Лабораторная работа №5 по дисциплине «Объектно-ориентированное программирование»

Выполнил: Чичкин Данила Александрович

Группа: 6204-010302D

Оглавление

Задание 1

В классе FunctionPoint были переопределены методы toString(), equals(), hashCode() и clone() (см. скрин 1).

```
81    /** Возвращает строковое представление точки в виде "(x; y)". ...*/
86    @Override  haidyonish
87    public String toString() {
88        return "(" + x + "; " + y + ")";
89    }
90
91    /** Сравнивает текущую точку с другим объектом. ...*/
99    @Override  haidyonish *
100   public boolean equals(Object o) {
101       if (this == o) {
102           return true;
103       }
104       if (!(o instanceof FunctionPoint point)) {
105           return false;
106       }
107       return Double.compare(x, point.x) == 0 && Double.compare(y, point.y) == 0;
108   }
109
110   /** Возвращает хэш-код точки. ...*/
115   @Override  haidyonish
116   public int hashCode() {
117       long xBits = Double.doubleToLongBits(x);
118       long yBits = Double.doubleToLongBits(y);
119
120       int hash = Long.hashCode(xBits);
121       hash ^= Long.hashCode(yBits);
122
123       return hash;
124   }
125
126   /** Создаёт и возвращает копию точки. ...*/
131   @Override  haidyonish
132   public Object clone() { return new FunctionPoint(this); }
```

Скрин 1 – методы toString(), equals(), hashCode() и clone() класса FunctionPoint

Метод toString() возвращает строковое представление точки в виде (x; y), где x и y — координаты точки по соответствующим осям.

Метод equals() сравнивает текущий объект с другим объектом. Точки считаются равными, если переданный объект также является точкой и разности соответствующих координат по модулю меньше заданного машинного эпсилона. Такой подход обеспечивает корректное сравнение чисел с плавающей точкой.

Метод hashCode() вычисляет хэш-код точки на основе значений координат x и y. Для этого значения типа double переводятся в представление типа long, после чего используется побитовое исключающее ИЛИ (XOR), что позволяет учесть всё состояние объекта.

Метод clone() возвращает копию объекта точки. Так как класс не содержит ссылок на другие объекты, используется простое клонирование путём создания нового объекта с теми же значениями координат.

Задание 2

В классе ArrayTabulatedFunction были переопределены методы toString(), equals(), hashCode() и clone() (см. скрины 2, 3).

```
204     /** Возвращает строковое представление табулированной функции. ...*/
209     @Override  haidyonish *
210     public String toString() {
211         StringBuilder sb = new StringBuilder("{");
212         for (int i = 0; i < size; i++) {
213             sb.append(points[i]);
214             if (i < size - 1) {
215                 sb.append(", ");
216             }
217         }
218         sb.append("}");
219         return sb.toString();
220     }
221
222     /** Сравнивает текущую функцию с другим объектом. ...*/
230     @Override  haidyonish *
231     public boolean equals(Object o) {
232         if (this == o) {
233             return true;
234         }
235         if (!(o instanceof TabulatedFunction other)) {
236             return false;
237         }
238         if (size != other.getPointsCount()) {
239             return false;
240         }
241         if (o instanceof ArrayTabulatedFunction arrayFunc) {
242             for (int i = 0; i < size; i++) {
243                 if (!points[i].equals(arrayFunc.points[i])) {
244                     return false;
245                 }
246             }
247             return true;
248         }
249         for (int i = 0; i < size; i++) {
250             if (!points[i].equals(other.getPoint(i))) {
251                 return false;
252             }
253         }
254         return true;
255     }
```

Скрин 2 – методы toString(), equals() класса ArrayTabulatedFunction

Скрин 3 – методы hashCode(), clone() класса ArrayTabulatedFunction

Метод toString() формирует строковое представление табулированной функции в виде списка точек, заключённых в фигурные скобки. Каждая точка выводится в формате (x; y).

Метод equals() сравнивает текущую функцию с другим объектом. Функции считаются равными, если переданный объект реализует интерфейс TabulatedFunction, количество точек совпадает и все соответствующие точки равны. В случае, если переданный объект также является экземпляром ArrayTabulatedFunction, используется прямой доступ к массиву точек, что сокращает время выполнения метода.

Метод hashCode() вычисляет хэш-код функции с использованием операции XOR. В вычислении участвуют хэш-коды всех точек функции, а также количество точек. Это гарантирует различие хэш-кодов для функций с разным набором точек.

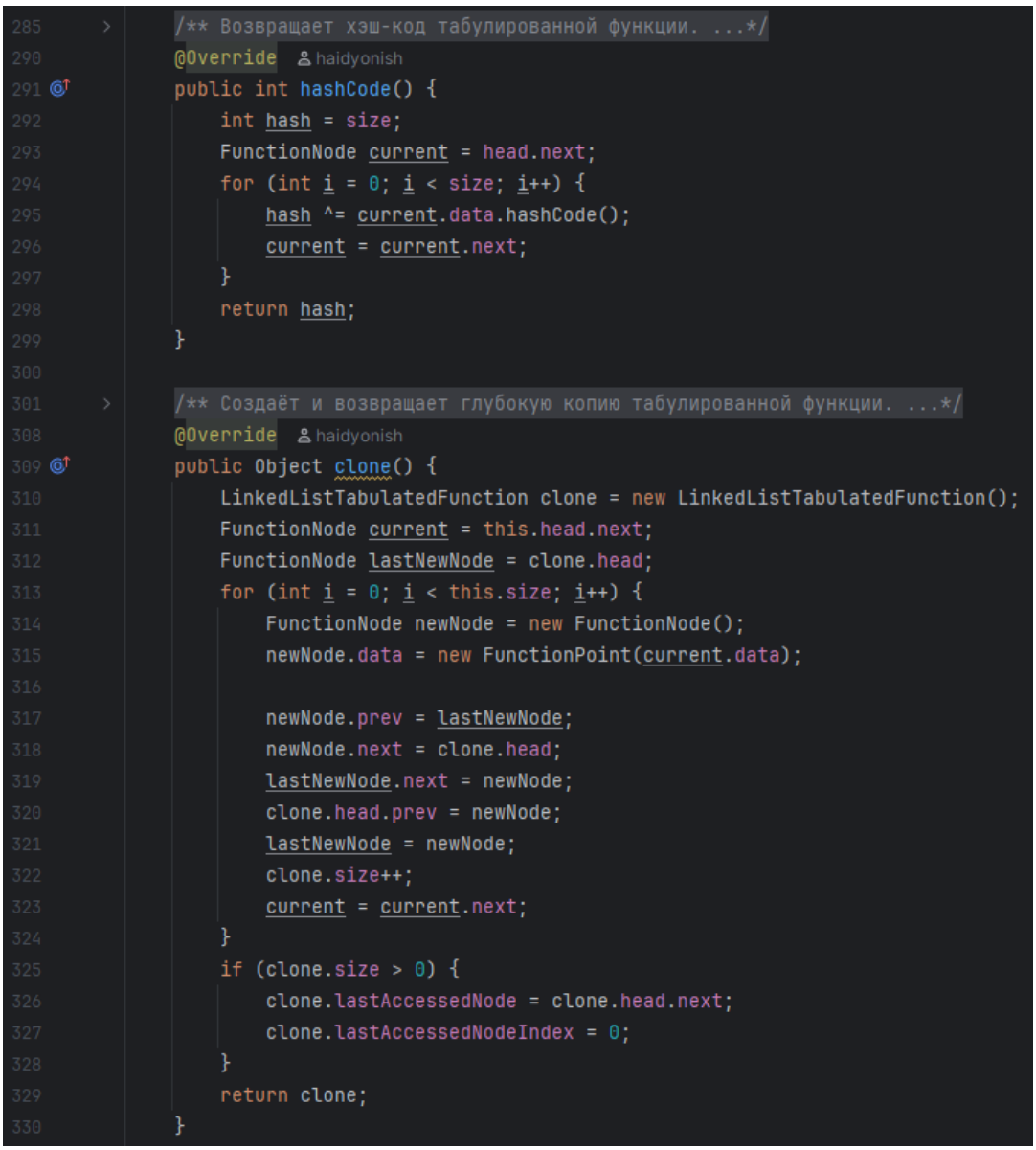
Метод clone() реализует глубокое клонирование объекта. Создаётся новый массив точек, каждая точка клонируется отдельно, после чего на основе этого массива создаётся новый объект табулированной функции.

Задание 3

В классе LinkedListTabulatedFunction были переопределены методы toString(), equals(), hashCode() и clone() (см. скрины 4, 5).

```
223     /** Возвращает строковое представление табулированной функции. ...*/
228     @Override  haidyonish
229     public String toString() {
230         StringBuilder sb = new StringBuilder("{");
231         FunctionNode current = head.next;
232         for (int i = 0; i < size; i++) {
233             sb.append(current.data);
234             if (i < size - 1) {
235                 sb.append(", ");
236             }
237             current = current.next;
238         }
239         sb.append("}");
240         return sb.toString();
241     }
242
243     /** Сравнивает текущую функцию с другим объектом. ...*/
251     @Override  haidyonish
252     public boolean equals(Object o) {
253         if (this == o) {
254             return true;
255         }
256         if (!(o instanceof TabulatedFunction other)) {
257             return false;
258         }
259         if (size != other.getPointsCount()) {
260             return false;
261         }
262         if (o instanceof LinkedListTabulatedFunction listFunc) {
263             FunctionNode n1 = this.head.next;
264             FunctionNode n2 = listFunc.head.next;
265
266             for (int i = 0; i < size; i++) {
267                 if (!n1.data.equals(n2.data)) {
268                     return false;
269                 }
270                 n1 = n1.next;
271                 n2 = n2.next;
272             }
273             return true;
274         }
275         FunctionNode current = head.next;
276         for (int i = 0; i < size; i++) {
277             if (!current.data.equals(other.getPoint(i))) {
278                 return false;
279             }
280             current = current.next;
281         }
282         return true;
283     }
```

Скрин 4 – методы toString(), equals() класса LinkedListTabulatedFunction.



```java
    /** Возвращает хэш-код табулированной функции. ...*/
    @Override  haidyonish
    public int hashCode() {
        int hash = size;
        FunctionNode current = head.next;
        for (int i = 0; i < size; i++) {
            hash ^= current.data.hashCode();
            current = current.next;
        }
        return hash;
    }

    /** Создаёт и возвращает глубокую копию табулированной функции. ...*/
    @Override  haidyonish
    public Object clone() {
        LinkedListTabulatedFunction clone = new LinkedListTabulatedFunction();
        FunctionNode current = this.head.next;
        FunctionNode lastNewNode = clone.head;
        for (int i = 0; i < this.size; i++) {
            FunctionNode newNode = new FunctionNode();
            newNode.data = new FunctionPoint(current.data);

            newNode.prev = lastNewNode;
            newNode.next = clone.head;
            lastNewNode.next = newNode;
            clone.head.prev = newNode;
            lastNewNode = newNode;
            clone.size++;
            current = current.next;
        }
        if (clone.size > 0) {
            clone.lastAccessedNode = clone.head.next;
            clone.lastAccessedNodeIndex = 0;
        }
        return clone;
    }
```

Скрин 5 – методы hashCode(), clone() класса LinkedListTabulatedFunction.

Метод toString() возвращает строковое представление функции в том же формате, что и для массивной реализации, последовательно обходя двусвязный циклический список узлов.

Метод equals() корректно работает при сравнении с любым объектом, реализующим интерфейс TabulatedFunction. Если переданный объект также является экземпляром LinkedListTabulatedFunction, сравнение выполняется напрямую по узлам списка, что позволяет сократить время работы метода.

Метод hashCode() реализован аналогично массивной версии. В вычислении участвует количество точек и хэш-коды всех точек функции, объединённые с помощью операции XOR.

Метод clone() выполняет глубокое клонирование объекта путём пересборки нового связного списка. Для каждого узла исходного списка создаётся новый узел и новая точка. При этом методы добавления элементов не используются, что позволяет избежать лишних операций и повысить производительность.

Задание 4

Для того чтобы все объекты типа TabulatedFunction были клонируемыми с точки зрения JVM, интерфейс TabulatedFunction был расширен интерфейсом Cloneable.

Также в интерфейс был добавлен метод clone(), который задаёт единый контракт клонирования для всех реализаций табулированных функций. Благодаря этому клонирование объектов возможно через ссылку на интерфейс TabulatedFunction, без приведения типов.

Задание 5

В классе Main была проведена проверка работы всех переопределённых методов (см. скрин 6).

```
===== СОЗДАНИЕ ФУНКЦИЙ =====

===== toString() =====
arrayFunc1:
{(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}
arrayFunc2:
{(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}
listFunc:
{(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}

===== equals() =====
arrayFunc1.equals(arrayFunc2): true
arrayFunc1.equals(listFunc): true
arrayFunc2.equals(listFunc): true

===== hashCode() =====
arrayFunc1.hashCode(): 1074266115
arrayFunc2.hashCode(): 1074266115
listFunc.hashCode(): 1074266115

Изменяем одну точку arrayFunc1 (y += 0.001)
arrayFunc1.hashCode() после изменения: 162683962
arrayFunc2.hashCode() (без изменений): 1074266115

===== clone() =====
arrayFunc2: {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}
hashCode arrayFunc2: 1074266115
arrayClone: {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}
hashCode arrayClone: 1074266115

listFunc: {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}
hashCode listFunc: 1074266115
listClone: {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}
hashCode listClone: 1074266115

Изменяем исходные объекты после клонирования
arrayFunc2:{(0.0; 100.0), (1.0; 2.0), (2.0; 3.0)}
hashCode arrayFunc2: 1067515907
arrayClone:{(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}
hashCode arrayClone: 1074266115

listFunc:{(0.0; 200.0), (1.0; 2.0), (2.0; 3.0)}
hashCode listFunc: 1066467331
listClone:{(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}
hashCode listClone: 1074266115

===== ПРОВЕРКА ЗАВЕРШЕНА =====
```

Скрин 6 – проверка работы методов toString(), equals(), hashCode() и clone().

Для объектов классов ArrayTabulatedFunction и LinkedListTabulatedFunction были выведены строковые представления, что позволило проверить корректность работы метода toString().

Метод equals() был проверен при сравнении одинаковых и различных объектов как одного, так и разных классов, реализующих интерфейс TabulatedFunction.

Для проверки hashCode() значения хэш-кодов были выведены в консоль. После незначительного изменения одной из координат точки было зафиксировано изменение хэш-кода, что подтверждает согласованность работы методов equals() и hashCode().

Метод clone() был проверен для обоих классов табулированных функций. После клонирования исходные объекты изменялись, при этом объекты-клоны оставались неизменными, что подтверждает корректность глубокого клонирования.

Хешкоды при изменении исходных объектов менялись, хешкоды объектов-клонов не менялись.