Лабораторная работа №5. ООП.

Выполнил лабораторную работу:
Гордеев Максим Дмитриевич
Группа: 6204-010302D

Цель лабораторной работы: Расширить возможности классов, связанных с табулированными функциями, переопределив в них методы, унаследованные из класса **Object**.

Ход выполнения лабораторной работы

Задание 1:

Переопределим в классе **FunctionPoint** следующие методы:

- toString()
- equals()
- hashCode()
- clone()

```
49      @Override
50      public String toString(){
51          return String.format(format: "(%.3f, %.3f)", coorX, coorY);
52      }
53
54      @Override
55      public boolean equals(Object a){
56          if (this == a){
57              return true;
58          }
59
60          if (!(a instanceof FunctionPoint) || a == null){
61              return false;
62          }
63
64          FunctionPoint b = (FunctionPoint) a;
65
66          if (equal(coorX, b.getCoorX()) && equal(coorY, b.getCoorY())){
67              return true;
68          }else{
69              return false;
70          }
71
72      }
```

```java
    @Override
    public int hashCode() {

        long xBits = Double.doubleToLongBits(coorX);
        long yBits = Double.doubleToLongBits(coorY);

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

Задание 2:

Теперь переопределим эти же методы для класса **ArrayTabulatedFunction:**

```java
    @Override
    public String toString(){
        String result = "{" + points[0].toString();
        for (int i = 1; i < size; i++){
            result += (", " +points[i].toString());
        }
        result += "}";

        return result;
    }

```

```java
@Override
public boolean equals(Object a) {
    if (this == a) { // Сравнение ссылок
        return true;
    }
    // Проверим является ли переданный объект табулированной функцией
    if (a == null || !(a instanceof TabulatedFunction) ) {
        return false;
    }


    TabulatedFunction b = (TabulatedFunction) a;
    // Перед тем как сравнивать координаты точек табулированных функций
    // Сравним их рааэмер и область определения
    if (this.size != b.getPointsCount()){
        return false;
    }

    if (!(equal(this.leftDomainBorder, b.getLeftDomainBorder()) && equal(this.rightDomainBorder,
            b.getRightDomainBorder()))){
                return false;
            }

    for (int i = 0; i < size; i++){
        if (!(equal(points[i].getCoorX(), b.getPointX(i)) && equal(points[i].getCoorY(), b.getPointY(i)))){
            return false;
        }
    }
    return true;
}
```

```java
@Override
public int hashCode() {
    int hash = size; // Начинаем с количества точек
    // Добавляем хэш-коды всех точек через XOR
    for (int i = 0; i < size; i++) {
        hash ^= points[i].hashCode();
    }
    return hash;
}

@Override
public ArrayTabulatedFunction clone(){
    FunctionPoint[] clone_points = new FunctionPoint[this.size];

    for (int i = 0; i < size; i++){ // Сначала собрали массив
        clone_points[i] = new FunctionPoint(points[i]);
    }

    return new ArrayTabulatedFunction(clone_points);
}
```

Задание 3:

Теперь переопределим методы для **LinkedListTabulatedFunction**:

```java
    @Override
    public String toString() {
        FunctionNode currNode = head;
        String result = "{" + currNode.getPoint().toString();
        currNode = currNode.getNext();
        for (int i = 1; i < size; i++) {
            result += (", " + currNode.getPoint().toString());
            currNode = currNode.getNext();
        }
        result += "}";

        return result;
    }

```

```java
    @Override
    public boolean equals(Object a){
        if (this == a) { // Сравнение ссылок
            return true;
        }
        // Проверим является ли переданный объект табулированной функцией
        if (a == null || !(a instanceof TabulatedFunction) ) {
            return false;
        }


        TabulatedFunction b = (TabulatedFunction) a;
        // Перед тем как сравнивать координаты точек табулированных функций
        // Сравним их раазмер и область определения
        if (this.size != b.getPointsCount()){
            return false;
        }

        if (!(equal(this.leftDomainBorder, b.getLeftDomainBorder()) && equal(this.rightDomainBorder,
                b.getRightDomainBorder()))){
                    return false;
                }

        for (int i = 0; i < size; i++){
            if (!(equal(this.getPointX(i), b.getPointX(i)) && equal(this.getPointY(i), b.getPointY(i)))){
                return false;
            }
        }
        return true;

        }

```

```java
    @Override
    public int hashCode() {
        FunctionNode currNode = head;

        int hash = size; // Начинаем с количества точек
        // Добавляем хэш-коды всех точек через XOR
        for (int i = 0; i < size; i++) {
            hash ^= currNode.getPoint().hashCode();
            currNode = currNode.getNext();
        }
        return hash;
    }

    @Override
    public LinkedListTabulatedFunction clone() {
        LinkedListTabulatedFunction cloned = new LinkedListTabulatedFunction();

        if (size == 0) {
            return cloned; // возвращаем пустой список
        }

        for (int i = 0; i < size; i++) {
            try {
                FunctionPoint point = this.getPoint(i);
                cloned.addPoint(new FunctionPoint(point));
            } catch (InappropriateFunctionPointException e) {
                throw new IllegalArgumentException(message: "Ошибка во время клонирования", e);
            }
        }

        return cloned;
    }

}
```

Задание 4:

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

```java
public class ArrayTabulatedFunction implements TabulatedFunction, Serializable, Cloneable{
```

```java
public class LinkedListTabulatedFunction implements TabulatedFunction, Externalizable, Cloneable{
```

```java
package functions;
public interface TabulatedFunction extends Function{


public int getPointsCount();

public FunctionPoint getPoint(int index) throws FunctionPointIndexOutOfBoundsException, IllegalStateException;

public void setPoint(int index, FunctionPoint point) throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;

public double getPointX(int index) throws FunctionPointIndexOutOfBoundsException, IllegalStateException;

public double getPointY(int index) throws FunctionPointIndexOutOfBoundsException, IllegalStateException;

public void setPointX(int index, double x) throws FunctionPointIndexOutOfBoundsException,InappropriateFunctionPointException, IllegalStateException;

public void setPointY(int index, double y) throws FunctionPointIndexOutOfBoundsException;

public void deletePoint(int index) throws FunctionPointIndexOutOfBoundsException, IllegalStateException;

public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;

public TabulatedFunction clone();
}
```

Задание 5:

Проверим работу добавленных методов, **Main.java**:

```java
import functions.*;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        try {
            System.out.println(x: "=== ТЕСТИРОВАНИЕ МЕТОДОВ toString(), equals(), hashCode(), clone() ===\n");

            // 1. Создаем тестовые данные
            System.out.println(x: "1. СОЗДАНИЕ ТЕСТОВЫХ ФУНКЦИЙ:");

            // ArrayTabulatedFunction
            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)
            };
            ArrayTabulatedFunction arrayFunc1 = new ArrayTabulatedFunction(points1);
            System.out.println(x: "   ArrayTabulatedFunction1 создан");

            // LinkedListTabulatedFunction с такими же точками
            LinkedListTabulatedFunction linkedFunc1 = new LinkedListTabulatedFunction(points1);
            System.out.println(x: "   LinkedListTabulatedFunction1 создан");

            // Функции с другими точками
            FunctionPoint[] points2 = {
                    new FunctionPoint(x: 0, y: 0),
                    new FunctionPoint(x: 1, y: 2),
                    new FunctionPoint(x: 2, y: 5)
            };
            ArrayTabulatedFunction arrayFunc2 = new ArrayTabulatedFunction(points2);
            System.out.println(x: "   ArrayTabulatedFunction2 создан");

            // 2. Тестирование toString()
            System.out.println(x: "\n2. ТЕСТИРОВАНИЕ toString():");
            System.out.println("   ArrayFunc1: " + arrayFunc1.toString());
            System.out.println("   LinkedFunc1: " + linkedFunc1.toString());
            System.out.println("   ArrayFunc2: " + arrayFunc2.toString());

            // 3. Тестирование equals()
            System.out.println(x: "\n3. ТЕСТИРОВАНИЕ equals():");
            System.out.println("   arrayFunc1.equals(linkedFunc1): " + arrayFunc1.equals(linkedFunc1));
            System.out.println("   arrayFunc1.equals(arrayFunc2): " + arrayFunc1.equals(arrayFunc2));
            System.out.println("   arrayFunc1.equals(arrayFunc1): " + arrayFunc1.equals(arrayFunc1));
            System.out.println("   arrayFunc1.equals(null): " + arrayFunc1.equals(a: null));

            // 4. Тестирование hashCode()
            System.out.println(x: "\n4. ТЕСТИРОВАНИЕ hashCode():");
            System.out.println("   arrayFunc1.hashCode(): " + arrayFunc1.hashCode());
            System.out.println("   linkedFunc1.hashCode(): " + linkedFunc1.hashCode());
            System.out.println("   arrayFunc2.hashCode(): " + arrayFunc2.hashCode());

            // Проверка согласованности equals() и hashCode()
            System.out.println(x: "\n   СОГЛАСОВАННОСТЬ equals() и hashCode():");
            System.out.println("   arrayFunc1.equals(linkedFunc1): " + arrayFunc1.equals(linkedFunc1));
            System.out.println("   arrayFunc1.hashCode() == linkedFunc1.hashCode(): " +
                    (arrayFunc1.hashCode() == linkedFunc1.hashCode()));

            // Изменяем точку и проверяем изменение хэш-кода
            ArrayTabulatedFunction arrayFunc1Modified = new ArrayTabulatedFunction(points1);
            arrayFunc1Modified.setPointY(index: 1, y: 1.001); // Незначительное изменение
            System.out.println(x: "\n   После изменения точки Y[1] с 1.0 на 1.001:");
            System.out.println("   Старый hashCode: " + arrayFunc1.hashCode());
            System.out.println("   Новый hashCode: " + arrayFunc1Modified.hashCode());
            System.out.println("   Хэши различны: " + (arrayFunc1.hashCode() != arrayFunc1Modified.hashCode()));
```

```
        // 5. Тестирование clone()
        System.out.println(x: "\n5. ТЕСТИРОВАНИЕ clone():");

        // Клонирование ArrayTabulatedFunction
        ArrayTabulatedFunction arrayClone = arrayFunc1.clone();
        System.out.println("   arrayFunc1.clone().equals(arrayFunc1): " + arrayClone.equals(arrayFunc1));
        System.out.println("   arrayFunc1 == clone: " + (arrayFunc1 == arrayClone));

        // Клонирование LinkedListTabulatedFunction
        LinkedListTabulatedFunction linkedClone = linkedFunc1.clone();
        System.out.println("   linkedFunc1.clone().equals(linkedFunc1): " + linkedClone.equals(linkedFunc1));
        System.out.println("   linkedFunc1 == clone: " + (linkedFunc1 == linkedClone));

        // 6. Проверка глубокого клонирования
        System.out.println(x: "\n6. ПРОВЕРКА ГЛУБОКОГО КЛОНИРОВАНИЯ:");

        // Изменяем оригинальные функции
        arrayFunc1.setPointY(index: 0, y: 999);
        linkedFunc1.setPointY(index: 0, y: 888);

        System.out.println(x: "   После изменения оригиналов:");
        System.out.println("   arrayClone.getPointY(0): " + arrayClone.getPointY(index: 0) + " (должно быть 0)");
        System.out.println("   linkedClone.getPointY(0): " + linkedClone.getPointY(index: 0) + " (должно быть 0)");
        System.out.println("   Клоны не изменились: " +
                (arrayClone.getPointY(index: 0) == 0 && linkedClone.getPointY(index: 0) == 0));

        // 7. Дополнительные проверки
        System.out.println(x: "\n7. ДОПОЛНИТЕЛЬНЫЕ ПРОВЕРКИ:");

        // Проверка с пустой функцией
        ArrayTabulatedFunction emptyArray = new ArrayTabulatedFunction(leftX: 0, rightX: 1, pointsCount: 2);
        ArrayTabulatedFunction emptyArrayClone = emptyArray.clone();
        System.out.println("   Пустая функция: " + emptyArray.toString());
        System.out.println("   Клон пустой функции: " + emptyArrayClone.toString());
        System.out.println("   Пустые функции равны: " + emptyArray.equals(emptyArrayClone));

        System.out.println(x: "\n=== ТЕСТИРОВАНИЕ ЗАВЕРШЕНО ===");

    } catch (Exception e) {
        System.err.println("Ошибка во время тестирования: " + e.getMessage());
        e.printStackTrace();
    }
```

Результат:

```
1. СОЗДАНИЕ ТЕСТОВЫХ ФУНКЦИЙ:
   ArrayTabulatedFunction1 создан
   LinkedListTabulatedFunction1 создан
   ArrayTabulatedFunction2 создан

2. ТЕСТИРОВАНИЕ toString():
   ArrayFunc1: {(0,000, 0,000), (1,000, 1,000), (2,000, 4,000), (3,000, 9,000)}
   LinkedFunc1: {(0,000, 0,000), (1,000, 1,000), (2,000, 4,000), (3,000, 9,000)}
   ArrayFunc2: {(0,000, 0,000), (1,000, 2,000), (2,000, 5,000)}

3. ТЕСТИРОВАНИЕ equals():
   arrayFunc1.equals(linkedFunc1): true
   arrayFunc1.equals(arrayFunc2): false
   arrayFunc1.equals(arrayFunc1): true
   arrayFunc1.equals(null): false

4. ТЕСТИРОВАНИЕ hashCode():
   arrayFunc1.hashCode(): 3801092
   linkedFunc1.hashCode(): 3801092
   arrayFunc2.hashCode(): 2145648643

   СОГЛАСОВАННОСТЬ equals() и hashCode():
   arrayFunc1.equals(linkedFunc1): true
   arrayFunc1.hashCode() == linkedFunc1.hashCode(): true

   После изменения точки Y[1] с 1.0 на 1.001:
   Старый hashCode: 3801092
   Новый hashCode: -1823557514
   Хэши различны: true

5. ТЕСТИРОВАНИЕ clone():
   arrayFunc1.clone().equals(arrayFunc1): true
   arrayFunc1 == clone: false
   linkedFunc1.clone().equals(linkedFunc1): true
   linkedFunc1 == clone: false

6. ПРОВЕРКА ГЛУБОКОГО КЛОНИРОВАНИЯ:
   После изменения оригиналов:
   arrayClone.getPointY(0): 0.0 (должно быть 0)
   linkedClone.getPointY(0): 0.0 (должно быть 0)
   Клоны не изменились: true
```