МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П.Королева»
(Самарский университет)

ОТЧЕТ ПО
ЛАБОРАТОРНОЙ РАБОТЕ № 5

«Расширение возможности классов, связанных с табулированными функциями, переопределив в них методы, унаследованные из класса Object»

по курсу
Объектно-ориентированное программирование

Выполнила: Гонтарь Анастасия Вячеславовна,
студент группы 6203-010302D

Содержание

Задание №1

Я переопределила в FunctionPoint методы

toString

```java
@Override
public String toString() { return String.format("(%.1f; %.1f)", this.x, this.y); }
```

equals

```java
@Override
public boolean equals(Object o){
    if (this == o) // если ссылки равны, то тот же объект
        return true;
    if (o == null || getClass() != o.getClass()) // проверка является ли объект точкой
        return false;

    double epsilon = 1e-10;
    return Math.abs(this.x - ((FunctionPoint) o).getX()) < epsilon && // совпадают ли координаты
            Math.abs(this.y - ((FunctionPoint) o).getY()) < epsilon;
}
```

hashCode

```java
@Override
public int hashCode() {
    long xBits = Double.doubleToLongBits(this.x);
    long yBits = Double.doubleToLongBits(this.y);

    int xHash = (int) (xBits ^ (xBits >>> 32));
    int yHash = (int) (yBits ^ (yBits >>> 32));

    return xHash ^ yHash;
}
```

clone

```java
@Override
public Object clone() { return new FunctionPoint(x, y); }
```

Задание №2

Я переопределила в ArrayTabulatedFunction методы

toString

```java
@Override
public String toString(){
    String result = "{";
    for (int i = 0; i<pointsCount; i++){
        result += String.format("(%.1f; %.1f)", points[i].getX(), points[i].getY());
    }
    result+= "}";
    return result;
}
```

equals

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

    if (o instanceof ArrayTabulatedFunction) { //если объект тот же
        ArrayTabulatedFunction func = (ArrayTabulatedFunction) o;
        if (this.pointsCount != func.pointsCount) // сравниваем количество точек
            return false;

        for (int i = 0; i < pointsCount; i++){
            if (!this.points[i].equals(func.points[i])) //сравниваем точки с помощью equals
                return false;

        }
    }
```

```java
    else { // если объект другой реализации TabulatedFunction
        TabulatedFunction func = (TabulatedFunction) o;
        if (this.getPointsCount() != func.getPointsCount()) // сравниваем количество точек
            return false;

        for (int i = 0; i < pointsCount; i++) {
            if (!this.getPoint(i).equals(func.getPoint(i))) // сравниваем точки
                return false;
        }
    }
    return true;
}
```

hashCode

```java
@Override
public int hashCode(){
    int hash = pointsCount; // начинаем с количества точек

    for (int i = 0; i < pointsCount; i++) {
        hash ^= points[i].hashCode();
    }
    return hash;
}
```

clone

```
@Override
public Object clone(){
    FunctionPoint[] pointsClone = new FunctionPoint[pointsCount];
    for(int i =0; i < pointsCount; i++){
        pointsClone[i] = (FunctionPoint) points[i].clone();
    }
    return new ArrayTabulatedFunction(pointsClone);
}
```

Задание №3

Я переопределила в LinkedListTabulatedFunction методы

toString

```
@Override
public String toString(){
    String result = "{";
    FunctionNode curr = head.getNext();
    while (curr!= head){
        result += String.format("(%.1f; %.1f)", curr.getPoint().getX(), curr.getPoint().getY());
        curr =  curr.getNext();
    }
    result+= "}";
    return result;
}
```

equals

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

    if (o instanceof LinkedListTabulatedFunction) { //если объект тот же
        LinkedListTabulatedFunction func = (LinkedListTabulatedFunction) o;
        if (this.pointsCount != func.pointsCount) // сравниваем количество точек
            return false;

        FunctionNode currFirst = this.head.getNext();
        FunctionNode currSecond = func.head.getNext();
        while (currFirst != head ){ // сравниваем точки
            if (!currFirst.getPoint().equals(currSecond.getPoint()))
                return false;
            currFirst = currFirst.getNext();
            currSecond = currSecond.getNext();
        }
    }
```

```
    else { // если объект другой реализации TabulatedFunction
        TabulatedFunction func = (TabulatedFunction) o;
        if (this.getPointsCount() != func.getPointsCount()) // сравниваем количество точек
            return false;

        for (int i = 0; i < pointsCount; i++) {
            if (!this.getPoint(i).equals(func.getPoint(i))) // сравниваем точки
                return false;
        }
    }
    return true;
}
```

hashCode

```
@Override
public int hashCode(){
    int hash = pointsCount; // начинаем с количества точек

    FunctionNode curr = head.getNext();
    while (curr != head ){ // сравниваем точки
        hash ^= curr.getPoint().hashCode();
        curr = curr.getNext();
    }
    return hash;
}
```

clone

```
@Override
public Object clone(){
    FunctionPoint[] pointsCopy = new FunctionPoint[pointsCount];// создаем временный массив для хранения копий всех точек
    FunctionNode curr = head.getNext();
   for (int i = 0; curr!= head; i++) {
        pointsCopy[i] = new FunctionPoint(curr.getPoint());
        curr = curr.getNext();
   }
   return new LinkedListTabulatedFunction(pointsCopy);
}
```

Задание №4

Для выполнения этого задания я объявила в интерфейсе TabulatedFunction метод clone, унаследованный из класса Cloneable

```
public interface TabulatedFunction extends Function, Cloneable {  13 usages  2 implementations

    public Object clone();  2 implementations
```

Задание №5

Тестирование написанных методов. Для этого я создала 3 массива типа FunctionPoint , два из которых идентичны

```java
import functions.*;

public class Main {
    public static void main(String[] args) {

        FunctionPoint[] points1 = {new FunctionPoint( x: 0.0,  y: 1.5), new FunctionPoint( x: 4.2,  y: 7), new FunctionPoint( x: 10,  y: 13.46)};
        FunctionPoint[] points2 = {new FunctionPoint( x: 0.0,  y: 1.5), new FunctionPoint( x: 4.2,  y: 7), new FunctionPoint( x: 10,  y: 13.46)};
        FunctionPoint[] points3 = {new FunctionPoint( x: 0.0,  y: 1.5), new FunctionPoint( x: 2,  y: 7), new FunctionPoint( x: 10,  y: 13.46)};

        System.out.println("== Тестирование toString() ==");
        ArrayTabulatedFunction arrayFunc1 = new ArrayTabulatedFunction(points1);
        LinkedListTabulatedFunction linkedlistFunc1 = new LinkedListTabulatedFunction(points1);

        System.out.println("Array: " + arrayFunc1.toString());
        System.out.println("LinkedList: " + linkedlistFunc1.toString());

        System.out.println("\n== Тестирование equals() ==");
        ArrayTabulatedFunction arrayFunc2 = new ArrayTabulatedFunction(points2);
        ArrayTabulatedFunction arrayFunc3 = new ArrayTabulatedFunction(points3);
        LinkedListTabulatedFunction linkedlistFunc2 = new LinkedListTabulatedFunction(points2);
        LinkedListTabulatedFunction linkedlistFunc3 = new LinkedListTabulatedFunction(points3);

        System.out.println("array1 == array2: " + arrayFunc1.equals(arrayFunc2)); // true
        System.out.println("linkedlist1 == linkedlist2: " + linkedlistFunc1.equals(linkedlistFunc2)); // true
        System.out.println("array1 == linkedlist1: " + arrayFunc1.equals(linkedlistFunc1)); // true
        System.out.println("array1 == array3: " + arrayFunc1.equals(arrayFunc3)); // false
```

```java
        System.out.println("\n== Тестирование hashCode() ==");
        System.out.println("array1 hashCode: " + arrayFunc1.hashCode());
        System.out.println("array2 hashCode: " + arrayFunc2.hashCode());
        System.out.println("array3 hashCode: " + arrayFunc3.hashCode());
        System.out.println("linkedlist1 hashCode: " + linkedlistFunc1.hashCode());
        System.out.println("linkedlist2 hashCode: " + linkedlistFunc2.hashCode());
        System.out.println("linkedlist3 hashCode: " + linkedlistFunc3.hashCode());

        System.out.println("\n== Проверка согласованности hashCode и equals ==");
        System.out.println("array1 hashCode == array2 hashCode: " + (arrayFunc1.hashCode() == arrayFunc2.hashCode()));
        System.out.println("linkedlist1 hashCode == linkedlist2 hashCode: " + (linkedlistFunc1.hashCode() == linkedlistFunc2.hashCode()));
        System.out.println("array1 hashCode == linkedlist1 hashCode: " + (arrayFunc1.hashCode() == linkedlistFunc1.hashCode()));

        System.out.println("\n== Проверка изменения hashCode() ==");
        int origHash = arrayFunc1.hashCode();
        arrayFunc1.setPointY( index: 1,  y: 7.00001); // незначительное изменение
        int newHash = arrayFunc1.hashCode();
        System.out.println("Изначальный: " + origHash);
        System.out.println("Измененный: " + newHash);
```

```java
        System.out.println("\n== Тестирование clone() ==");
        ArrayTabulatedFunction arrayClone = (ArrayTabulatedFunction) arrayFunc1.clone();
        LinkedListTabulatedFunction linkedlistClone = (LinkedListTabulatedFunction) linkedlistFunc1.clone();

        System.out.println("Изначальный array: " + arrayFunc1.toString());
        System.out.println("Клон array: " + arrayClone.toString());
        System.out.println("Изначальный linkedlist: " + linkedlistFunc1.toString());
        System.out.println("Клон linkedlist : " + linkedlistClone.toString());

        System.out.println("\n== Проверка глубокого копирования ==");
        arrayFunc1.setPointY( index: 0,  y: 10); // изменяем оригинал
        linkedlistFunc1.setPointY( index: 0,  y: 10);  // изменяем оригинал

        System.out.println("Измененный array: " + arrayFunc1.toString());
        System.out.println("Клон array: " + arrayClone.toString());
        System.out.println("Измененный linkedlist: " + linkedlistFunc1.toString());
        System.out.println("Клон linkedlist : " + linkedlistClone.toString());
```

вывод:

```
== Тестирование toString() ==
Array: {(0,0; 1,5)(4,2; 7,0)(10,0; 13,5)}
LinkedList: {(0,0; 1,5)(4,2; 7,0)(10,0; 13,5)}

== Тестирование equals() ==
array1 == array2: true
linkedlist1 == linkedlist2: true
array1 == linkedlist1: true
array1 == array3: false

== Тестирование hashCode() ==
array1 hashCode: -309413269
array2 hashCode: -309413269
array3 hashCode: 559069802
linkedlist1 hashCode: -309413269
linkedlist2 hashCode: -309413269
linkedlist3 hashCode: 559069802

== Проверка согласованности hashCode и equals ==
array1 hashCode == array2 hashCode: true
linkedlist1 hashCode == linkedlist2 hashCode: true
array1 hashCode == linkedlist1 hashCode: true

== Проверка изменения hashCode() ==
Изначальный: -309413269
Измененный: 1922567029
```

```
== Тестирование clone() ==
Изначальный array: {(0,0; 1,5)(4,2; 7,0)(10,0; 13,5)}
Клон array: {(0,0; 1,5)(4,2; 7,0)(10,0; 13,5)}
Изначальный linkedlist: {(0,0; 1,5)(4,2; 7,0)(10,0; 13,5)}
Клон linkedlist : {(0,0; 1,5)(4,2; 7,0)(10,0; 13,5)}

== Проверка глубокого копирования ==
Измененный array: {(0,0; 10,0)(4,2; 7,0)(10,0; 13,5)}
Клон array: {(0,0; 1,5)(4,2; 7,0)(10,0; 13,5)}
Измененный linkedlist: {(0,0; 10,0)(4,2; 7,0)(10,0; 13,5)}
Клон linkedlist : {(0,0; 1,5)(4,2; 7,0)(10,0; 13,5)}
```