Макаров Дмитрий Александрович
магистрант
ФГБОУ ВО «Армавирский государственный
педагогический университет»
г. Армавир, Краснодарский край
Коновалова Виктория Александровна
учитель обществознания
МАОУ – СОШ №7 им. Г.К. Жукова
г. Армавир, Краснодарский край

РЕАЛИЗАЦИЯ МЕТОДА EQUALS () В JAVA

Аннотация: *в данной статье рассматриваются особенности и требования реализации метода equals ().*

Ключевые слова: *программирование, метод equals, сеть Интернет, веб-технологии.*

Языки программирования постоянно эволюционируют и с середины 20 века. Действительную популярность язык программирования приобрел только в 1990-х годах. Расцвет данной парадигмы пришлось на развитие глобальной сети интернет в сфере веб-технологий.

В Java для проверки равнозначности двух объектов применяется метод equals () из класса Object. А поскольку метод equals () реализован в классе Object, то в нем определяется только следующее: ссылаются ли переменные на один и тот же объект [3, с. 12].

Однако этого не всегда достаточно, поэтому определим ряд особенностей и требований для правильной реализации метода equals ().

1. Рефлексивность. При вызове x.equals(x) по любой ненулевой ссылке x должно возвращаться логическое значение true.

2. Симметричность. При вызове x.equals(y) по любым ссылкам x и y должно возвращаться логическое значение true тогда и только тогда, когда при вызове y equals (x) возвращается логическое значение true.

3. Транзитивность. Если при вызовах x.equals(y) и y.equals(z) по любым ссылкам x, y и z возвращается логическое значение true, то и при вызове x.equals(z) возвращается логическое значение true.

4. Согласованность. Если объекты, на которые делаются ссылки x и y, не изменяются, то при повторном вызове x.equals(y) должно возвращаться то же самое значение.

5. При вызове x.equals(null) по любой непустой ссылке x должно возвращаться логическое значение false.

Обоснованность приведенных выше правил не вызывает сомнений. Так, совершенно очевидно, что результаты проверки не должны зависеть от того, делается ли в программе вызов x.equals(y) или y.equals(x).

В стандартной библиотеке Java содержится более 150 реализаций метода equals (). В одних из них применяются различные сочетания операции instance of, вызовы метода get Class () и фрагменты кода, предназначенные для обработки исключения Class Cast Exception, а в других не выполняется практически ни каких действий [2, с. 18].

Ниже приведены рекомендации для создания приближающегося к идеалу метода equals().

1. Присвойте явному параметру имя *other Object*. Впоследствии его тип нужно будет привести к типу другой переменной под названием *other*.

2. Проверьте, одинаковы ли ссылки *this* и *other Object*, следующим образом:

if (this == other Object) return true.

Это выражение составлено лишь в целях оптимизации проверки. Ведь намного быстрее проверить одинаковость ссылок, чем сравнивать поля объектов.

3. Выясните, является ли ссылка *other Object* пустой (null), как показано ниже. Если она является таковой, следует возвратить логическое значение *false*. Эту проверку нужно сделать обязательно.

if (other Object == null) return false.

4. Сравните классы *this* и *other Object*. Если семантика проверки может измениться в подклассе, воспользуйтесь методом *get Class ()* следующим образом:

if (get Class()!= other Object. Get Class()) return false.

Если одна и та же семантика остается справедливой для всех подклассов, произведите проверку с помощью операции *instance of* следующим образом:

if (! (other Object instance of Class Name)) return false.

5. Приведите тип объекта *other Object* к типу переменной требуемого класса, как показано ниже.

Имя Класса other = (Имя Класса) other Object.

6. Сравните все поля, как показано ниже. Для полей примитивных типов служит операция ==, а для объектных полей – метод Objects. equals (). Если все поля двух объектов совпадают, возвращается логическое значение true, а иначе – логическое значение false.

Если вы переопределяете в подклассе метод equals(), в него следует включить вызов super.equals(other).

Таким образом, если проверка на равнозначность реализована в подклассе, правило симметричности требует использовать метод get Class(), если проверка производится средствами суперкласса, можно выполнить операцию instance of. В этом случае возможна ситуация, когда два объекта разных классов будут признаны равнозначными.

Список литературы

1. Class Object [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html

2. Cay S. Horstmann, Gary Cornell Core Java, Volume I-Fundamentals // Ninth Edition. – Prentice Hall, 2012.

3. Matt Weisfeld. The Object-Oriented Thought Process // Fourth Edition. – Addison-Wesley, 2013.

4. История развития диалога человека с ЭВМ // fb.ru [Электронный ресурс]. – Режим доступа: http://m.fb.ru/article/45551/protsedurnoe-programmirovanie-istoriya-razvitiya-dialoga-cheloveka-s-evm