Coder

17 уровень

23 сентября 2015 58222 47

Перегрузка методов equals() и hashCode() в Java

Переопределение методов equals() и hashCode() в Java

`Equals` и `hashCode` являются фундаментальными методами объявленные в классе `Object` и содержатся в стандартных библиотеках Java.

Метод `equals()` используется для сравнения объектов, а `hashCode` - для генерации целочисленного кода объекта.

извлечению объектов в `HashMap`. Метод `equal` также используются для обеспечения хранения только уникальных объектов в `HashSet` и других `Set` реализациях, а также в любых других случаях, когда нужно сравнивать объекты.

Реализация по умолчанию метода `equals()` в классе `java.lang.Object` сравнивает ссылки на адреса в памяти, которые хранят переменные, и возвращает `true` только в том случае, если адреса совпадают, другими словами переменные ссылаются на один и тот же объект.

Java рекомендует переопределять методы `equals()` и `hashCode()`, если предполагается, что сравнение должно осуществляться в соответсвии с естественной логикой или бизнес-логикой. Многие классы в стандартных библиотеках Java переопределяет их, например в классе `String` переопределяется `equals` таким образом, что возвращается `true`, если содержимое двух сравниваемых объектов одинаковое. В классе-обертке `Integer` метод `equal` переопределяется для выполнения численного сравнения, и так далее.

Так как `HashMap` и `HashTable` в Java полагаются на методы `equals()` и `hashCode()` для сравнения своих `key` и `values`, то Java предлагает следующие правила для переопределения этих методов:

1. Рефлексивность: Объект должен равняться себе самому.
2. Симметричность: если `a.equals(b)` возвращает `true`, то `b.equals(a)` должен тоже вернуть `true`.
3. Транзитивность: если `a.equals(b)` возвращает `true` и `b.equals(c)` тоже возвращает `true`, то `c.equals(a)` тоже должен возвращать `true`.
4. Согласованность: повторный вызов метода `equals()` должен возвращать одно и тоже значение до тех пор, пока какое-либо значение свойств объекта не будет изменено. То есть, если два объекта равны в Java, то они будут равны пока их свойства остаются неизменными.
5. Сравнение `null`: объект должны быть проверен на `null`. Если объект равен `null`, то метод должен вернуть `false`, а не `NullPointerException`. Например, `a.equals(null)` должен вернуть `false`.

Соглашение между equals и hashCode в Java

2. Если объекты не равны по результатам выполнения метода `equals`, тогда их `hashcode` могут быть как одинаковыми, так и разными. Однако для повышения производительности, лучше, чтобы разные объекты возвращали разные коды.

Как переопределять метод equals в Java

1.
```
@Override
public boolean equals(Object obj) {
/*1. Проверьте*/
 if (obj == this) {
 /*и верните */ return true;
          }
```

2. Проверьте объект на `null`, а также проверьте, чтобы объекты были одного типа. Не делайте проверку с помощью `instanceof` так как такая проверка будет возвращать `true` для подклассов и будет работать правильно только в случае если ваш класс объявлен как `immutable`. Вместо этого можно использовать `getClass()`;

```
if (obj == null || obj.getClass() != this.getClass()) {
            return false;
}
```

3. Объявите переменную типа, который вы сравниваете, и приведите `obj` к этому типу. Потом сравнивайте каждый атрибут типа начиная с численных атрибутов (если имеются) потому что численные атрибуты проверяются быстрей. Сравнивайте атрибуты с помощью операторов И и ИЛИ (так называемые `short-circuit logical operators`) для объединения проверок с другими атрибутами.

```
Person guest = (Person) obj;
        return id == guest.id && (firstName == guest.firstName ||
            (firstName != null && firstName.equals(guest.getFirstNam
                && (lastName == guest.lastName || (lastName != null
}
```

Полный пример переопределения метода equals в Java

```
/** * Person class with equals and hashcode implementation in Java * @au
public class Person {
    private int id;
```

```
6      public int getId() { return id; }
7      public void setId(int id) { this.id = id;}
8
9      public String getFirstName() { return firstName; }
10     public void setFirstName(String firstName) { this.firstName = firstN
11     public String getLastName() { return lastName; }
12     public void setLastName(String lastName) { this.lastName = lastName
13     @Override
14     public boolean equals(Object obj) {
15     if (obj == this) {
16         return true;
17     }
18     if (obj == null || obj.getClass() != this.getClass()) {
19         return false;
20     }
21
22     Person guest = (Person) obj;
23     return id == guest.id
24         && (firstName == guest.firstName
25             || (firstName != null &&firstName.equals(guest.getFirstName(
26             || (lastName != null && lastName .equals(guest.getLastName()
27             ));
28     }
29     @Override
30     public int hashCode() {
31     final int prime = 31;
32     int result = 1;
33     result = prime * result + ((firstName == null) ? 0 : firstName.hashC
34         ((lastName == null) ? 0 : lastName.hashCode()); return result;
35     }
36 }
```

Распространенные ошибки при переопределении equals в Java

1. Вместо того, чтобы переопределять метод `equals (Override)` программист перегружает его `(Overload)` Синтаксис метода `equals()` в классе `Object` определен как `public boolean equals(Object obj)`, но многие программисты ненароком перегружают метод: `public boolean equals(Person obj)` - вместо `Object` в качестве аргумента используют имя своего класса (напр. Person). Эту

сделает это корректно. Однако, если вы положите ваш объект в коллекцию, например `ArrayList` и вызовите метод `contains()`, работа которого основана на методе `equals()`, то метод `contains` не сможет обнаружить ваш объект.

2. При переопределении метода `equals()` не проверять на `null` переменные, что в конечном итоге заканчивается `NullPointerException` при вызове `equals()`. Ниже представлен корректный код.

```
1  firstname == guest.firstname || (firstname != null &&
2       firstname.equals(guest.firstname));
```

3. Третья распространенная ошибка это не переопределять метод `hashCode()`, а только `equals()`. Вы обязаны переопределять оба метода `equals()` и `hashCode()` в Java. Метод `hashCode` используется в `hash` -коллекциях(например `HashSet`), и чем меньше будет коллизий (одинаковый код при разных объектах) тем эффективнее эти коллекции будут работать с объектами вашего класса.

4. Последняя распространенная ошибка программистов в том, что при переопределении метода `equals()` не сохраняется соответствие между методами `equals()` и `compareTo()`, что является неформальным требованием для избежания хранения дубликатов в `Set (SortedSet, TreeSet)`.

Подсказки как писать в Java метод equals

1. Большинство IDE такие как NetBeans, Eclipse и IntelliJ IDEA обеспечивают поддержку генерации методов `equals()` и `hashCode()`. В Eclipse нажмите правую кнопку -> source -> `generate equals()` и `hashCode()`.

2. Если в классе есть уникальный бизнес-ключ, то будет достаточно сделать проверку только на равенство этих полей. Как в нашем примере “id” - уникальный номер для каждого Person.

3. При переопределении `hashCode()` в Java удостоверьтесь в использовании всех полей, что были использованы в методе `equals()`.

4. `String` и классы-оболочки такие как `Integer`, `Float` и `Double` переопределяют метод `equals()`, но `StringBuffer` не переопределяет.

6. При сравнении `String` объектов используйте `equals()` вместо оператора `==`.
7. Два объекта которые логически равны, но загружены из разных `ClassLoader` не могут быть равными. Помните, что проверка с помощью `getClass()` вернет `false` если класс-загрузчик разный.
8. Используйте `@Override` аннотацию также для метода `hashCode`, так как это предупреждает неуловимые ошибки, например возвращаемое значение метода `int`, однако некоторые программисты возвращают `long`.

P.S. Уважаемые коллеги! Мне оказалась полезной эта статья при решении задач 21 го уровня! Желаю удачи при разборе данной темы, пользуйтесь переводом!

Я надеюсь, что вы мне поможете поднять мой рейтинг, так как сейчас я даже не могу оставлять комментарий на этом форуме. Всем огромное спасибо!

Оригинал статьи

Я некоторые моменты опустил в связи с нехваткой свободного времени, однако перевел все, что необходимо знать для решения задач 21о уровня.
