**1. Introduction**

So sánh các đối tượng là một tính năng thiết yếu của ngôn ngữ lập trình hướng đối tượng.

Trong hướng dẫn này, chúng ta sẽ xem xét một số tính năng của ngôn ngữ Java cho phép chúng ta so sánh các đối tượng. Ngoài ra, chúng tôi sẽ xem xét các tính năng như vậy trong các thư viện bên ngoài.

## 2. == and != Operators

Hãy bắt đầu với các toán tử == và! = Có thể cho biết hai đối tượng Java tương ứng có giống nhau hay không.

### 2.1. Primitives

(1==1)=> true

**Integer** a = **new** **Integer**(1);

(1 == a)=> true

### 2.2. Objects

**Integer** a = **new** **Integer**(1);

**Integer** b = **new** **Integer**(1);

assertThat(a == b).isFalse(); không bằng nhau

Nhưng gán b=a thì bằng

**Integer** a = **new** **Integer**(1);

**Integer** b = a;

assertThat(a == b).isTrue();

**Integer** a = Integer.valueOf(1);

**Integer** b = Integer.valueOf(1);

assertThat(a == b).isTrue();

Trong trường hợp này, chúng được coi là giống nhau. Điều này là do phương thức valueOf () lưu trữ Số nguyên trong bộ nhớ cache để tránh tạo quá nhiều wrapper objects có cùng giá trị. Do đó, phương thức trả về cùng một thể hiện Số nguyên cho cả hai lần gọi.

Cũng đúng đối với String

assertThat("Hello!" == "Hello!").isTrue();

Nhưng khi được tạo bằng new String() thì ko giống nhau

2 đối tượng null

assertThat(null == null).isTrue();

các đối tượng khác null so sánh == vs null => false

## 3. Object#equals Method

Phương thức này được định nghĩa trong lớp Object để mọi đối tượng Java kế thừa nó.

Theo mặc định, việc triển khai của nó so sánh các địa chỉ bộ nhớ đối tượng, vì vậy nó hoạt động giống như toán tử ==. Tuy nhiên, chúng ta có thể ghi đè phương thức này để xác định ý nghĩa của bình đẳng đối với các đối tượng của chúng ta.

**Integer** a = **new** **Integer**(1);

**Integer** b = **new** **Integer**(1);

assertThat(a.equals(b)).isTrue();

Chúng ta có thể override method equals() để so sánh 2 đối tượng.

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Person that = (Person) o;

return firstName.equals(that.firstName) &&

lastName.equals(that.lastName);

}

Trong Java, toán tử **==** và phương thức **equals()** là hai cách khác nhau để so sánh giá trị của các đối tượng. Tuy nhiên, chúng có một số sự khác nhau như sau:

1. Toán tử **==** so sánh xem hai biến có tham chiếu đến cùng một đối tượng hay không, trong khi phương thức **equals()** so sánh giá trị của hai đối tượng.
2. Toán tử **==** chỉ làm việc với các kiểu dữ liệu nguyên thủy (primitive data types) và các đối tượng (objects) trong Java, trong khi phương thức **equals()** có thể được ghi đè (override) bởi các lớp con (subclasses) để cung cấp một cách tùy chỉnh để so sánh giá trị của các đối tượng.
3. Toán tử **==** là toán tử so sánh bằng (equality operator), trong khi phương thức **equals()** là phương thức kiểm tra sự bằng nhau (equality method).
4. Khi so sánh các đối tượng tham chiếu null, toán tử **==** trả về true nếu cả hai biến đều tham chiếu đến null, trong khi phương thức **equals()** sẽ trả về false nếu đối tượng tham chiếu null.
5. Toán tử **==** nhanh hơn phương thức **equals()** vì nó chỉ kiểm tra tham chiếu, trong khi phương thức **equals()** phải thực hiện nhiều thao tác để kiểm tra giá trị của đối tượng.

Vì vậy, nếu bạn muốn so sánh giá trị của hai đối tượng, bạn nên sử dụng phương thức **equals()**. Tuy nhiên, nếu bạn chỉ muốn kiểm tra xem hai biến có tham chiếu đến cùng một đối tượng hay không, bạn có thể sử dụng toán tử **==**.

## 4. Objects#equals Static Method

Không thể sử dụng null. equals(a)

Nhưng **equals() method of the Objects helper class solves that problems.**

Person joe = new Person("Joe", "Portman");

Person joeAgain = new Person("Joe", "Portman");

Person natalie = new Person("Natalie", "Portman");

assertThat(Objects.equals(joe, joeAgain)).isTrue();

assertThat(Objects.equals(joe, natalie)).isFalse();

## 5. Comparable Interface

Sử dụng để sắp xếp list object.

The Comparable interface is generic

Chỉ có 1 method compareTo(), nhận đối số là generic, return về int

Giá trị trả về là âm nếu giá trị này thấp hơn đối số, 0 nếu chúng bằng nhau và dương ngược lại.

Ví dụ khi so sánh đối tượng Person bằng last name:

public class Person implements Comparable<Person> {

//...

@Override

public int compareTo(Person o) {

return this.lastName.compareTo(o.lastName);

}

}