

Лабораторная работа №5

Выполнил: Мурзяков Егор 6204 – 010302D

Ход выполнения задания № 1

В классе “FunctionPoint” были переопределены следующие методы:

Метод “toString()” возвращает текстовое описание точки.

Метод “equals(Object o)” возвращает true только когда переданный объект также является точкой и его координаты совпадают с координатами текущего объекта.

Метод “hashCode()” возвращает значение хэш-кода для объекта точки. Для преобразования координат типа double в int использовался метод “Double.doubleToLongBits()”, который преобразует double в long без потерь. Затем каждый long разбивался на два int с помощью операции XOR между старшими и младшими 32 битами. Финальный хэш-код вычислялся как XOR между хэш-кодами обеих координат.

Метод “clone()” возвращает объект-копию точки.

Ход выполнения задания № 2

В классе “ArrayTabulatedFunction” были переопределены следующие методы:

Метод “toString()” возвращает описание табулированной функции, для этого был использован переопределенный метод “toString()” объекта “FunctionPoint” из предыдущего задания.

Метод “equals(Object o)” возвращает true только когда переданный объект также является табулированной функцией и её набор точек совпадает с текущей функцией. Если переданный объект является экземпляром класса "ArrayTabulatedFunction", используется оптимизация - прямое обращение к элементам массива точек. Для других табулированных функций используется универсальный подход через интерфейс.

Метод “hashCode()” возвращает значение хэш-кода, вычисляемое как XOR между количеством точек и хэш-кодами всех точек функции.

Метод “clone()” возвращает объект-копию табулированной функции. Поскольку функция содержит массив точек, клонирование является глубоким - создается новый массив с клонированными точками.

Ход выполнения задания № 3

В классе “LinkedListTabulatedFunction” были переопределены следующие методы:

Метод “toString()” возвращает описание табулированной функции в том же формате, что и для массива. Для этого используется обход связанного списка и поэлементное добавление строкового представления точек.

Метод “equals(Object o)” возвращает true только когда переданный объект также является табулированной функцией и её набор точек совпадает с текущей функцией. При сравнении с объектом типа “LinkedListTabulatedFunction” используется оптимизация - прямой обход узлов списка. Для других табулированных функций используется универсальный подход через интерфейс.

Метод “hashCode()” возвращает значение хэш-кода, вычисляемое как XOR между количеством точек и хэш-кодами всех точек функции. Обход точек осуществляется по связанному списку.

Метод “clone()” возвращает объект-копию табулированной функции. Для создания копии используем массив точек и передаем его в конструктор “LinkedListTabulatedFunction”, что обеспечивает глубокое клонирование без необходимости пересборки узлов списка вручную.

Ход выполнения задания № 4

В интерфейс “TabulatedFunction” было добавлено наследование от интерфейса “Cloneable”, чтобы сделать все объекты типа “TabulatedFunction” клонируемыми с точки зрения JVM.

В интерфейс был добавлен метод “clone()”, который должен возвращать объект-копию табулированной функции.

Ход выполнения задания № 5

Для тестирования методов были созданы объекты функций с некоторыми значениями (рис. 1).

```
// Создание тестовых данных
FunctionPoint[] points1 = {
    new FunctionPoint(_x: 0.0, _y: 1.0),
    new FunctionPoint(_x: 1.0, _y: 3.0),
    new FunctionPoint(_x: 2.0, _y: 5.0)
};

FunctionPoint[] points2 = {
    new FunctionPoint(_x: 0.0, _y: 1.0),
    new FunctionPoint(_x: 1.0, _y: 3.0),
    new FunctionPoint(_x: 2.0, _y: 5.0)
};

FunctionPoint[] points3 = {
    new FunctionPoint(_x: 0.0, _y: 1.0),
    new FunctionPoint(_x: 1.0, _y: 3.5), // Отличается от points1 и points2
    new FunctionPoint(_x: 2.0, _y: 5.0)
};

// Создание объектов функций
ArrayTabulatedFunction arrayFunc1 = new ArrayTabulatedFunction(points1);
ArrayTabulatedFunction arrayFunc2 = new ArrayTabulatedFunction(points2);
ArrayTabulatedFunction arrayFunc3 = new ArrayTabulatedFunction(points3);

LinkedListTabulatedFunction linkedFunc1 = new LinkedListTabulatedFunction(points1);
LinkedListTabulatedFunction linkedFunc2 = new LinkedListTabulatedFunction(points2);
LinkedListTabulatedFunction linkedFunc3 = new LinkedListTabulatedFunction(points3);
```

Рис. 1

1. Тестирование метода “toString()”

Было выведено строковое представление объектов “arrayFunc1” и “linkedFunc1” (рис. 2).

```
1. Тестирование toString():
arrayFunc1: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}
linkedFunc1: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}
```

Рис. 2

2. Тестирование метода “equals()”

Проверялось сравнение одинаковых и различающихся объектов, а также объектов разных классов (рис. 3).

```
2. Тестирование equals():  
arrayFunc1.equals(arrayFunc2): true  
arrayFunc1.equals(arrayFunc3): false  
linkedFunc1.equals(linkedFunc2): true  
linkedFunc1.equals(linkedFunc3): false  
arrayFunc1.equals(linkedFunc1): true  
linkedFunc2.equals(arrayFunc1): true
```

Рис. 3

3. Тестирование метода “hashCode()”

Выводились значения хэш-кодов для всех объектов и проверялась согласованность с методом “equals()”

```
3. Тестирование hashCode():
arrayFunc1.hashCode(): 1075576835
arrayFunc2.hashCode(): 1075576835
arrayFunc3.hashCode(): 1075314691
linkedFunc1.hashCode(): 1075576835
linkedFunc2.hashCode(): 1075576835
linkedFunc3.hashCode(): 1075314691

Согласованность equals и hashCode:
arrayFunc1.equals(arrayFunc2): true
arrayFunc1.hashCode() == arrayFunc2.hashCode(): true

arrayFunc1.equals(arrayFunc3): false
arrayFunc1.hashCode() == arrayFunc3.hashCode(): false
```

Рис. 4

4. Тестирование хэш-кода при незначительном изменении объекта (рис. 5).

```
4. Тестирование изменения объекта:
До изменения - arrayFunc3: {(0.0; 1.0), (1.0; 3.5), (2.0; 5.0)}
Хэш-код до изменения: 1075314691
После изменения - arrayFunc3: {(0.0; 1.0), (1.0; 3.501), (2.0; 5.0)}
Хэш-код после изменения: 161635386
```

Рис. 5

5. Тестирование глубокого копирования
Проверялось, что после клонирования изменение исходного объекта не влияет на клон (рис. 6).

```
5. Тестирование clone() и глубокого копирования:
arrayFunc1: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}
arrayClone: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}
arrayFunc1 == arrayClone: false
arrayFunc1.equals(arrayClone): true
После изменения arrayFunc1:
arrayFunc1: {(0.0; 999.0), (1.0; 3.0), (2.0; 5.0)}
arrayClone: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}
Клон не изменился: true

linkedFunc1: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}
linkedClone: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}
linkedFunc1 == linkedClone: false
linkedFunc1.equals(linkedClone): true
После изменения linkedFunc1:
linkedFunc1: {(0.0; 888.0), (1.0; 3.0), (2.0; 5.0)}
linkedClone: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}
Клон не изменился: true
```

Рис. 6