Toán tử instanceof trong Java

Toán tử instanceof trong Java được sử dụng để kiểm tra xem đối tượng có là instance của kiểu cụ thể: lớp hoặc lớp con hoặc interface hay không. Toán tử instanceof trong Java còn được biết đến như là toán tử so sánh kiểu bởi vì nó so sánh instance với kiểu. Nó trả về true hoặc false. Nếu chúng ta áp dụng toán tử instanceof với bất cứ biến nào mà có giá trị null, thì nó trả về false.

Ví dụ đơn giản về toán tử instanceof trong Java

Ví dụ sau kiểm tra xem đối tượng có phải là lớp hiện tại không.

```
class Simple1{
 public static void main(String args[]){
 Simple1 s=new Simple1();
 System.out.println(s instanceof Simple);//true
 }
}
```

Một đối tượng của kiểu lớp con cũng là một kiểu của lớp cha. Ví dụ, nếu Dog kế thừa Animal thì đối tượng Dog có thể được tham chiếu bởi hoặc lớp Dog hoặc lớp Animal.

Ví dụ khác về toán tử instanceof trong Java

```
class Animal{}
class Dog1 extends Animal
{//Dog ke thua Animal

 public static void main(String args[]){
 Dog1 d=new Dog1();
 System.out.println(d instanceof Animal);//true
 }
}
```

Ví dụ instanceof trong Java với biến mà có giá trị null

Nếu chúng ta áp dụng toán tử instanceof với bất cứ biến nào mà có giá trị null, thì nó trả về false. Trong ví dụ sau, chúng ta áp dụng toán tử instanceof với biến có giá trị null.

```
class Dog2{
```

```
 public static void main(String args[]){
  Dog2 d=null;
  System.out.println(d instanceof Dog2);//false
 }
}
```

Downcasting với toán tử instanceof trong Java

Khi kiểu lớp con tham chiếu tới đối tượng của lớp cha, thì đó là downcasting. Nếu chúng ta thực hiện nó trực tiếp, Compiler sẽ cho một lỗi biên dịch. Nếu bạn thực hiện bởi typecasting (ép kiểu), thì **ClassCastException** được ném tại runtime. Nhưng nếu chúng ta sử dụng toán tử instanceof thì downcasting là có thể.

```
Dog d=new Animal();// gay ra loi tai thoi gian bien dich
```

Nếu bạn thực hiện downcasting bởi typecasting (ép kiểu), thì **ClassCastException** được ném tại runtime.

```
Dog d=(Dog)new Animal();
//Bien dich thanh cong nhung ClassCastException bi nem tai runtime
```

Và có thể thực hiện downcasting với instanceof

Trong ví dụ sau, bạn sẽ thực hiện downcasting với toán tử instanceof trong Java.

```
class Animal { }


class Dog3 extends Animal {
  static void method(Animal a) {
    if(a instanceof Dog3){
       Dog3 d=(Dog3)a; //day la downcasting
       System.out.println("Bay gio downcasting duoc thuc hien");
    }
  }


  public static void main (String [] args) {
    Animal a=new Dog3();
    Dog3.method(a);
```

```
  }


 }
```

Downcasting cũng có thể được thực hiện mà không sử dụng toán tử instanceof, như sau:

```
class Animal { }
class Dog4 extends Animal {
  static void method(Animal a) {
       Dog4 d=(Dog4)a;//Day la downcasting
       System.out.println("downcasting duoc thuc hien");
  }
   public static void main (String [] args) {
    Animal a=new Dog4();
    Dog4.method(a);
  }
}
```

Bạn cùng nhìn lại thật kĩ vào ví dụ trên, đối tượng thực sự được tham chiếu bởi a, là một đối tượng của lớp Dog. Vì thế nếu chúng ta thực hiện downcasting, thì việc đó là ổn. Nhưng điều gì xảy ra nếu chúng ta viết:

```
Animal a=new Animal();
Dog.method(a);
//Bay gio la ClassCastException nhung khong trong truong hop cua toan tu instanceof
```

Sự sử dụng thực sự của instanceof trong Java

Cùng theo dõi ví dụ sau để hiểu sự sử dụng thực sự của từ khóa instanceof trong Java:

```
interface Printable{}
class A implements Printable{
public void a(){System.out.println("Phuong thuc a");}
}
class B implements Printable{
public void b(){System.out.println("Phuong thuc b");}
}
```

```
class Call{
void invoke(Printable p)
{ //Day la upcasting
if(p instanceof A){
A a=(A)p;//Day la Downcasting
a.a();
}
if(p instanceof B){
B b=(B)p;//Day la Downcasting
b.b();
}

}
}//Phan cuoi cua lop Call

class Test4{
public static void main(String args[]){
Printable p=new B();
Call c=new Call();
c.invoke(p);
}
}
```

Kết quả là:

```
Output: b method
```