Отчет по лабораторной работе №5 предмета ООП

Выполнил Брусенцев Степан Эдуардович
Студент группы 6204-010302D

Задание на лабораторную работу:

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

Задание 1

Переопредел в классе FunctionPoint следующие методы:
String toString()
boolean equals(Object o)
int hashCode()
int hashCode()

```
    @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 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() { return new FunctionPoint(x, y); }
}
```

Задание 2

Переопредел в классе ArrayTabulatedFunction аналогические методы.

```java
@Override
public String toString() {
    StringBuilder sb = new StringBuilder("{");
    for (int i = 0; i < len; i++) {
        if (i > 0) {
            sb.append(", ");
        }
        sb.append(funct[i].toString());
    }
    sb.append("}");
    return sb.toString();
}
```

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

    TabulatedFunction that = (TabulatedFunction) o;

    if (len != that.getPointsCount()) return false;

    // Оптимизация для ArrayTabulatedFunction
    if (o instanceof ArrayTabulatedFunction) {
        ArrayTabulatedFunction arrThat = (ArrayTabulatedFunction) o;
        for (int i = 0; i < len; i++) {
            if (Double.compare(funct[i].getX(), arrThat.funct[i].getX()) != 0 ||
                Double.compare(funct[i].getY(), arrThat.funct[i].getY()) != 0) {
                return false;
            }
        }
    } else {
        // Общий случай для других реализаций TabulatedFunction
        for (int i = 0; i < len; i++) {
            FunctionPoint thatPoint = that.getPoint(i);
            if (Double.compare(funct[i].getX(), thatPoint.getX()) != 0 ||
                Double.compare(funct[i].getY(), thatPoint.getY()) != 0) {
                return false;
            }
        }
    }

    return true;
}
```

```java
    @Override
    public int hashCode() {
        int result = len;
        for (int i = 0; i < len; i++) {
            result ^= funct[i].hashCode();
        }
        return result;
    }

    @Override
    public Object clone() {
        FunctionPoint[] clonedPoints = new FunctionPoint[len];
        for (int i = 0; i < len; i++) {
            clonedPoints[i] = (FunctionPoint) funct[i].clone();
        }
        return new ArrayTabulatedFunction(clonedPoints);
    }

}
```

Задание 3

Также переопределил методы в классе LinkedListTabulatedFunction.

```java
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder("{");
        FunctionNode current = head.next;
        boolean first = true;
        while (current != head) {
            if (!first) {
                sb.append(", ");
            }
            sb.append(current.value.toString());
            current = current.next;
            first = false;
        }
        sb.append("}");
        return sb.toString();
    }
```

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

        TabulatedFunction that = (TabulatedFunction) o;

        if (size != that.getPointsCount()) return false;

        // Оптимизация для LinkedListTabulatedFunction
        if (o instanceof LinkedListTabulatedFunction) {
            LinkedListTabulatedFunction listThat = (LinkedListTabulatedFunction) o;
            FunctionNode currentThis = head.next;
            FunctionNode currentThat = listThat.head.next;

            while (currentThis != head && currentThat != listThat.head) {
                if (Double.compare(currentThis.value.getX(), currentThat.value.getX()) != 0 ||
                    Double.compare(currentThis.value.getY(), currentThat.value.getY()) != 0) {
                    return false;
                }
                currentThis = currentThis.next;
                currentThat = currentThat.next;
            }
        } else {
            // Общий случай для других реализаций TabulatedFunction
            FunctionNode current = head.next;
            int index = 0;
            while (current != head) {
                FunctionPoint thatPoint = that.getPoint(index);
                if (Double.compare(current.value.getX(), thatPoint.getX()) != 0 ||
                    Double.compare(current.value.getY(), thatPoint.getY()) != 0) {
                    return false;
                }
                current = current.next;
                index++;
```

```java
    @Override
    public int hashCode() {
        int result = size;
        FunctionNode current = head.next;
        while (current != head) {
            result ^= current.value.hashCode();
            current = current.next;
        }
        return result;
    }

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

        // Пересборка списка без использования методов добавления
        FunctionNode current = head.next;
        FunctionNode prevNode = cloned.head;

        while (current != head) {
            FunctionNode newNode = new FunctionNode((FunctionPoint) current.value.clone());
            newNode.prev = prevNode;
            newNode.next = cloned.head;
            prevNode.next = newNode;
            cloned.head.prev = newNode;
            prevNode = newNode;
            cloned.size++;
            current = current.next;
        }

        return cloned;
    }
}
```

Задание 4

Сделал так, чтобы все объекты типа TabulatedFunction были клонируемы и добавил их в интерфейс.

Задание 5

Провел тесты написанных методов.
=== ТЕСТ 1: toString() ===

ArrayTabulatedFunction.toString():
{(0.0; 1.2), (1.0; 3.8), (2.0; 15.2)}

LinkedListTabulatedFunction.toString():
{(0.0; 1.2), (1.0; 3.8), (2.0; 15.2)}

=== ТЕСТ 2: equals() ===

1. Сравнение одинаковых ArrayTabulatedFunction:
  arrayFunc1.equals(arrayFunc2): true
  arrayFunc2.equals(arrayFunc1): true

2. Сравнение одинаковых LinkedListTabulatedFunction:
  listFunc1.equals(listFunc2): true
  listFunc2.equals(listFunc1): true

3. Сравнение разных ArrayTabulatedFunction:
  arrayFunc1.equals(arrayFunc3): false
  arrayFunc3.equals(arrayFunc1): false

4. Сравнение разных LinkedListTabulatedFunction:
  listFunc1.equals(listFunc3): false
  listFunc3.equals(listFunc1): false

5. Сравнение ArrayTabulatedFunction и LinkedListTabulatedFunction (одинаковые данные):
  arrayFunc1.equals(listFunc1): true

listFunc1.equals(arrayFunc1): true

6. Сравнение с null:
listFunc1.equals(null): false
listFunc1.equals(null): false

7. Сравнение с объектом другого типа:
arrayFunc1.equals("строка"): false
listFunc1.equals(new Object()): false

=== ТЕСТ 3: hashCode() ===

1. Хэш-коды одинаковых ArrayTabulatedFunction:
arrayFunc1.hashCode(): 1930428419
arrayFunc2.hashCode(): 1930428419
Совпадают: true

2. Хэш-коды одинаковых LinkedListTabulatedFunction:
listFunc1.hashCode(): 1930428419
listFunc2.hashCode(): 1930428419
Совпадают: true

3. Хэш-коды разных ArrayTabulatedFunction:
arrayFunc1.hashCode(): 1930428419
arrayFunc3.hashCode(): 359137283
Совпадают: false

4. Хэш-коды разных LinkedListTabulatedFunction:
listFunc1.hashCode(): 1930428419
listFunc3.hashCode(): 359137283
Совпадают: false

5. Хэш-коды ArrayTabulatedFunction и LinkedListTabulatedFunction (одинаковые данные):
arrayFunc1.hashCode(): 1930428419
listFunc1.hashCode(): 1930428419

Совпадают: true

6. Проверка согласованности equals() и hashCode():
Если arrayFunc1.equals(arrayFunc2), то hash1 == hash2: true
Если arrayFunc1.equals(arrayFunc3), то hash1 == hash5: true

7. Изменение объекта и проверка хэш-кода:
Исходный хэш-код arrayFunc1: 1930428419
Новый хэш-код после изменения Y на 0.001: -1521046294
Хэш-код изменился: true
Хэш-код после восстановления: 1930428419
Хэш-код восстановлен: true

=== ТЕСТ 4: clone() ===

1. Тест clone() для ArrayTabulatedFunction:
Исходный объект: {(0.0; 1.2), (1.0; 3.8), (2.0; 15.2)}
Клонированный объект: {(0.0; 1.2), (1.0; 3.8), (2.0; 15.2)}
Объекты равны: true
Это разные объекты (по ссылке): true

После изменения исходного объекта:
Исходный объект: {(0.0; 1.2), (1.0; 13.8), (2.0; 15.2)}
Клонированный объект: {(0.0; 1.2), (1.0; 3.8), (2.0; 15.2)}
Клон не изменился: true

2. Тест clone() для LinkedListTabulatedFunction:
Исходный объект: {(0.0; 1.2), (1.0; 3.8), (2.0; 15.2)}
Клонированный объект: {(0.0; 1.2), (1.0; 3.8), (2.0; 15.2)}
Объекты равны: true
Это разные объекты (по ссылке): true

После изменения исходного объекта:
Исходный объект: {(0.0; 1.2), (1.0; 13.8), (2.0; 15.2)}
Клонированный объект: {(0.0; 1.2), (1.0; 3.8), (2.0; 15.2)}
Клон не изменился: true

3. Проверка глубокого клонирования точек:
ArrayTabulatedFunction:
Сравниваем точку с индексом 0:
Исходная точка из arrayFunc: (0.0; 1.2) (x=0.0, y=1.2)
Клонированная точка из clonedArray: (0.0; 1.2) (x=0.0, y=1.2)
Исходная точка (по ссылке): 1528902577
Клонированная точка (по ссылке): 1927950199
Это разные объекты: true
Но они равны по содержимому (координаты одинаковые): true
Координаты совпадают: x одинаковы? true, y одинаковы? true

LinkedListTabulatedFunction:
Сравниваем точку с индексом 0:
Исходная точка из listFunc: (0.0; 1.2) (x=0.0, y=1.2)
Клонированная точка из clonedList: (0.0; 1.2) (x=0.0, y=1.2)
Исходная точка (по ссылке): 989110044
Клонированная точка (по ссылке): 424058530
Это разные объекты: true
Но они равны по содержимому (координаты одинаковые): true
Координаты совпадают: x одинаковы? true, y одинаковы? true