Лабораторная работа №4

Расширение возможности классов, связанных с табулированными функциями, переопределение в них методов, унаследованных из класса Object.

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

Филянов Кирилл Владимирович
6203-010302D

Ход выполнения работы.

Переопределение в классе FunctionPoint методы:

toString() - текстовое описание точки
equals(Object o) - сравнение координат с учетом погрешности
hashCode() - вычисление хэш-кода через XOR битовых представлений
clone() - создание копии объекта

1. Метод toString():

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

    FunctionNode curr = head.next;
    boolean first = true;
    while (curr != head) {
        if (!first) {
            sb.append(", ");
        }
        first = false;
        sb.append(curr.point.toString());
        curr = curr.next;
    }

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

2. Метод equals(Object o)

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

    // Сравнение с другим LinkedListTabulatedFunction
    if (o instanceof LinkedListTabulatedFunction) {
        LinkedListTabulatedFunction other = (LinkedListTabulatedFunction) o;

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

        FunctionNode thisCurr = this.head.next;
        FunctionNode otherCurr = other.head.next;

        while (thisCurr != this.head && otherCurr != other.head) {
            if (!thisCurr.point.equals(otherCurr.point)) {
                return false;
            }
            thisCurr = thisCurr.next;
            otherCurr = otherCurr.next;
        }

        // Проверяем, что оба списка закончились одновременно
        return thisCurr == this.head && otherCurr == other.head;
    }
```

```java
    // Общий случай для любого TabulatedFunction
    if (!(o instanceof TabulatedFunction)) return false;

    TabulatedFunction other = (TabulatedFunction) o;

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

    // Сравниваем все точки
    FunctionNode curr = head.next;
    int i = 0;
    while (curr != head && i < pointsCount) {
        double x1 = curr.point.getX();
        double y1 = curr.point.getY();

        double x2 = other.getPointX(i);
        double y2 = other.getPointY(i);

        // Сравнение при помощи машинного эпсилона
        final double eps = 1e-10;
        if (Math.abs(x1 - x2) > eps || Math.abs(y1 - y2) > eps) {
            return false;
        }

        curr = curr.next;
        i++;
    }
    return true;
}
```

3. Метод hashCode():

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

    // XOR хэш-кодов всех точек
    FunctionNode curr = head.next;
    while (curr != head) {
        hash ^= curr.point.hashCode();
        curr = curr.next;
    }

    return hash;
}
```

4. Метод clone():

```java
@Override
public Object clone() throws CloneNotSupportedException {
    try {

        // Создаем массив точек из текущего списка (глубокое копирование точек)
        FunctionPoint[] points = new FunctionPoint[pointsCount];
        FunctionNode curr = head.next;
        int i = 0;
        while (curr != head) {
            points[i++] = new FunctionPoint(curr.point);
            curr = curr.next;
        }

        LinkedListTabulatedFunction cloned = new LinkedListTabulatedFunction(points);

        cloned.lastAccessed = null;
        cloned.lastIndex = -1;

        return cloned;
    } catch (Exception e) {
        throw new CloneNotSupportedException("Ошибка при клонировании: " + e.getMessage());
    }
}
```

Сделать все объекты типа TabulatedFunction клонируемыми.

1. Добавлен extends Cloneable в интерфейс TabulatedFunction

2. Добавлен метод Object clone() throws CloneNotSupportedException в интерфейс

3. Обе реализации (ArrayTabulatedFunction и LinkedListTabulatedFunction) уже реализуют Cloneable

```
package functions;

public interface TabulatedFunction extends Function, Cloneable {  30 usages  2 implementations
    int getPointsCount();  14 usages  2 implementations
    FunctionPoint getPoint(int index);  no usages  2 implementations
    void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException;
    double getPointX(int index);  6 usages  2 implementations
    void setPointX(int index, double x) throws InappropriateFunctionPointException;  no usages  2
    double getPointY(int index);  16 usages  2 implementations
    void setPointY(int index, double y);  8 usages  2 implementations
    void deletePoint(int index);  no usages  2 implementations
    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;  1 usage  2 imp
    Object clone() throws CloneNotSupportedException;  2 implementations
}
```

Проверка работы всех переопределенных методов.

```
1. ТЕСТИРОВАНИЕ ВСЕХ МЕТОДОВ:
------------------------------------------------

a) toString() - проверка формата вывода:
   Array:    {(0,00; 0,00), (1,00; 2,00), (2,00; 8,00), (3,00; 18,00)}
   LinkedList: {(0,00; 0,00), (1,00; 2,00), (2,00; 8,00), (3,00; 18,00)}
   ✓ Оба метода выводят в формате {(x; y), ...}

b) equals() - основные случаи:
   1. Одинаковые Array: true (true)
   2. Одинаковые LinkedList: true (true)
   3. Array vs LinkedList (одинаковые данные): true (true)
   4. Array vs Array (разные данные): false (false)
   5. С null: false (false)
   ✓ Все сравнения работают корректно

c) hashCode() - проверка контракта:
   Array equals→hashCode: true (true)
   LinkedList equals→hashCode: true (true)
   Проверка изменения хэш-кода:
   Хэш-код изменяется при модификации Y: true (true)
   ✓ Контракт hashCode/equals соблюдается

d) clone() - глубокое копирование:
   Array глубокое копирование: true (true)
   LinkedList глубокое копирование: true (true)
   ✓ Оба метода реализуют глубокое копирование
```

```
2. ПРОВЕРКА ТРЕБОВАНИЙ ЗАДАНИЯ 5:
------------------------------------------------

1. toString() выводит строковое представление:
   Array: {(-1,00; 1,00), (0,00; 0,00), (1,00; 1,00)}
   LinkedList: {(-1,00; 1,00), (0,00; 0,00), (1,00; 1,00)}
   ✓ Оба метода работают

2. equals() для различных случаев:
   а) Два одинаковых Array: true (true)
   б) Array и LinkedList с одинаковыми данными: true (true)
   в) Два разных Array: false (false)
   г) Array и null: false (false)
   ✓ Все сравнения корректны

3. hashCode() и согласованность с equals():
   а) Хэш-коды одинаковых объектов:
      arrayA.hashCode(): 3801092
      arrayB.hashCode(): 3801092
      linkedA.hashCode(): 3801092
   б) Консистентность equals/hashCode: true (true)
   в) Изменение хэш-кода при модификации:
      До: 3801092, После: 1856867910, Изменился: true (true)

4. clone() - глубокое копирование:
   а) Array: оригинал ≠ клон после модификации Y: true (true)
   б) LinkedList: оригинал ≠ клон после модификации Y: true (true)
   в) LinkedList: разная структура после добавления точки: true (true)
   ✓ Глубокое копирование подтверждено
```

```
==================================================
ИТОГ ПРОВЕРКИ:
1. toString() - ✓ работает для обоих классов
2. equals() - ✓ корректно сравнивает все случаи
3. hashCode() - ✓ согласован с equals(), меняется при модификации
4. clone() - ✓ глубокое копирование для обоих классов
5. Все методы - ✓ соответствуют требованиям задания 5

======================================================================
✓ ВСЕ ТРЕБОВАНИЯ ЗАДАНИЯ 5 ВЫПОЛНЕНЫ!
======================================================================
```