Лабораторная работа №5

Задание на лабораторную работу

Расширить возможности классов, связанных с табулированными функциями, переопределив в них методы, унаследованные из класса `Object`.

Задание 1 *(1,2 балла)*

Переопределите в классе `FunctionPoint` следующие методы.

Метод `String toString()` должен возвращать текстовое описание точки. Например, оно может иметь следующий вид: `(1.1; -7.5)`, где `1.1` и -7.5 – абсцисса и ордината точки соответственно.

Метод `boolean equals(Object o)` должен возвращать истинное значение тогда и только тогда, когда переданный в метод объект также является точкой и его координаты в точности совпадают с координатами объекта, у которого вызывается метод.

Метод `int hashCode()` должен возвращать значение хэш-кода для объекта точки. Можете выбрать реализацию хэш-функции или воспользоваться простейшей реализацией, основанной на применении операции исключающего или. В этом случае значение хэш-кода рассчитывается как значение побитового исключающего или для набора значений, имеющих тип `int`. Этот набор значений должен включать в себя всю информацию, описывающую состояние объекта, т.е. в данном случае – два значения координат. Поскольку эти значения имеют тип `double`, необходимо без потерь перевести эту информацию к типу `int`, например, представив одно значение типа `double` (8 байт) как два значения типа `int` (4 байта и 4 байта). Сделать это можно с помощью метода `Double.doubleToLongBits()`, оператора побитового и (для выделения младших четырёх байтов) и оператора битового логического сдвига (для выделения старших четырёх байтов).

Метод `Object clone()` должен возвращать объект-копию для объекта точки. Поскольку точка не имеет ссылок на другие объекты, достаточно будет простого клонирования.

Задание 2 *(1,6 балла)*

Переопределите в классе `ArrayTabulatedFunction` следующие методы.

Метод `String toString()` должен возвращать описание табулированной функции. Например, оно может иметь следующий вид: `{(0.0; 1.2), (1.0; 3.8), (2.0; 15.2)}`, где в круглых скобках указываются координаты точек, задающих табулированную функцию.

Метод `boolean equals(Object o)` должен возвращать истинное значение тогда и только тогда, когда переданный в метод объект также является табулированной функцией (реализует интерфейс `TabulatedFunction`) и её набор точек в точности совпадает с набором точек функции, у которой вызывается метод. В случае если переданный в метод объект является экземпляром класса `ArrayTabulatedFunction`, время работы метода должно быть сокращено за счёт прямого обращения к элементам состояния переданного в метод объекта.

Метод `int hashCode()` должен возвращать значение хэш-кода для объекта табулированной функции. Можете выбрать реализацию хэш-функции или воспользоваться простейшей реализацией, основанной на применении операции исключающего или. В этом случае значение хэш-кода рассчитывается как значение побитового исключающего или для набора значений, имеющих тип `int`. В данном случае в этот набор значений входят значения хэш-кодов всех точек табулированной функции, а также количество точек в функции.

Последнее нужно для того, чтобы значения хэш-кода были различны для функций, отличающихся наличием нулевой точки (например, {(-1; 1), (0; 0), (1, 1)} и {(-1; 1), (1, 1)}).

Метод `Object clone()` должен возвращать объект-копию для объекта табулированной функции. Поскольку табулированная функция ссылается на другие объекты, клонирование должно быть глубоким.

Задание 3 *(1,6 балла)*

Аналогично, переопределите методы `toString()`, `equals()`, `hashCode()` и `clone()` в классе `LinkedListTabulatedFunction`. При написании методов учтите следующие особенности.

Метод `equals()` также должен корректно работать при сравнении с любым объектом типа `TabulatedFunction`, а при сравнении с объектом типа `LinkedListTabulatedFunction` время работы метода должно быть сокращено за счёт возможности прямого обращения к полям переданного объекта.

Клонирование в методе `clone()` тоже должно быть глубоким, однако классическое глубокое клонирование в данном случае не совсем разумно. Действительно, если сделать объекты класса `FunctionNode` клонируемыми, после их клонирования значения полей ссылок придётся изменить (т.к. они будут ссылаться на объекты из исходного списка), и значение ссылающегося на объект точки поля тоже придётся изменить (т.к. его нужно будет заменить клоном объекта точки). Т.е. клонирование этих объектов просто не имеет смысла, поэтому проще окажется «пересобрать» новый объект списка, причём сделать это проще без использования методов добавления в список, т.к. это приведёт к выполнению большого количества нерезультативных операций и заметно скажется на скорости выполнения программы.

Задание 4 *(0,2 балла)*

Сделайте так, чтобы все объекты типа `TabulatedFunction` были клонируемыми с точки зрения JVM и внесите метод `clone()` в этот интерфейс.

Задание 5 *(0,4 балла)*

Проверьте работу написанных методов.

Проверьте работу метода `toString()` для объектов типов `ArrayTabulatedFunction` и `LinkedListTabulatedFunction`, выведя строковое представление объектов в консоль.

Проверьте работу метода `equals()`, вызывая его для одинаковых и различающихся объектов одинаковых и различающихся классов.

Проверьте работу метода `hashCode()`, выведя в консоль его значения для всех использованных объектов. Убедитесь в согласованности работы методов `equals()` и `hashCode()`. Также попробуйте незначительно изменить один из объектов (например, изменить одну из координат одной из точек на несколько тысячных) и проверьте, как изменится значение хэш-кода объекта.

Проверьте работу метода `clone()` для объектов обоих классов табулированных функций. Убедитесь, что произведено именно глубокое клонирование: для этого после клонирования измените исходные объекты и проверьте, что объекты-клоны не изменились.