Лабораторная работа №5

Автор: Терин Ярослав

Группа: 6203-010302D

Задание 1

В классе FunctionPoint мы добавили конкретную реализацию методов. Для метода toString мы прописали возврат строки в формате "(" + x + "; " + y + ")", где x и y - это координаты точки. Это позволяет при выводе объекта видеть его координаты в понятном виде. Для метода equals мы написали код, который сначала проверяет совпадение ссылок через this == o, затем проверяет что объект не null и принадлежит тому же классу, и только после этого сравнивает координаты. Важным моментом было использование статического метода compareDouble из ArrayTabulatedFunction для сравнения double-значений с заданной точностью 1e-10, что решает проблему точности вычислений с плавающей точкой. В методе hashCode мы применили технику преобразования double в long через Double.doubleToLongBits, затем разбили каждое long-значение на два int-значения: младшие 4 байта получаем через (int)bits, а старшие 4 байта через (int)(bits >>> 32). После этого выполняем XOR между полученными хэшами координат. Метод clone() мы реализовали через вызов конструктора копирования new FunctionPoint, что создает новый объект с такими же координатами.

Задание 2

В этом классе мы реализовали toString через StringBuilder, который в цикле проходит по всем элементам массива points_arr и добавляет каждую точку в строку, разделяя их запятыми. Для equals мы написали две ветки логики: если переданный объект тоже ArrayTabulatedFunction, то сравниваем длины массивов и затем каждый элемент через equals, если же это другой тип TabulatedFunction, то используем getPointsCount и getPoint для поэлементного сравнения. HashCode реализован через инициализацию хэша количеством точек, а затем в цикле выполняем XOR над хэш-кодом каждой точки. Метод clone выполняет глубокое копирование: сначала вызывает super.clone, затем создает новый массив points_arr такой же длины, и в цикле клонирует каждую точку.

Задание 3-4

В LinkedListTabulatedFunction для toString мы использовали обход связанного списка: начинаем с head.getNext и проходим по всем узлам, пока не вернемся к head. В equals также две стратегии: для LinkedListTabulatedFunction сравниваем узлы напрямую, двигаясь по списку одновременно в обоих объектах; для других TabulatedFunction используем getPoint для сравнения. HashCode аналогичен массиву: начинаем с size, затем в цикле проходим по узлам и выполняем XOR с хэш-кодом точки каждого узла. Clone реализован через пересборку: создаем новый объект, инициализируем пустой список через initializeList(), затем проходим по исходному списку и для каждого узла создаем новый узел с клонированной точкой через addNodeToTail. Для интерфейса TabulatedFunction мы изменили его объявление на extends Function, Cloneable и добавили метод Object clone() в список методов интерфейса.

Задание 5

```
fun1: y = 4x
{(1.0; 4.0), (12.0; 48.0), (23.0; 92.0), (34.0; 136.0), (45.0; 180.0), (56.0; 224.0), (67.0; 268.0), (78.0; 312.0), (89.0; 356.0), (100.0; 400.0)}

fun2: Функция с значениями из массива
{(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0), (6.0; 12.0), (7.0; 14.0)}

fun3: log(exp)
{(1.0; 1.0), (2.0; 2.0), (3.0; 3.0000000000000004), (4.0; 4.0), (5.0; 4.999999999999999), (6.0; 6.0), (7.0; 6.999999999999999), (8.0; 7.999999999999998)

fun4: fun1 + fun3
{(1.0; 5.0), (2.0; 10.0), (3.0; 15.0), (4.0; 20.0), (5.0; 25.0), (6.0; 30.0), (7.0; 35.0), (8.0; 40.0), (9.0; 45.0), (10.0; 50.0), (11.0; 55.0)}

fun5: Функция с линейными значениями
{(1.0; 4.0), (12.0; 48.0), (23.0; 92.0), (34.0; 136.0), (45.0; 180.0), (56.0; 224.0), (67.0; 268.0), (78.0; 312.0), (89.0; 356.0), (100.0; 400.0)}

fun6: Копия fun4
{(1.0; 5.0), (2.0; 10.0), (3.0; 15.0), (4.0; 20.0), (5.0; 25.0), (6.0; 30.0), (7.0; 35.0), (8.0; 40.0), (9.0; 45.0), (10.0; 50.0), (11.0; 55.0)}

Равенство функций fun1 и fun2 (объекты разных классов)
Функции не равны

Равенство функций fun1 и fun3 (объекты одного класса)
Функции не равны

Равенство функций fun1 и fun5 (объекты разных классов)
Функции равны


Равенство функций fun4 и fun6 (объекты одного класса)
Функции равны

Хэш-код для fun1
2143289354

Хэш-код для fun2
2146435079

Хэш-код для fun3
-4194294

Хэш-код для fun4
2140438539

Хэш-код для fun5
2143289354

Хэш-код для fun6
2140438539

Изменили точку с индексом 2 на 0.005
{(1.0; 5.0), (2.0; 10.0), (3.0; 15.005), (4.0; 20.0), (5.0; 25.0), (6.0; 30.0), (7.0; 35.0), (8.0; 40.0), (9.0; 45.0), (10.0; 50.0), (11.0; 55.0)}

Измененный хэш-код для fun4
599553863
{(1.0; 5.0), (2.0; 10.0), (3.0; 15.0), (4.0; 20.0), (5.0; 25.0), (6.0; 30.0), (7.0; 35.0), (8.0; 40.0), (9.0; 45.0), (10.0; 50.0), (11.0; 55.0)}

fun7: Копия fun2
{(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0), (6.0; 12.0), (7.0; 14.0)}

Изменили исходную fun2
{(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0), (6.0; 12.0), (7.0; 14.0), (18.0; 20.0)}

Копия fun7 не изменилась
{(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0), (6.0; 12.0), (7.0; 14.0)}
```