

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

**Выполнил:** Чичкин Данила Александрович

**Группа:** 6204-010302D

Оглавление

Задание 1 ..... 3

Задание 2 ..... 4

Задание 3 ..... 6

Задание 4 ..... 8

Задание 5 ..... 9

Задание 6 ..... **Ошибка! Закладка не определена.**

Задание 7 ..... **Ошибка! Закладка не определена.**

# 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
87 public String toString() {
88     return "(" + x + "; " + y + ")";
89 }
90
91 > /** Сравнивает текущую точку с другим объектом. ...*/
99 @Override
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
108     return Math.abs(x - point.x) < 1e-10 && Math.abs(y - point.y) < 1e-10;
109 }
110
111 > /** Возвращает хэш-код точки. ...*/
116 @Override
117 public int hashCode() {
118     long xBits = Double.doubleToLongBits(x);
119     long yBits = Double.doubleToLongBits(y);
120
121     int hash = Long.hashCode(xBits);
122     hash ^= Long.hashCode(yBits);
123
124     return hash;
125 }
126
127 > /** Создаёт и возвращает копию точки. ...*/
132 @Override
133 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 машинного эпсилона.

Метод `hashCode()` вычисляет хэш-код точки на основе значений координат `x` и `y`.

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
210 public String toString() {
211     StringBuilder sb = new StringBuilder("{");
212
213     for (int i = 0; i < size; i++) {
214         sb.append(points[i]);
215         if (i < size - 1) {
216             sb.append(", ");
217         }
218     }
219
220     sb.append("}");
221     return sb.toString();
222 }
223
224 > /** Сравнивает текущую функцию с другим объектом. ...*/
232 @Override
233 public boolean equals(Object o) {
234     if (this == o) {
235         return true;
236     }
237     if (!(o instanceof TabulatedFunction other)) {
238         return false;
239     }
240
241     if (size != other.getPointsCount()) {
242         return false;
243     }
244
245     if (o instanceof ArrayTabulatedFunction arrayFunc) {
246         for (int i = 0; i < size; i++) {
247             if (!points[i].equals(arrayFunc.points[i])) {
248                 return false;
249             }
250         }
251         return true;
252     }
253
254     for (int i = 0; i < size; i++) {
255         if (!points[i].equals(other.getPoint(i))) {
256             return false;
257         }
258     }
259
260     return true;
261 }
```

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

```

263 > /** Возвращает хэш-код табулированной функции. ...*/
268 @Override
269 public int hashCode() {
270     int hash = size;
271
272     for (int i = 0; i < size; i++) {
273         hash ^= points[i].hashCode();
274     }
275
276     return hash;
277 }
278
279 > /** Создаёт и возвращает глубокую копию табулированной функции. ...*/
284 @Override
285 public Object clone() {
286     FunctionPoint[] pointsCopy = new FunctionPoint[size];
287     for (int i = 0; i < size; i++) {
288         pointsCopy[i] = (FunctionPoint) points[i].clone();
289     }
290     return new ArrayTabulatedFunction(pointsCopy);
291 }

```

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.

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
229 @
230 public String toString() {
231     StringBuilder sb = new StringBuilder("{}");
232     FunctionNode current = head.next;
233     for (int i = 0; i < size; i++) {
234         sb.append(current.data);
235         if (i < size - 1) {
236             sb.append(", ");
237         }
238         current = current.next;
239     }
240     sb.append("}");
241     return sb.toString();
242 }
243 > /** Сравнивает текущую функцию с другим объектом. ...*/
251 @Override
252 @
253 public boolean equals(Object o) {
254     if (this == o) {
255         return true;
256     }
257     if (!(o instanceof TabulatedFunction other)) {
258         return false;
259     }
260     if (size != other.getPointsCount()) {
261         return false;
262     }
263     if (o instanceof LinkedListTabulatedFunction listFunc) {
264         FunctionNode n1 = this.head.next;
265         FunctionNode n2 = listFunc.head.next;
266
267         for (int i = 0; i < size; i++) {
268             if (!n1.data.equals(n2.data)) {
269                 return false;
270             }
271             n1 = n1.next;
272             n2 = n2.next;
273         }
274         return true;
275     }
276     FunctionNode current = head.next;
277     for (int i = 0; i < size; i++) {
278         if (!current.data.equals(other.getPoint(i))) {
279             return false;
280         }
281         current = current.next;
282     }
283     return true;
284 }
```

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

```

285 > /** Возвращает хэш-код табулированной функции. ...*/
290 @Override
291 public int hashCode() {
292     int hash = size;
293     FunctionNode current = head.next;
294     for (int i = 0; i < size; i++) {
295         hash ^= current.data.hashCode();
296         current = current.next;
297     }
298     return hash;
299 }
300
301 > /** Создаёт и возвращает глубокую копию табулированной функции. ...*/
308 @Override
309 public Object clone() {
310     LinkedListTabulatedFunction clone = new LinkedListTabulatedFunction();
311     FunctionNode current = this.head.next;
312     FunctionNode lastNewNode = clone.head;
313     for (int i = 0; i < this.size; i++) {
314         FunctionNode newNode = new FunctionNode();
315         newNode.data = new FunctionPoint(current.data);
316
317         newNode.prev = lastNewNode;
318         newNode.next = clone.head;
319         lastNewNode.next = newNode;
320         clone.head.prev = newNode;
321         lastNewNode = newNode;
322         clone.size++;
323         current = current.next;
324     }
325     if (clone.size > 0) {
326         clone.lastAccessedNode = clone.head.next;
327         clone.lastAccessedNodeIndex = 0;
328     }
329     return clone;
330 }

```

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)}
arrayClone:
{(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)}
listClone:
{(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}

Изменяем исходные объекты после клонирования
arrayFunc2 (изменён):
{(0.0; 100.0), (1.0; 2.0), (2.0; 3.0)}
arrayClone (должен остаться прежним):
{(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}

listFunc (изменён):
{(0.0; 200.0), (1.0; 2.0), (2.0; 3.0)}
listClone (должен остаться прежним):
{(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}

===== 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.