**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);

}

## 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);

}

}