Instance Initializer Block trong Java

Instance Initializer Block được sử dụng để khởi tạo thành viên dữ liệu instance. Nó chạy mỗi khi đối tượng của lớp được tạo. Sự khởi tạo của biến instance có thể là trực tiếp nhưng có thể là các hoạt động phụ được thực hiện trong khi khởi tạo biến instance trong Instance Initializer Block.

Câu hỏi: Sự sử dụng của Instance Initializer Block là gì trong khi chúng ta có thể trực tiếp gán một giá trị trong thành viên dữ liệu instance? Ví dụ:

```
class Bike{
    int speed=100;
}
```

Tại sao sử dụng Instance Initializer Block?

Giả sử bạn phải thực hiện một số hoạt động trong khi gán giá trị tới thành viên dữ liệu instance, ví dụ như cho vòng lặp để điền vào một mảng phức hợp hoặc xử lý lỗi, …

Bạn theo dõi ví dụ đơn giản sau về Instance Initializer Block mà thực hiện việc khởi tạo:

```
class Bike7{
    int speed;

    Bike7(){System.out.println("speed is "+speed);}

    {speed=100;}

    public static void main(String args[]){
    Bike7 b1=new Bike7();
    Bike7 b2=new Bike7();
    }
}
```

Instance Initializer Block hay Constructor được triệu hồi trước nhất?

```
class Bike8{
    int speed;
```

```
    Bike8(){System.out.println("constructor duoc trieu hoi");}

    {System.out.println("instance initializer block duoc trieu hoi");}

    public static void main(String args[]){
    Bike8 b1=new Bike8();
    Bike8 b2=new Bike8();
    }
}
```

Trong ví dụ trên, có vẻ như là Instance Initializer Block được triệu hồi đầu tiên, nhưng**KHÔNG**. Instance Initializer Block được triệu hồi tại thời điểm tạo đối tượng. Java Compiler sao chép Instance Initializer Block trong Constructor sau lệnh super() đầu tiên. Vì thế được triệu hồi đầu tiên là Constructor.

Ghi chú: Java Compiler sao chép code của Instance Initializer Block trong mỗi constructor.

cho Instance Initializer Block

Dưới đây là một số qui tắc chính cho Instance Initializer Block:

- Được tạo khi instance của lớp được tạo.
- Được triệu hồi sau khi constructor của lớp cha được triệu hồi (ví dụ: sau lời gọi super() constructor).
- Được triệu hồi sau khi constructor của lớp cha được triệu hồi (ví dụ: sau lời gọi super() constructor).

Ví dụ chương trình mà Instance Initializer Block được triệu hồi sau super()

```
class A{
A(){
System.out.println("constructor cua lop cha duoc trieu hoi");
}
}
```

```
class B2 extends A{
B2(){
super();
System.out.println("constructor cua lop con duoc trieu hoi");
}

{System.out.println("instance initializer block duoc trieu hoi");}

public static void main(String args[]){
B2 b=new B2();
}
}
```

Ví dụ khác

```
class A{
A(){
System.out.println("constructor cua lop cha duoc trieu hoi");
}
}

class B3 extends A{
B3(){
super();
System.out.println("constructor cua lop con duoc trieu hoi");
}

B3(int a){
super();
System.out.println("constructor cua lop con duoc trieu hoi "+a);
}

{System.out.println("instance initializer block duoc trieu hoi");}
```

```
public static void main(String args[]){
B3 b1=new B3();
B3 b2=new B3(10);
}
}
```

Chương trình Java trên sẽ cho kết quả

```
Output:constructor cua lop cha duoc trieu hoi
       instance initializer block duoc trieu hoi
       constructor cua lop con duoc trieu hoi
       constructor cua lop cha duoc trieu hoi
       instance initializer block duoc trieu hoi
       constructor cua lop con duoc trieu hoi 10
```