ОТЧЕТ ПО
ЛАБОРАТОРНОЙ РАБОТЕ №5

Выполнил:
Батуров Максим Дмитриевич
Группа: 6203-010302D

Оглавление

Задание №1

В классе FunctionPoint были переопределены основные методы. Для toString() реализовано возвращение строки в формате (x; y) с координатами точки через конкатенацию. В equals() добавлена проверка типа объекта и сравнение координат с использованием epsilon для корректной работы с числами с плавающей точкой. HashCode() преобразует координаты double в long через Double.doubleToLongBits(), разделяет каждый long на младшие и старшие 4 байта, затем комбинирует все части через XOR. Метод clone() создает новую точку через конструктор копирования.

```java
// возвращает текстовое описание точки
@Override
public String toString() {
    return "(" + x + "; " + y + ")";
}

// сравнивает текущую точку с другим объектом
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;

    FunctionPoint that = (FunctionPoint) o;

    // сравниваем координаты с учетом погрешности
    final double EPSILON = 1e-10;
    return Math.abs(this.x - that.x) < EPSILON &&
            Math.abs(this.y - that.y) < EPSILON;
}

// вычисляет хэш-код точки на основе её координат
@Override
public int hashCode(){
    long xBit = Double.doubleToLongBits(this.x);
    long yBit = Double.doubleToLongBits(this.y);

    int x1 = (int)(xBit & 0xFFFFFFFF);
    int x2 = (int)(xBit >>> 32);
    int y1 = (int)(yBit & 0xFFFFFFFF);
    int y2 = (int)(yBit >>> 32);

    return x1 ^ x2 ^ y1 ^ y2;
}
```

```java
// создает и возвращает копию текущей точки
@Override
public Object clone() {
    try {
        return new FunctionPoint(this);
    } catch (Exception e) {
        throw new InternalError("ошибка при клонировании: " + e.getMessage());
    }
}
```

Задание №2

Для ArrayTabulatedFunction реализован toString(), он формирует строку в формате {(x1; y1), (x2; y2), тд.} с использованием StringBuilder для эффективной конкатенации. В equals() реализована оптимизированная проверка для объектов ArrayTabulatedFunction через прямой доступ к массивам точек, а для других реализаций TabulatedFunction используется интерфейсный подход. HashCode() начинает с количества точек и комбинирует хэш-коды всех точек через операцию XOR. Clone() выполняет глубокое копирование путем клонирования каждой точки и создания нового массива.

```java
// возвращает текстовое описание табулированной ф-ции
@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("{");
    for (int i = 0; i < pointsCount; i++) {
        sb.append(points[i].toString()); // используем toString точки
        if (i < pointsCount - 1) {
            sb.append(", ");
        }
    }
    sb.append("}");
    return sb.toString();
}
```

```java
// сравнивает текущую ф-цию с другим объектом
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof TabulatedFunction)) return false;

    TabulatedFunction other = (TabulatedFunction) o;

    // проверяем количество точек
    if (this.pointsCount != other.getPointsCount()) return false;

    // если объект arraytabulatedfunction
    if (other instanceof ArrayTabulatedFunction) {
        ArrayTabulatedFunction arrayOther = (ArrayTabulatedFunction) other;
        for (int i = 0; i < pointsCount; i++) {
            if (!this.points[i].equals(arrayOther.points[i])) {
                return false;
            }
        }
    } else {
        // для других реализаций TabulatedFunction используем интерфейс
        for (int i = 0; i < pointsCount; i++) {
            FunctionPoint otherPoint = other.getPoint(i);
            if (!this.points[i].equals(otherPoint)) {
                return false;
            }
        }
    }

    return true;
}
```

```java
// вычисляет хэш-код ф-ции на основе её точек
@Override
public int hashCode() {
    int hash = pointsCount; // начинаем с количества точек

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

    return hash;
}

// создает и возвращает глубокую копию текущей ф-ции
@Override
public Object clone() {
    try {
        // создаем массив копий точек
        FunctionPoint[] clonedPoints = new FunctionPoint[points.length];
        for (int i = 0; i < pointsCount; i++) {
            clonedPoints[i] = (FunctionPoint) points[i].clone(); // глубокая копия точки
        }

        // создаем новую функцию с копиями точек
        ArrayTabulatedFunction cloned = new ArrayTabulatedFunction();
        cloned.points = clonedPoints;
        cloned.pointsCount = this.pointsCount;

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

Задание №3

В LinkedListTabulatedFunction toString() проходит по узлам списка и формирует строковое представление в формате {(x1; y1), (x2; y2), ...}. Equals() содержит оптимизированную проверку для объектов того же класса через прямое сравнение узлов, а для других реализаций TabulatedFunction использует методы интерфейса. HashCode() последовательно обходит все узлы списка и комбинирует хэш-коды точек через XOR. В Clone() создается массив копий точек, затем используется существующий конструктор для формирования нового связного списка, что исключает необходимость ручного связывания узлов.

```java
// возвращает текстовое описание табулированной ф-ции
@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("{");
    FunctionNode current = head.getNext();
    while (current != head) {
        sb.append(current.getPoint().toString());
        current = current.getNext();
        if (current != head) {
            sb.append(", ");
        }
    }
    sb.append("}");
    return sb.toString();
}
```

```java
// сравнивает текущую ф-цию с другим объектом
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof TabulatedFunction)) return false;

    TabulatedFunction other = (TabulatedFunction) o;

    // проверяем количество точек
    if (this.pointcount != other.getPointsCount()) return false;

    // если объект linkedlisttabulatedfunction
    if (other instanceof LinkedListTabulatedFunction) {
        LinkedListTabulatedFunction listOther = (LinkedListTabulatedFunction) other;
        FunctionNode currentThis = this.head.getNext();
        FunctionNode currentOther = listOther.head.getNext();

        while (currentThis != this.head && currentOther != listOther.head) {
            if (!currentThis.getPoint().equals(currentOther.getPoint())) {
                return false;
            }
            currentThis = currentThis.getNext();
            currentOther = currentOther.getNext();
        }
    } else {
        // для других реализаций TabulatedFunction используем интерфейс
        for (int i = 0; i < pointcount; i++) {
            FunctionPoint thisPoint = this.getPoint(i);
            FunctionPoint otherPoint = other.getPoint(i);
            if (!thisPoint.equals(otherPoint)) {
                return false;
            }
        }
    }

    return true;
}
```

```java
// вычисляет хэш-код ф-ции на основе её точек
@Override
public int hashCode() {
    int hash = pointcount; // начинаем с количества точе

    // комбинируем хэш-коды всех точек через xor
    FunctionNode current = head.getNext();
    while (current != head) {
        hash ^= current.getPoint().hashCode();
        current = current.getNext();
    }

    return hash;
}
```

```java
// создает и возвращает копию текущей ф-ции
@Override
public Object clone() {
    FunctionPoint[] pointArr = new FunctionPoint[pointcount];
    FunctionNode cur = head.getNext();
    int i = 0;
    while (cur != head) {
        pointArr[i] = (FunctionPoint) cur.getPoint().clone();
        cur = cur.getNext();
        i++;
    }
    return new LinkedListTabulatedFunction(pointArr);
}
```

Задание №4

Интерфейс TabulatedFunction был расширен с помощью Cloneable. Метод clone() был объявлен в интерфейсе. В классах ArrayTabulatedFunction и LinkedListTabulatedFunction методы clone() имеют модификатор public для обеспечения корректного вызова механизма клонирования.

```java
package functions;

public interface TabulatedFunction extends Function, Cloneable {  13 usages  2 implementations
    // возвращает кол-во точек в табулированной ф-ции
    int getPointsCount();  7 usages  2 implementations

    // возвращает точку по указанному индексу
    FunctionPoint getPoint(int index) throws FunctionPointIndexOutOfBoundsException;  11 usages  2 implementations

    // заменяет точку по указанному индексу
    void setPoint(int index, FunctionPoint point) throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;

    // возвращает координату x точки по индексу
    double getPointX(int index) throws FunctionPointIndexOutOfBoundsException;  2 usages  2 implementations

    // устанавливает координату x точки по индексу
    void setPointX(int index, double x) throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;  no usages  2

    // возвращает координату y точки по индексу
    double getPointY(int index) throws FunctionPointIndexOutOfBoundsException;  2 usages  2 implementations

    // устанавливает координату y точки по индексу
    void setPointY(int index, double y) throws FunctionPointIndexOutOfBoundsException;  3 usages  2 implementations

    // удаляет точку по указанному индексу
    void deletePoint(int index) throws FunctionPointIndexOutOfBoundsException, IllegalStateException;  no usages  2 implementations

    // добавляет новую точку в ф-цию
    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;  no usages  2 implementations

    // создает и возвращает глубокую копию текущей ф-ции
    Object clone();  2 implementations
}
```

Задание №5

```
1. Тестирование toString():
ArrayFunc1: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}
ListFunc1: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}

2. Тестирование equals():
arrayFunc1.equals(arrayFunc2): true
arrayFunc1.equals(arrayFunc3): false
listFunc1.equals(listFunc2): true
listFunc1.equals(listFunc3): false
arrayFunc1.equals(listFunc1): true
listFunc1.equals(arrayFunc1): true

3. Тестирование hashCode():
arrayFunc1.hashCode(): 1075576835
arrayFunc2.hashCode(): 1075576835
arrayFunc3.hashCode(): 1074003971
listFunc1.hashCode(): 1075576835
listFunc2.hashCode(): 1075576835
listFunc3.hashCode(): 1074003971

4. Проверка согласованности equals() и hashCode():
arrayFunc1.equals(arrayFunc2) && arrayFunc1.hashCode() == arrayFunc2.hashCode(): true
listFunc1.equals(listFunc2) && listFunc1.hashCode() == listFunc2.hashCode(): true

5. Изменение объекта и проверка хэш-кода:
Исходный хэш-код: 1075576835
Хэш-код после изменения Y[1] на 2.001: 162421818
arrayFunc1.equals(arrayFunc4) после изменения: false

6. Тестирование clone():
arrayFunc1.equals(arrayClone): true
arrayFunc1 == arrayClone: false
listFunc1.equals(listClone): true
listFunc1 == listClone: false
7. Проверка глубокого клонирования:
До изменения - arrayFunc1[1]: (1.0; 3.0)
После изменения arrayFunc1[1]: (1.0; 10.0)
Клон arrayClone[1]: (1.0; 3.0)
Клон не изменился: true
До изменения - listFunc1[1]: (1.0; 3.0)
После изменения listFunc1[1]: (1.0; 15.0)
Клон listClone[1]: (1.0; 3.0)
Клон не изменился: true

8. Тестирование FunctionPoint методов:
pt1: (1.5; 2.5)
pt1.equals(pt2): true
pt1.equals(pt3): false
pt1.hashCode(): 2147221504
pt2.hashCode(): 2147221504
pt3.hashCode(): -1288699903
pt1.equals(ptClone): true
pt1 == ptClone: false
```