Access Modifier trong Java

Có hai loại Modifier trong Java, đó là: Access Modifier và Non-access Modifier. Access Modifer trong Java xác định phạm vi có thể truy cập của thành viên dữ liệu, phương thức, constructor hoặc lớp. Có 4 loại Access Modifier là: private, default, protected, và public.

- Default: Truy cập trong nội bộ package
- Private: Truy cập trong nội bộ lớp
- Public: Thành phần công khai, truy cập tự do từ bên ngoài
- Protected: Thành phần được bảo vệ, bị hạn chế truy nhập từ bên ngoài

Ngoài ra còn có nhiều Non-access Modifier như static, abstract, synchronized, native, volatile, transient, … Chương này chúng ta sẽ tìm hiểu về Access Modifier.

Bạn cùng theo dõi bảng sau để có cái nhìn sơ lược về các loại Access Modifier trong Java:

Access Modifier	Bên trong lớp	Bên trong package	Bên ngoài package chỉ bởi lớp con	Bên ngoài package
private	C	K	K	K
default	C	C	K	K
protected	C	C	C	K
public	C	C	C	C

Private Access Modifier trong Java

Các phương thức, biến và constructor mà được khai báo private chỉ có thể được truy cập trong chính lớp được khai báo đó.

Private Access Modifier là chế độ truy cập mang tính hạn chế nhất. Lớp và interface không thể là private.

Các biến được khai báo private có thể được truy cập bên ngoài lớp nếu phương thức public getter có mặt trong lớp đó.

Sử dụng Private Access Modifier trong Java là cách chủ yếu để một đối tượng bao đóng chính nó và ẩn dữ liệu với bên ngoài.

Trong ví dụ dưới đây, chúng ta tạo hai lớp A và Simple. Lớp A chứa thành viên dữ liệu private và phương thức private. Chúng ta đang truy cập các thành viên private này từ bên ngoài lớp, và điều này dẫn đến một Compile time error:

```
class A{
private int data=40;
private void msg(){System.out.println("Hello java");}
}

public class Simple{
 public static void main(String args[]){
   A obj=new A();
   System.out.println(obj.data);//Compile Time Error
   obj.msg();//Compile Time Error
   }
}
```

Qui tắc cho Private Access Modifier trong Java

Nếu bạn tạo bất cứ constructor nào là private, bạn không thể tạo instance (sự thể hiện) của lớp đó từ bên ngoài lớp. Ví dụ:

```
class A{
private A(){}//private constructor
void msg(){System.out.println("Hello java");}
}
public class Simple{
 public static void main(String args[]){
```

```
   A obj=new A();//Compile Time Error
 }
}
```

Default Access Modifier trong Java

Default Access Modifier nghĩa là chúng ta không khai báo một cách rõ ràng một Access Modifier cho một lớp, trường, phương thức, ... Nói cách khác, nếu bạn không sử dụng bất cứ Modifier nào, thì theo mặc định nó được xem như là default. Default Modifier là chỉ có thể truy cập bên trong package.

Một biến hoặc phương thức được khai báo mà không có bất kỳ Access Modifier nào là có sẵn cho mọi lớp trong cùng package. Các trường này trong một interface là hoàn toàn public static final và các phương thức trong một interface là public theo mặc định.

Ví dụ

Trong ví dụ này, chúng ta tạo hai package là pack và mypack. Chúng ta đang truy cập lớp A từ bên ngoài package của nó. Khi lớp A không là public, thì nó không thể được truy cập từ bên ngoài package.

```
//Luu duoi dang A.java
package pack;
class A{
  void msg(){System.out.println("Hello");}
}
//Luu duoi dang B.java
package mypack;
import pack.*;
class B{
  public static void main(String args[]){
   A obj = new A();//Compile Time Error
   obj.msg();//Compile Time Error
  }
}
```

Trong ví dụ trên, phạm vi của lớp A và phương thức msg() của nó là default, vì thế nó không thể được truy cập từ bên ngoài package.

Protected Access Modifier trong Java

Protected Access Modifier là có thể truy cập bên trong package và bên ngoài package nhưng chỉ thông qua tính kế thừa. Protected Access Modifier có thể được áp dụng trên thành viên dữ liệu. Nó không thể được áp dụng trên lớp. Các biến, phương thức và constructor, mà được khai báo protected trong một lớp cha (superclass), chỉ được truy cập bởi các lớp cha trong package khác hoặc bất kỳ lớp nào bên trong package đó của lớp được protected.

Protected Access Modifier không thể được áp dụng cho lớp và interface. Các phương thức và trường có thể được khai báo protected, tuy nhiên, các phương thức và trường trong một interface không thể được khai báo là protected.

Chế độ protected cung cấp cho lớp phụ cơ hội để sử dụng phương thức hoặc biến helper, trong khi ngăn cản một lớp không liên quan từ việc cố gắng sử dụng nó.

Ví dụ

Trong ví dụ này, chúng ta tạo hai package là pack và mypack. Một lớp A của pack package là public, vì thế có thể được truy cập từ bên ngoài package. Nhưng phương thức msg của package này được khai báo là protected, vì thế nó có thể được truy cập từ bên ngoài lớp nhưng chỉ thông qua tính kế thừa.

```
//Luu duoi dang A.java
package pack;
public class A{
protected void msg(){System.out.println("Hello");}
}
//Luu duoi dang B.java
package mypack;
import pack.*;

class B extends A{
  public static void main(String args[]){
   B obj = new B();
   obj.msg();
```

```
  }
}
```

Public Access Modifier trong Java

Public Access Modifier là có thể truy cập ở bất cứ đâu. Nó có phạm vi rộng nhất trong tất cả Modifier. Một lớp, phương thức, constructor, interface, ... được khai báo public có thể được truy cập từ bất cứ lớp nào khác. Do đó, các trường, phương thức và khối được khai báo bên trong một lớp public có thể được truy cập từ bất kỳ lớp nào trong thế giới Java.

Tuy nhiên, nếu lớp public chúng ta đang cố gắng truy cập là trong một package khác, thì lớp public này vẫn cần để được import.

Bởi vì tính kế thừa lớp, tất cả phương thức và biến của một lớp được kế thừa bởi các lớp phụ của nó.

Ví dụ

```
//Luu duoi dang A.java

package pack;
public class A{
public void msg(){System.out.println("Hello");}
}
//Luu duoi dang B.java

package mypack;
import pack.*;

class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

Access Modifier trong Java với Ghi đè phương thức

Nếu bạn đang ghi đè bất cứ phương thức nào, phương thức được ghi đè (ví dụ được khai báo trong lớp con) phải không nhiều giới hạn.

```
class A{
protected void msg(){System.out.println("Hello java");}
}


public class Simple extends A{
void msg(){System.out.println("Hello java");}//Compile Time Error
 public static void main(String args[]){
   Simple obj=new Simple();
   obj.msg();
   }
}
```

Default Modifier là nhiều giới hạn hơn protected. Đó là lý do tại sao đây là compile time error.

Access Modifier và tính kế thừa trong Java

Các qui tắc sau là bắt buộc cho các phương thức được kế thừa trong Java:

- Các phương thức được khai báo public trong một lớp cha cũng phải là public trong tất cả lớp phụ.
- Các phương thức được khai báo protected trong một lớp cha phải hoặc là protected hoặc public trong các lớp phụ; chúng không thể là private.
- Các phương thức được khai báo mà không có điều khiển truy cập (không sử dụng modifier nào) có thể được khai báo private trong các lớp phụ.
- Các phương thức được khai báo private không được kế thừa, do đó không có qui tắc nào cho chúng.