создаем новую голову для клонированного списка
        cloned.head = new FunctionNode();
        cloned.head.setNext(cloned.head);
        cloned.head.setPrev(cloned.head);
        cloned.pointsCount = 0;

        // "Пересобираем" список, копируя точки из исходного
        FunctionNode current = this.head.getNext();
        while (current != this.head) {
            FunctionPoint pointCopy = new FunctionPoint(current.getPoint().getX(), current.getPoint().getY());
            cloned.addNodeToTail().setPoint(pointCopy);
            current = current.getNext();
        }

        return cloned;
    } catch (CloneNotSupportedException e) {
        throw new AssertionError("Клонирование не поддерживается", e);
    }
}
```

Фото 12

Задание 4

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

```java
package functions;

public interface TabulatedFunction extends Function, Cloneable {
    int getPointsCount();
    void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException;
    double getPointX(int index);
    void setPointX(int index, double x) throws InappropriateFunctionPointException;
    double getPointY(int index);
    void setPointY(int index, double y);
    void deletePoint(int index);
    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;
    TabulatedFunction clone();
}
```

Фото 13

```java
@Override
public TabulatedFunction clone() {
```

Фото 14

Задание 5

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

public class Main {
    Run | Debug
    public static void main(String[] args) {
        System.out.println("ТЕСТИРОВАНИЕ МЕТОДОВ");

        try {
            // Создаем тестовые данные
            FunctionPoint[] points1 = { new FunctionPoint(x: 0, y: 0),new FunctionPoint(x: 1, y: 1), new FunctionPoint(x: 2, y: 4),new FunctionPoint(x:

            FunctionPoint[] points2 = { new FunctionPoint(x: 0, y: 0),new FunctionPoint(x: 1, y: 1),new FunctionPoint(x: 2, y: 4),new FunctionPoint(x:

            FunctionPoint[] points3 = {new FunctionPoint(x: 0, y: 0),new FunctionPoint(x: 1, y: 2), new FunctionPoint(x: 2, y: 4), new FunctionPoint(x:

            //ArrayTabulatedFunction
            System.out.println("\n Тестирование ArrayTabulatedFunction");
            ArrayTabulatedFunction arrayFunc1 = new ArrayTabulatedFunction(points1);
            ArrayTabulatedFunction arrayFunc2 = new ArrayTabulatedFunction(points2);
            ArrayTabulatedFunction arrayFunc3 = new ArrayTabulatedFunction(points3);
            //toString()
            System.out.println("arrayFunc1.toString(): " + arrayFunc1.toString());
            System.out.println("arrayFunc2.toString(): " + arrayFunc2.toString());
            System.out.println("arrayFunc3.toString(): " + arrayFunc3.toString());
            //equals()
            System.out.println("\n Тестирование equals()");
            System.out.println("arrayFunc1.equals(arrayFunc2): " + arrayFunc1.equals(arrayFunc2));
            System.out.println("arrayFunc1.equals(arrayFunc3): " + arrayFunc1.equals(arrayFunc3));
            System.out.println("arrayFunc1.equals(null): " + arrayFunc1.equals(o: null));
            System.out.println("arrayFunc1.equals(\"строка\"): " + arrayFunc1.equals(o: "ыаааууцс"));
            //hashCode()
            System.out.println("\n Тестирование hashCode()");
            System.out.println("arrayFunc1.hashCode(): " + arrayFunc1.hashCode());
            System.out.println("arrayFunc2.hashCode(): " + arrayFunc2.hashCode());
            System.out.println("arrayFunc3.hashCode(): " + arrayFunc3.hashCode());
            System.out.println("arrayFunc1.hashCode() == arrayFunc2.hashCode(): " + (arrayFunc1.hashCode() == arrayFunc2.hashCode()));
            //изменения объекта
            System.out.println("\n Тестирование изменения объекта");
            double originalY = arrayFunc1.getPointY(index: 1);
            System.out.println("Исходный Y в точке 1: " + originalY);
            System.out.println("Исходный hashCode: " + arrayFunc1.hashCode());
            arrayFunc1.setPointY(index: 1, originalY + 0.001);
            System.out.println("Измененный Y в точке 1: " + arrayFunc1.getPointY(index: 1));
            System.out.println("Новый hashCode: " + arrayFunc1.hashCode());
            System.out.println("Хеш-коды различаются: " + (arrayFunc1.hashCode() != arrayFunc2.hashCode()));
            // Возвращаем исходное значение
            arrayFunc1.setPointY(index: 1, originalY);
```

Фото 15

```java
            //LinkedListTabulatedFunction
            System.out.println("\n Тестирование LinkedListTabulatedFunction");
            LinkedListTabulatedFunction linkedFunc1 = new LinkedListTabulatedFunction(points1);
            LinkedListTabulatedFunction linkedFunc2 = new LinkedListTabulatedFunction(points2);
            System.out.println("linkedFunc1.toString(): " + linkedFunc1.toString());
            System.out.println("linkedFunc2.toString(): " + linkedFunc2.toString());
            System.out.println("linkedFunc1.equals(linkedFunc2): " + linkedFunc1.equals(linkedFunc2));
            System.out.println("linkedFunc1.equals(linkedFunc1): " + linkedFunc1.equals(linkedFunc1));
            System.out.println("linkedFunc1.hashCode(): " + linkedFunc1.hashCode());
            System.out.println("linkedFunc2.hashCode(): " + linkedFunc2.hashCode());
            System.out.println("Согласованность equals/hashCode: " + (linkedFunc1.equals(linkedFunc2) == (linkedFunc1.hashCode() == linkedFunc2.hashCo
            // Тестирование сравнения между классами
            System.out.println("\n Тестирование сравнения между классами");
            System.out.println("arrayFunc1.equals(linkedFunc1): " + arrayFunc1.equals(linkedFunc1));
            System.out.println("linkedFunc1.equals(arrayFunc1): " + linkedFunc1.equals(arrayFunc1));
            System.out.println("arrayFunc1.hashCode(): " + arrayFunc1.hashCode());
            System.out.println("linkedFunc1.hashCode(): " + linkedFunc1.hashCode());
            System.out.println("Хеш-коды совпадают: " + (arrayFunc1.hashCode() == linkedFunc1.hashCode()));
            // Тестирование клонирования
            System.out.println("\n Тестирование глубокого клонирования");
            // ArrayTabulatedFunction
            System.out.println("\n ArrayTabulatedFunction");
            ArrayTabulatedFunction originalArray = new ArrayTabulatedFunction(points1);
            ArrayTabulatedFunction clonedArray = (ArrayTabulatedFunction) originalArray.clone();
            System.out.println("Оригинал (до изменения): " + originalArray.toString());
            System.out.println("Клон (до изменения): " + clonedArray.toString());
            // Изменяем оригинал
            try {
                originalArray.setPointY(index: 1, y: 100);
                originalArray.setPointX(index: 2, x: 2.5);
            } catch (InappropriateFunctionPointException e) {
                System.out.println("Ошибка при изменении точки: " + e.getMessage());
            }
            System.out.println("Оригинал (после изменения): " + originalArray.toString());
            System.out.println("Клон (после изменения оригинала): " + clonedArray.toString());
            System.out.println("Клон не изменился: " + (clonedArray.getPointY(index: 1) == 1 && clonedArray.getPointX(index: 2) == 2));
            // LinkedListTabulatedFunction
            System.out.println("\n LinkedListTabulatedFunction");
            LinkedListTabulatedFunction originalLinkedList = new LinkedListTabulatedFunction(points1);
            LinkedListTabulatedFunction clonedLinkedList = (LinkedListTabulatedFunction) originalLinkedList.clone();

            System.out.println("Оригинал (до изменения): " + originalLinkedList.toString());
            System.out.println("Клон (до изменения): " + clonedLinkedList.toString());
```

Фото 16

```
        // Изменяем оригинал
        try {
            originalLinkedList.setPointY(index: 1, y: 200);
            originalLinkedList.setPointX(index: 2, x: 3.5);
        } catch (InappropriateFunctionPointException e) {
            System.out.println("Ошибка при изменении точки: " + e.getMessage());
        }
        System.out.println("Оригинал (после изменения): " + originalLinkedList.toString());
        System.out.println("Клон (после изменения оригинала): " + clonedLinkedList.toString());
        System.out.println("Клон не изменился: " + (clonedLinkedList.getPointY(index: 1) == 1 && clonedLinkedList.getPointX(index: 2) == 2));
        // Проверка ссылочной независимости
        System.out.println("\n Проверка ссылочной независимости");
        System.out.println("originalArray == clonedArray: " + (originalArray == clonedArray));
        System.out.println("originalLinkedList == clonedLinkedList: " + (originalLinkedList == clonedLinkedList));
        System.out.println("originalArray.equals(clonedArray) после изменений: " + originalArray.equals(clonedArray));
        System.out.println("originalLinkedList.equals(clonedLinkedList) после изменений: " + originalLinkedList.equals(clonedLinkedList));
        // Дополнительная проверка FunctionPoint.clone()
        System.out.println("\n Проверка FunctionPoint.clone()");
        FunctionPoint originalPoint = new FunctionPoint(x: 5, y: 10);
        FunctionPoint clonedPoint = (FunctionPoint) originalPoint.clone();
        System.out.println("Original point: " + originalPoint);
        System.out.println("Cloned point: " + clonedPoint);
        System.out.println("Points are equal: " + originalPoint.equals(clonedPoint));
        System.out.println("Points are same object: " + (originalPoint == clonedPoint));

    } catch (Exception e) {
        System.out.println("Произошла ошибка: " + e.getMessage());
        e.printStackTrace();
    }
}
```

Фото 17

```
 Тестирование ArrayTabulatedFunction
arrayFunc1.toString(): {(0.0;0.0),(1.0;1.0),(2.0;4.0),(3.0;9.0)}
arrayFunc2.toString(): {(0.0;0.0),(1.0;1.0),(2.0;4.0),(3.0;9.0)}
arrayFunc3.toString(): {(0.0;0.0),(1.0;2.0),(2.0;4.0),(3.0;9.0)}

 Тестирование equals()
arrayFunc1.equals(arrayFunc2): true
arrayFunc1.equals(arrayFunc3): false
arrayFunc1.equals(null): false
arrayFunc1.equals("строка"): false

 Тестирование hashCode()
arrayFunc1.hashCode(): -16121852
arrayFunc2.hashCode(): -16121852
arrayFunc3.hashCode(): -15073276
arrayFunc1.hashCode() == arrayFunc2.hashCode(): true

 Тестирование изменения объекта
Исходный Y в точке 1: 1.0
Исходный hashCode: -16121852
Измененный Y в точке 1: 1.001
Новый hashCode: 1805565942
Хеш-коды различаются: true

 Тестирование LinkedListTabulatedFunction
linkedFunc1.toString(): {(0.0;0.0),(1.0;1.0),(2.0;4.0),(3.0;9.0),}
linkedFunc2.toString(): {(0.0;0.0),(1.0;1.0),(2.0;4.0),(3.0;9.0),}
linkedFunc1.equals(linkedFunc2): false
linkedFunc1.equals(linkedFunc1): true
linkedFunc1.hashCode(): -16121852
linkedFunc2.hashCode(): -16121852
Согласованность equals/hashCode: false

 Тестирование сравнения между классами
arrayFunc1.equals(linkedFunc1): true
linkedFunc1.equals(arrayFunc1): true
arrayFunc1.hashCode(): -16121852
linkedFunc1.hashCode(): -16121852
Хеш-коды совпадают: true
```

Фото 18

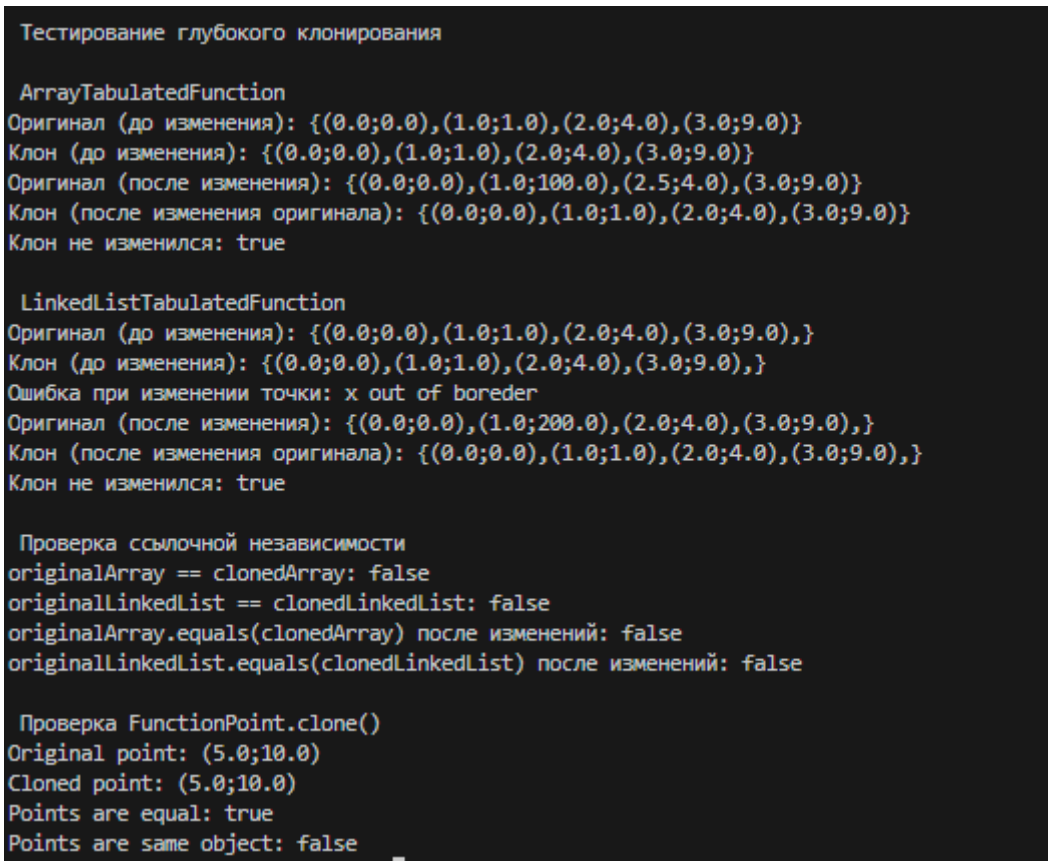
Тестирование глубокого клонирования
ArrayTabulatedFunction
Оригинал (до изменения): {(0.0;0.0),(1.0;1.0),(2.0;4.0),(3.0;9.0)}
Клон (до изменения): {(0.0;0.0),(1.0;1.0),(2.0;4.0),(3.0;9.0)}
Оригинал (после изменения): {(0.0;0.0),(1.0;100.0),(2.5;4.0),(3.0;9.0)}
Клон (после изменения оригинала): {(0.0;0.0),(1.0;1.0),(2.0;4.0),(3.0;9.0)}
Клон не изменился: true
LinkedListTabulatedFunction
Оригинал (до изменения): {(0.0;0.0),(1.0;1.0),(2.0;4.0),(3.0;9.0),}
Клон (до изменения): {(0.0;0.0),(1.0;1.0),(2.0;4.0),(3.0;9.0),}
Ошибка при изменении точки: x out of boreder
Оригинал (после изменения): {(0.0;0.0),(1.0;200.0),(2.0;4.0),(3.0;9.0),}
Клон (после изменения оригинала): {(0.0;0.0),(1.0;1.0),(2.0;4.0),(3.0;9.0),}
Клон не изменился: true
Проверка ссылочной независимости
originalArray == clonedArray: false
originalLinkedList == clonedLinkedList: false
originalArray.equals(clonedArray) после изменений: false
originalLinkedList.equals(clonedLinkedList) после изменений: false
Проверка FunctionPoint.clone()
Original point: (5.0;10.0)
Cloned point: (5.0;10.0)
Points are equal: true
Points are same object: false

Фото 19