Сравнение сущностей

- Объект-значение
- Сущность

Сравнение сущностей обладает своими особенностями, которые значительно влияют на код. В этом уроке вы узнаете, чем сравнение сущностей отличается от обычных Java-объектов.

Объект-значение

Разберем на примере. Ниже мы рассмотрим определение модели `City`, которая не считается сущностью. В таком случае проверка равенства выполняется на основе значения полей, в этом случае — имени. Если объект A представляет конкретный город, то другой объект с тем же городом — это тот же самый объект A:

```
import lombok.AllArgsConstructor;
import lombok.Getter;

@AllArgsConstructor
@Getter
class City {
    private String name;
}

var city1 = new City("London");
var city2 = new City("London");

// Равны, потому что это один и тот же город
city1.getName() == city2.getName();
```

В таком случае метод `equals()` легко реализуется с помощью Lombok:

```
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.EqualsAndHashCode;
```

```
@AllArgsConstructor
@Getter
@EqualsAndHashCode
class City {
    private String name;
}

var city1 = new City("London");
var city2 = new City("London");

city1.equals(city2); // true
```

Для объектов такого рода существует термин Value Object. Это объекты, которые представляют собой какое-то значение — например, адрес. Они сравниваются на основе своего значения или значений набора полей.

Сущность

В отличие от обычных Java-объектов, у сущностей есть **идентификация**. Другими словами, у каждой сущности есть свой уникальный идентификатор, который обычно представлен числом. Этот идентификатор однозначно указывает, с какой сущностью мы имеем дело независимо от того, совпадает ли содержимое других полей у этих сущностей:

```
import lombok.AllArgsConstructor;
import lombok.Getter;

@AllArgsConstructor
@Getter
class User {
    private Long id;
    private String firstName;
    private String lastName;
}

var user1 = new User(1, "John", "Travolta");
var user2 = new User(2, "John", "Travolta");

// Не равны, потому что это разные люди
user1.getId() != user2.getId(); // true
```

```
var user3 = new User(1, "Mike", "Travolta");

// Равны, потому что это один и тот же человек
user1.getId() == user3.getId(); // true
```

Равенство или неравенство остальных данных этих объектов не имеет значения. Объекты равны только тогда, когда совпадают идентификаторы. Если идентификаторы не совпадают, то объекты не равны.

Реализовать такое поведение можно с помощью той же аннотации `@EqualsAndHashCode`. Единственное отличие от предыдущего примера состоит в том, что мы явно указываем необходимость сравнения только по идентификатору:

```
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.EqualsAndHashCode;

@AllArgsConstructor
@Getter
@EqualsAndHashCode(of = {"id"})
class User {
    private String id;
    private String firstName;
    private String lastName;
}

var user1 = new User(1, "John", "Travolta");
var user2 = new User(2, "John", "Travolta");

user1.equals(user2); // false

var user3 = new User(1, "Mike", "Travolta");

// Равны, потому что это один и тот же человек
user1.equals(user3); // true
```

Далее →