Từ khóa static trong Java

Từ khóa static trong Java được sử dụng chính để quản trị bộ nhớ. Chúng ta có thể áp dụng từ khóa static với biến (cũng được gọi là biến lớp, biến class), phương thức (cũng được gọi là phương thức lớp), khối, các lớp được lặp. Từ khóa static thuộc về lớp chứ không thuộc về instance (sự thể hiện) của lớp.

Biến static trong Java

Khi bạn khai báo một biến là static, thì biến đó được gọi là biến tĩnh, hay biến static.

Biến static có thể được sử dụng để tham chiếu thuộc tính chung của tất cả đối tượng (mà không là duy nhất cho mỗi đối tượng), ví dụ như tên công ty của nhân viên, tên trường học của các sinh viên, …

Biến static lấy bộ nhớ chỉ một lần trong Class Area tại thời gian tải lớp đó.

Lợi thế của biến static

Biến static giúp bộ nhớ chương trình của bạn được sử dụng hiệu quả hơn (tiết kiệm bộ nhớ).

Tìm hiểu vấn đề xảy ra khi không có biến static

```
class Student{
     int rollno;
     String name;
     String college="BachKhoa";
}
```

Giả sử có 500 sinh viên trong trường đại học, bây giờ instance của các thành viên dữ liệu sẽ lấy bộ nhớ mỗi khi đối tượng được tạo. Tất cả sinh viên có rollno và name duy nhất vì thế instance của thành viên dữ liệu là tốt. Ở đây, college là thuộc tính chung của tất cả đối tượng. Nếu chúng ta tạo nó là static, thì trường này sẽ chỉ lấy bộ nhớ một lần.

Ghi chú: Thuộc tính static trong Java được chia sẻ tới tất cả đối tượng.

Ví dụ về biến static trong Java

```
//Chuong trinh vi du ve bien static trong Java


class Student8{
```

```
   int rollno;
   String name;
   static String college ="BachKhoa";

   Student8(int r,String n){
   rollno = r;
   name = n;
   }
 void display (){System.out.println(rollno+" "+name+" "+college);}

 public static void main(String args[]){
 Student8 s1 = new Student8(111,"Hoang");
 Student8 s2 = new Student8(222,"Thanh");

 s1.display();
 s2.display();
 }
}
```

Chương trình Counter mà không sử dụng biến static

Trong ví dụ, chúng ta tạo một biến instance có tên count mà được tăng lên trong constructor. Khi biến instance này lấy bộ nhớ tại thời điểm tạo đối tượng, mỗi đối tượng sẽ có bản sao của biến instance đó, nếu nó được tăng lên, nó sẽ không phản ánh các đối tượng khác. Vì thế mỗi đối tượng sẽ có giá trị 1 trong biến count.

```
class Counter{
int count=0; //se lay bo nho (memory) khi bien instance duoc tao

Counter(){
count++;
System.out.println(count);
}

public static void main(String args[]){
```

```
Counter c1=new Counter();
Counter c2=new Counter();
Counter c3=new Counter();

 }
}
```

Chương trình counter với biến static trong Java

Như bạn đã thấy ở trên, biến static sẽ lấy bộ nhớ chỉ một lần, nếu bất cứ đối tượng nào thay đổi giá trị của biến static, nó sẽ vẫn ghi nhớ giá trị của nó.

```
class Counter2{
static int count=0; //se lay bo nho chi mot lan và giu lai gia tri cua no

Counter2(){
count++;
System.out.println(count);
}

public static void main(String args[]){

Counter2 c1=new Counter2();
Counter2 c2=new Counter2();
Counter2 c3=new Counter2();

 }
}
```

Phương thức static trong Java

Nếu bạn áp dụng từ khóa static với bất cứ phương thức nào, thì phương thức đó được gọi là phương thức static.

- Một phương thức static thuộc lớp chứ không phải đối tượng của lớp.

- Một phương thức static có thể được triệu hồi mà không cần tạo một instance của một lớp.
- Phương thức static có thể truy cập thành viên dữ liệu static và có thể thay đổi giá trị của nó.

Ví dụ về phương thức static trong Java

```
//Chuong trinh thay doi thuoc tinh chung cua tat ca doi tuong (truong static).


class Student9{
     int rollno;
     String name;
     static String college = "BachKhoa";

     static void change(){
     college = "QuocGia";
     }

     Student9(int r, String n){
     rollno = r;
     name = n;
     }

     void display (){System.out.println(rollno+" "+name+" "+college);}

    public static void main(String args[]){
    Student9.change();

    Student9 s1 = new Student9 (111,"Hoang");
    Student9 s2 = new Student9 (222,"Thanh");
    Student9 s3 = new Student9 (333,"Nam");

    s1.display();
    s2.display();
    s3.display();
    }
```

```
}
```

Ví dụ khác về phương thức static mà thực hiện phép tính toán thông thường

```
//Chuong trinh lay cube (gia tri lap phuong) cua so da cho boi phuong thuc static


class Calculate{
  static int cube(int x){
  return x*x*x;
  }


  public static void main(String args[]){
  int result=Calculate.cube(5);
  System.out.println(result);
  }
}
```

Một số hạn chế cho phương thức static

Có hai hạn chế chính cho phương thức static. Đó là:

- Phương thức static không thể sử dụng thành viên dữ liệu non-static hoặc gọi trực tiếp phương thức non-static.
- Từ khóa this và super không thể được sử dụng trong ngữ cảnh static.

```
class A{
 int a=40;//non static


 public static void main(String args[]){
  System.out.println(a);
 }
}
```

Chạy chương trình trên sẽ cho kết quả là Compile Time Error.

Câu hỏi: Tại sao phương thức main trong Java là static?

Bởi vì đối tượng là không cần thiết để gọi phương thức static nếu nó là phương thức non-static, JVM đầu tiên tạo đối tượng và sau đó gọi phương thức main() mà có thể gây ra vấn đề về cấp phát bộ nhớ bộ nhớ phụ.

Khối static trong Java

Được sử dụng để khởi tạo thành viên dữ liệu static. Nó được thực thi trước phương thức main tại thời gian tải lớp. Dưới đây là ví dụ về khối static trong Java:

```
class A2{
  static{System.out.println("Khoi static duoc trieu hoi");}
  public static void main(String args[]){
   System.out.println("Hello main");
  }
}
```

Câu hỏi: Chúng ta có thể thực thi một chương trình mà không có phương thức main()?

Có, một trong các cách đó là khối static trong phiên bản trước của JDK, không trong JDK 1.7.

```
class A3{
  static{
  System.out.println("Khoi static duoc trieu hoi");
  System.exit(0);
  }
}
```