Вопросы по теме Equals и HashCode:

1. Отличие equals() от == ?
2. Условия для переопределения equals()?
3. Правила переопределения equals()?
4. Для чего нужен метод hashCode()?
5. Как реализованы hashCode() и equals() в классе Object?
6. Что будет, если переопределить equals, но не переопределить hashCode?
7. Как пользоваться интерфейсом Comparable?
8. Как пользоваться интерфейсом Comparator?

Метод equals().

Метод equals(Object obj) - указывает, является ли какой-либо другой объект «равным» этому. В классе Object определен следующим образом:

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

Т.е. в стандартном исполнении метод equals() сравнивает ссылки через оператор == . Но для определения равенства объектов такое сравнение не годится, т.к. объекты могут быть равны и иметь разные ссылки. Поэтому программист должен переопределять метод equals(), сравнивая в нем только те параметры, по которым можно утверждать, что два объекта равны.

Условия (основные правила) для переопределения метода equals():

1. рефлексивность: x.equals(x) должно вернуть true;
2. симметричность: если x.equals(y) возвращает true, то и y.equals(x) должно вернуть true;
3. транзитивность: если x.equals(y) и y.equals(z) возвращает true, то и x.equals(z) должно вернуть true;
4. непротиворечивость: если несколько раз вызывать x.equals(x), то возвращаемое значение должно всегда оставаться одним и тем же (true либо false);
5. для null метод equals() всегда возвращает false.

Порядок переопределения метода equals():

public Boolean equals(Object obj) {

1. obj == this return true (если ссылки равны, то и объекты равны)
2. obj == null || obj.getClass() != this.getClass() return true (если один из объектов null, то объекты не равны; если у объектов разные типы, то объекты не равны)
3. MyClass myClass = (MyClass) obj (преобразуем аргумент к нужному нам типу)
4. выполняем сравнение всех значимых полей объектов (напр, field == myClass.field)

}

При переопределении метода equals() нужно обязательно переопределять метод hashcode(). Иначе могут возникнуть проблемы, например, у HashMap пара «ключ-значение» может потеряться (одинаковые объекты с разными хэш-кодами считаются разными объектами, в итоге значение не перезапишется, а сохранится в новой ячейке).

Метод hashcode().

Метод hashcode() – возвращает значение хэш-кода для объекта. В классе Object определен следующим образом:

public native int hashCode();

По умолчанию реализация equals() возвращает номер ячейки памяти, где объект сохраняется. Ключевое слово native указывает, что метод реализован на другом языке программирования (не java).

Метод hashcode() предназначен для повышения производительности сравнения двух объектов (числа сравнивать быстрее, чем объекты):

1. если два объекта равны, то хэш-коды этих объектов равны;
2. если хэш-коды равны, то объекты не обязательно равны;
3. если hashcode() несколько раз вызвать на одном и том же объекте, то всегда должно возвращаться одно и то же значение.

Для генерации хэш-кода лучше выбирать уникальные поля, например id. Если поля задействованы при вычислении hashcode(), то они должны быть задействованы при реализации equals().

Коллизия – ситуация, когда разные объекты имеют одинаковые хэш-коды. Может возникнуть при неоптимальной реализации либо из-за ограниченного (ограниченно диапазоном типа данных int) количества возвращаемых значений метода hashcode().

Интерфейс java.lang.Comparable.

Интерфейс Comparable предназначен для придания объектам возможности сравнивать себя друг с другом (у объектов класса, реализующих данный интерфейс, появляется свойство сравнивать себя с другими объектами этого же класса). Создает наиболее естественный порядок сортировки (т.е. если это числа, то по возрастанию / убыванию, если строки, то лексикографически). Он содержит всего один метод:

public int compareTo(T o);

Порядок реализации интерфейса Comparable:

1. реализуем интерфейс Comparable<T> (T – тип объекта, который будет сравниваться);
2. реализуем метод compareTo(T o) (возвращает отрицательное целое число, ноль или положительное целое число, в зависимости от: this объект меньше, равен или больше указанного объекта соответственно).

Интерфейс java.util.Comparator.

Интерфейс Comparator предназначен для сравнения объектов. При его использовании мы создаем объект, который сравнивает два других объекта между собой. Он содержит два метода:

int compare(T o1, T o2);

boolean equals(Object obj);

Порядок реализации интерфейса Comparator:

1. реализуйте интерфейс Comparator<T>;
2. реализуем метод compare(T o1, T o2);
3. используем получившийся объект для сравнения других объектов.