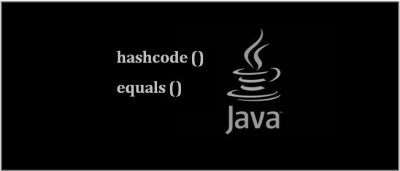
**Учимся определять equals() и hashCode();**



Привет. Не для кого наверное уже не секрет, что в Java есть два взаимосвязанных метода это equals() и hashCode() которые надо переопределять, либо не переопределять, если переопределять, то оба сразу.

Поясню для новичков и тех кто может быть забыл.

**Метод equals()**

Метод equals() используется для сравнения двух объектов, самый часто встречаемый вариант это сравнение строк, что-то типа этого

if (avatarPath.equals(DEFAULT\_AVATAR\_PATH)) {  
 // ...  
}

Метод equals() определен в классе Object (базовом классе для всех классов Java). Его реализация по-умолчанию сравнивает ссылки на объекты в памяти.

public boolean equals(Object o) {  
 return this == o;  
}

Этот метод советуется переопределять в своих классах если требуется сравнение объектов вашего класса. Более того, почти во всех классах Java Core этот метод переопределен.

Метод hashCode()

Этот метод возвращает hash code объекта, нужно это для использования внутри таких структур как Hashtable и HashMap ну и так далее. Если вы переопределяете equals() то вы и обязаны переопределить hashCode() потому что эти два метода взаимосвязаны. Одно из правил определения этих метедов гласит

*Если объекты равны (тоесть equals(Object o) == true) то и хэш-коды этих объектов должны быть равны. То есть obj1.equals(obj2) == true? то и obj1.hashCode() == obj2.hashCode()*

Правило выше не в коем случае не говорит о том, что теперь объекты можно сравнивать по хэш коду, лучше так же использовать метод equals(). Реализация по-умолчанию метода hashCode() написана на C, если вы посмотрите на этот метод в классе Object, то увидите следующее

public native int hashCode();

native означает что метод этот реализован не на языке Java и проброшен в класс Object через JNI. Что возвращает этот метод по-умолчанию сказать сложно, потому что это зависит от виртуальной машины.

Давайте определимся с правилами написания методов equals() и hashCode().

Легче всего писать метод hashCode(), потому что начиная с JDK 7 появился объект Objects в котором есть метод для создания hash кода на основе полей класса. Выглядит это примерно так

public class Main {  
 private String name = "Vjacheslav";  
 private int age = 19;  
 private String liveCity = "St.Petersburg";  
  
 @Override  
 public int hashCode() {  
 return Objects.hashCode(name, age, liveCity);  
 }  
}

На основе полей класса будет сгенерирован уникальный хэш код. Как я и говорил, с hashCode() все просто, но есть еще и equals(). Давайте определим его.

public class Main {  
 private String name = "Vjacheslav";  
 private int age = 19;  
 private String liveCity = "St.Petersburg";  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) { return true; }  
 if (o == null || o.getClass() != getClass()) { return false; }  
  
 Main main = (Main) o;  
  
 return main.name == this.name &&  
 main.age == this.age &&  
 main.liveCity == this.liveCity;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hashCode(name, age, liveCity);  
 }  
}

Давайте теперь рассмотрим метод equals(). Первым делом мы сраниваем объект o с текущим объектом (ссылки)

if (this == o) { return true; }

Это типа некоторая оптимизация, чтобы не производить сравнение значений и так далее если объект сравнивается сам с собой.

Далее

if (o == null || o.getClass() != getClass()) { return false; }

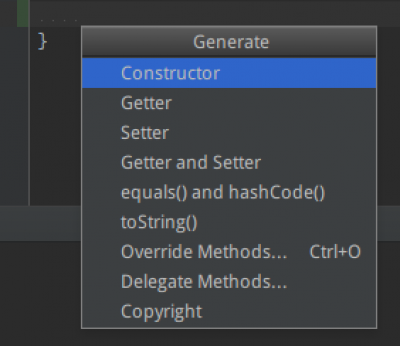
Есть соглашение о том, что equals() не должен валиться с NPE если объект o является null'ом. По-этому проверка на null должна быть обязательной. Далее идет сравнение классов, тут неважно какой инстанс объекта у вас создан, getClass() будет одинаковым, вы можете еще увидеть вместо сравнения классов следующую запись

if (o == null || !(o instanceof Main)) { return false; }

Это в принципе одно и тоже, мне нравится вариант больше со сравнением классов. А дальше мы сравниваем значения полей классов

Main main = (Main) o;  
  
 return main.name == this.name &&  
 main.age == this.age &&  
 main.liveCity == this.liveCity;

Эта секция тоже обязательна. В принципе эти методы за вас может генерировать IDE, в Intellij просто жмете сочетание клавишь alt + insert на windows и linux и cmd + n на MacOS. У вас выпаден такая менюшка



И в этом списке выбирайте просто equals() and hashCode(), после чего вам сгенерируется оба эти метода точно так как я показал выше.

Но знать как это работает обязательно. Так как на собеседовании с листком и ручкой вам IDE не поможет :(.

Ну а если интересно, если вы за чистоту в коде и вам не хочется чтобы в ваших классах явно были определены методы equals и hashCode то можете воспользоваться либой [lombok](https://megahub.me/go?url=https://projectlombok.org/). Она на основе аннотаций генерирует методы equals и hashCode, а так же toString и геттеры с сеттерами. Прошлый пример выглядел бы так.

import lombok.EqualsAndHashCode;  
  
@EqualsAndHashCode  
public class Main {  
 private String name = "Vjacheslav";  
 private int age = 19;  
 private String liveCity = "St.Petersburg";  
}