Bài: Các loại phạm vi truy cập trong lập trình hướng đối tượng

Xem bài học trên website để ủng hộ Kteam: Các loại phạm vi truy cập trong lập trình hướng đối tượng

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Trong bài trước, Kteam có nhắc đến cho các bạn về CLASS TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG. Hôm nay hãy đi sâu về **Phạm vi truy cập trong lập trình hướng đối tượng**.

Nội dung

Để đọc hiểu bài này, tốt nhất các bạn nên có kiến thức cơ bản về các phần sau:

- CÁC BIẾN TRONG JAVA.
- CÁC KIỂU DỮ LIỆU TRONG JAVA.
- CÁC HẠNG TOÁN TỬ TRONG JAVA
- CẤU TRÚC RẼ NHÁNH TRONG JAVA
- VÒNG LẶP WHILE TRONG JAVA
- VÒNG LẶP FOR TRONG JAVA
- MẢNG TRONG JAVA
- VÒNG LẶP FOR-EACH TRONG JAVA
- VAI TRÒ BREAK, CONTINUE TRONG VÒNG LẶP JAVA
- SWITCH TRONG JAVA
- LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- CLASS TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Bài này chúng ta sẽ tìm hiểu những vấn đề sau:

- Phạm vi truy cập là gì? Package là gì?
- Các loại phạm vi truy cập

Phạm vi truy cập là gì? Package là gì?

Phạm vi truy cập (access modifiers) là xác định độ truy cập phạm vi vào dữ liệu của các thuộc tính, phương thức hoặc class.

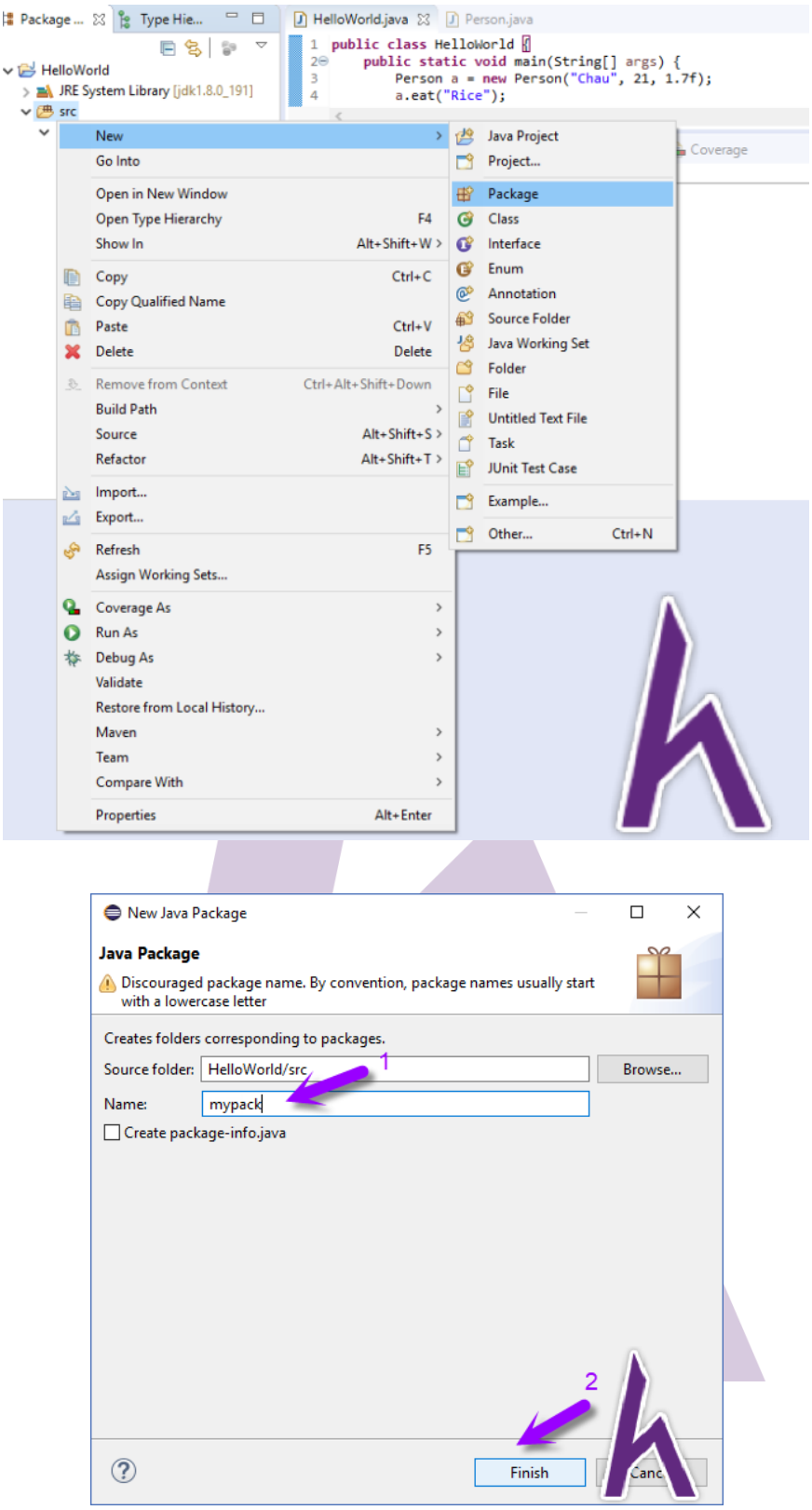
Package (gói) là nhóm các **class**, **interface** hoặc các **package** con liên quan lại với nhau. Việc dùng package dùng để nhóm các class liên quan với nhau thành thư viện như thư viện swing, awt,...Ngoài ra, mục đích của package ngăn cản xung đột đặt tên, điều kiện truy cập, thuận tiện tìm kiếm và lưu trữ.

Các loại phạm vi truy cập

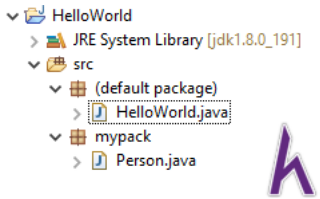
Có 4 loại phạm vi truy cập:

- Private
- (Default)
- Protected
- Public

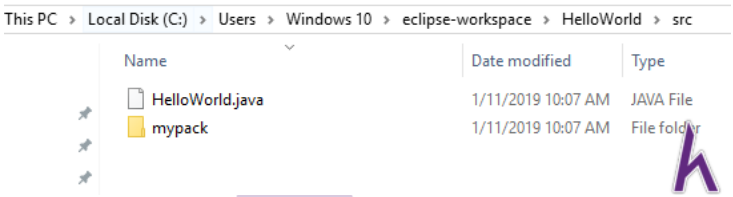
Bây giờ Kteam sẽ tạo một **package** tên là **mypack** và đưa class Person qua, việc này phục vụ cho những ví dụ phía dưới. Các bạn làm như sau trên Eclipse:



Rồi ta kéo **class Person** qua **package mypack**



Nếu các bạn sử dụng theo cách truyền thống. Thì bản chất package trong **Java** chính là folder chứa class mà thôi. Khi bạn xem trong source project, nó sẽ đổi như sau:



Và bạn sẽ để ý, trong các file java nó sẽ có sửa đổi như sau:

Person.java

```
1  package mypack;
2
3
4  public class Person {
5      public String name;
6      public int age;
7      public float height;
8
9      public Person(String name, int age, float height) {
10         this.name = name;
11         this.age = age;
12         this.height = height;
13     }
14
15     public void eat(String foodName) {
16         System.out.println(name + " is eating "+ foodName);
17     }
18
19     public int getAge() {
20         return age;
21     }
22 }
23
```

HelloWorld.java

```
1  import mypack.Person;
2
3  public class HelloWorld {
4      public static void main(String[] args) {
5          Person a = new Person("Chau", 21, 1.7f);
6          a.eat("Rice");
7      }
8  }
```

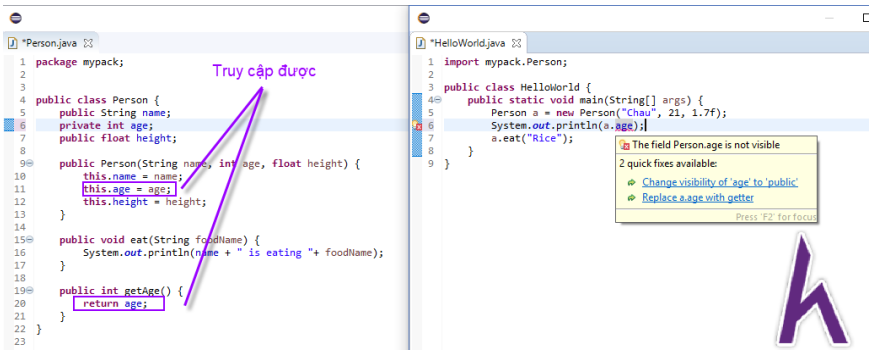
Việc thay đổi này do **IDE Eclipse** tự sửa cho chúng ta:

- Khi ta tạo một class nằm trong một package nào đó thì phải khai báo **package** đầu tiên: Ở đây class Person nằm trong **package mypack**. Ta sẽ viết **package mypack**;
- Khi ta sử dụng một class nằm ở packge khác, ta phải import qua để chương trình hiểu ta đang sử class của package nào. Theo đường dẫn class Person nằm trong **package mypack**, nên ở class HelloWorld ta sẽ viết **import mypack.Person;**

Private

Private chỉ cho phép truy cập nội bổ của một class.

Ví dụ: cho thuộc tính age của class **Person** ở dạng **private**, thì chỉ có thể truy cập age trong class Person.



(Default)

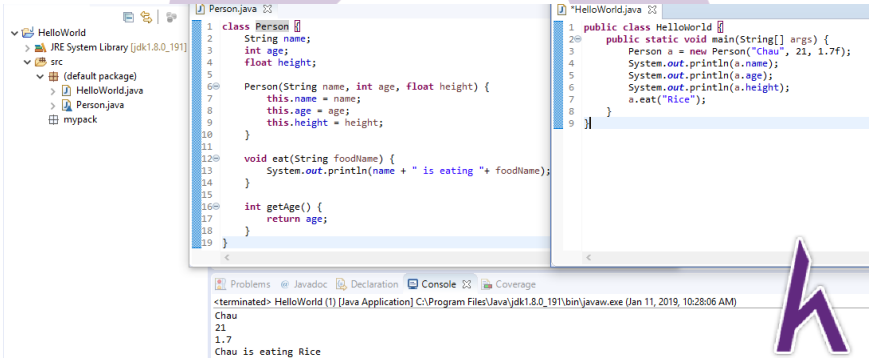
Đây là phạm vị mặc định, khi bạn không ghi gì hết thì nó để phạm vị truy cập dạng này: Ở mặc định, phạm vi truy cập chỉ nằm trong nội bộ package.

Ví dụ: ta xóa tất cả phạm vị truy cập ở class Person như sau

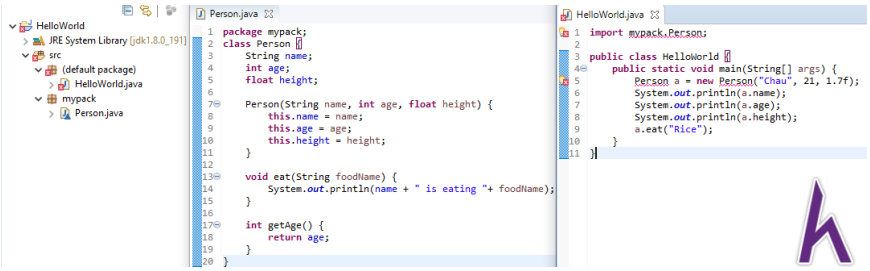
java:

```
class Person {
	String name;
	int age;
	float height;

	void eat(String foodName) {
		System.out.println(name + " is eating "+ foodName);
	}

	int getAge() {
		return age;
	}
}
```

Nếu ta quay lại cho class **Person** và **HelloWorld** chung một package thì sử dụng được:



Nhưng nếu để class **Person** ở package **mypack** thì sẽ báo lỗi:



Protected

Protected là phạm vi truy cập có thể từ trong và ngoài **package**, nhưng phải thông qua tính kế thừa. Tính kế thừa sẽ được Kteam giải thích rõ hơn trong bài TÍNH KẾ THỪA TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG. **Protected** chỉ có thể áp dụng bên trong class như thuộc tính, phương thức hay lớp con. Không thể áp dụng cho lớp ngoài hay **interface**.

Ví dụ: ta sửa class **Person** như sau:

java:

```
package mypack;
public class Person {
	protected String name;
	protected int age;
	protected float height;

	public Person(String name, int age, float height) {
		this.name = name;
		this.age = age;
		this.height = height;
	}

	protected void eat(String foodName) {
		System.out.println(name + " is eating "+ foodName);
	}

	protected int getAge() {
		return age;
	}
}
```

Để truy cập được **Person**, ta sẽ cho class **HelloWorld** kế thừa class **Person**. Rồi sẽ tạo đối tượng **HelloWorld** trong main:

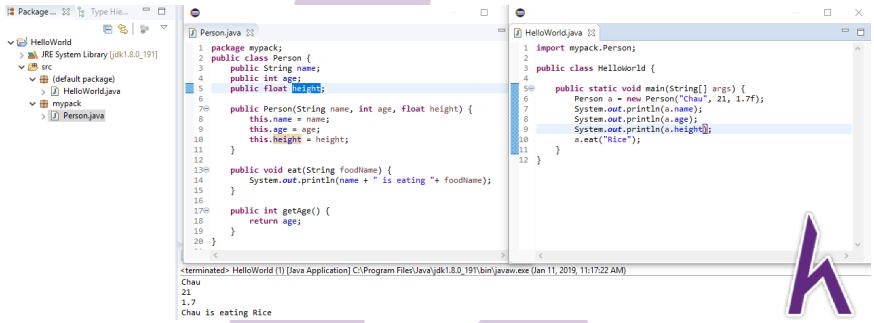
java:

```
import mypack.Person;

public class HelloWorld extends Person {

	protected HelloWorld(String name, int age, float height) {
		//phương thức khởi tạo HelloWorld sẽ gọi phương thức Person
		super(name, age, height);
	}

	public static void main(String[] args) {
		HelloWorld a = new HelloWorld("Chau", 21, 1.7f);
		a.eat("Rice");
	}
}
```

Chắc chắn các bạn sẽ thấy khó hiểu ở dòng code trên. Không sao, vì đó là một khái niệm khác nên các bạn có thể đọc qua và xem kĩ những bài sau.

Public

Đây phạm vi truy cập rộng, có thể truy cập bất cứ đâu trong **project Java**. Tất nhiên khi khác **package** để cần phải khai báo **import** để xác định ví trí của class như phần giải thích trên trên.



Kết

Như vậy chúng ta đã tìm hiểu các loại phạm vi truy cập trong lập trình hướng đối tượng

Ở bài sau, Kteam sẽ giới thiệu đến bạn về TỪ KHOÁ STATIC TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".