Ministry of SMEs and Startups × KOSME Korea SMEs and Startups Agency × LIKELION

Package, Function

Nguyễn Anh Tuấn

K TECH COLLEGE

Nội dung bài giảng

1 Phạm vi truy cập

2 Package

3 Lớp giao diện, lớp trừu tượng

4 Phương thức, chức năng

Phạm vi truy cập

Phạm vi truy cập

- **public**: Có thể được truy cập từ bất kỳ đâu, không giới hạn phạm vi truy cập.

```
public void publicMethod() {
    // public access
}
```

- **private**: Chỉ có thể được truy cập từ bên trong chính lớp đó. Không thể truy cập từ các lớp khác, kể cả lớp con.

```
private void privateMethod() {
    // private access
}
```

- **protected**: Có thể được truy cập từ bất kỳ lớp nào trong cùng một package và bất kỳ lớp con nào, kể cả khi lớp con ở package khác.

```
protected void protectedMethod() {
    // protected access
}
```

- **default** (không khai báo): Có thể được truy cập từ bất kỳ lớp nào trong cùng một package.

```
void defaultMethod() {
    // default access
}
```

So sánh các phạm vi truy cập

Các phạm vi truy cập cung cấp các mức độ bảo mật khác nhau cho các thành phần của lớp, giúp bảo vệ dữ liệu và đảm bảo mã nguồn được tổ chức hợp lý, dễ bảo trì.

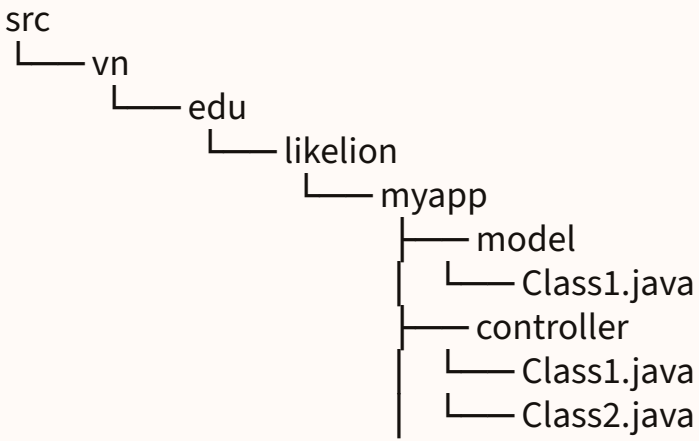
Phạm vi truy cập	Truy cập trong cùng lớp	Truy cập từ lớp khác trong cùng package	Truy cập từ lớp con ở package khác	Truy cập từ lớp không liên quan ở package khác
public	✅	✅	✅	✅
protected	✅	✅	✅	❌
default	✅	✅	❌	❌
private	✅	❌	❌	❌

Package

Định nghĩa, quy ước đặt tên package

Định nghĩa: Package (gói) là một nhóm các class (lớp), interface (lớp giao diện) và các package con. Nó được sử dụng để tổ chức và nhóm các lớp liên quan lại với nhau.
Quy ước: package domain.company.myapp;

Lưu ý: Package có quy ước đặt tên ngược lại với tên miền của công ty quản lý dự án.
Trường hợp tên miền công ty quản lý dự án là **likelion.edu.vn**, thì sử dụng **vn.edu.likelion** làm tiền tố.
Ví dụ: *package vn.edu.likelion.myapp;*



Sử dụng các lớp trong cùng 1 package

Class 1

```
// File: src/com/example/animals/Animal.java
package com.example.animals;

public class Animal {
        public void makeSound() { }
}
```

Class 2

```
// File: src/com/example/animals/Dog.java
package com.example.animals;

public class Dog extends Animal {
        @Override
        public void makeSound() { }
}
```

Sử dụng các lớp khác package

Class 1

```
// File: src/com/example/animals/Animal.java
package com.example.animals;

public class Animal {
        public void makeSound() { }
}
```

Class 2

```
// File: src/com/example/vehicles/Car.java
package com.example.vehicles;

import com.example.animals.Animal;

public class Car extends Animal {
        @Override
        public void makeSound() { }
}
```

Lớp giao diện, lớp trừu tượng

Lớp giao diện (interface class)

- Interface class là một tập hợp các phương thức trừu tượng (abstract methods) mà bất kỳ lớp nào triển khai (implement) nó phải cung cấp cài đặt (implementation) cho tất cả các phương thức đó.
- Các phương thức trong interface không có thân (body) và chỉ định tên, tham số, kiểu trả về.
- Interface không chứa các biến thể hiện (instance variables), nhưng nó có thể chứa các biến public static final (constants) để định nghĩa các giá trị hằng số cho các lớp triển khai.
- Interface không phải là một đối tượng và không thể được khởi tạo.
- Một lớp khi thực hiện một interface class sẽ sử dụng từ khóa “implement”.

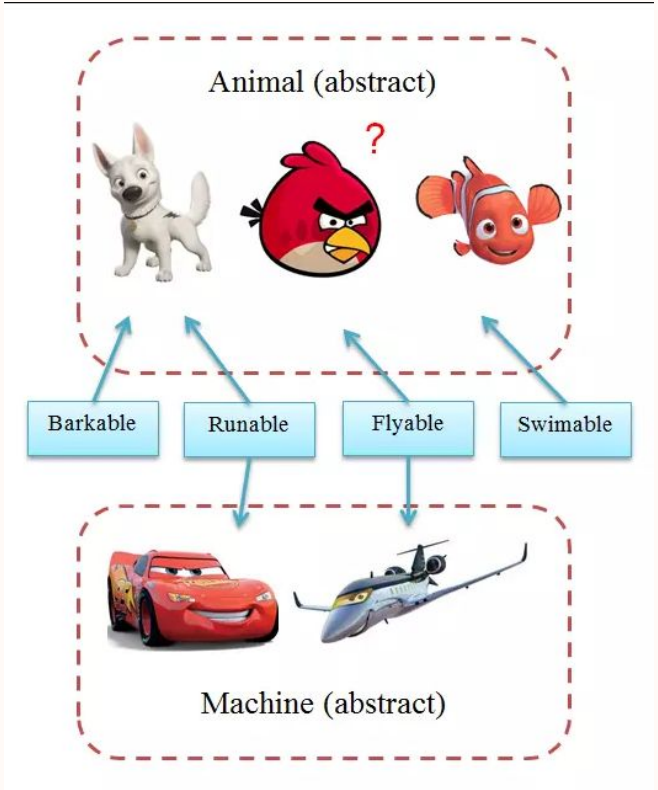
Lớp trừu tượng (abstract class)

- Abstract class trong Java là một lớp mà không thể tạo đối tượng từ nó. Nó có thể chứa cả phương thức trừu tượng (chưa được cài đặt) và phương thức cụ thể (đã được cài đặt).
- Một lớp kế thừa (subclass) phải cài đặt tất cả các phương thức trừu tượng trong abstract class hoặc bản thân nó cũng phải được khai báo là abstract.
- Một abstract class được khai báo bằng cách sử dụng từ khóa “abstract” trước từ khóa “class”.
- Một lớp khi kế thừa abstract class sẽ sử dụng từ khóa “extends”.

Minh họa về Interface và Abstract class

- Abstract class:
  - Animal
    - Chó
    - Chim
    - Cá
  - Machine
    - Xe hơi
    - Máy bay
- Interface class:
  - Sủa
  - Chạy
  - Bay
  - Bơi

=> Abstract class chỉ nên sử dụng chủ yếu cho các đối tượng có liên quan chặt chẽ, trong khi Interface phù hợp để cung cấp chức năng chung cho các lớp không liên quan.



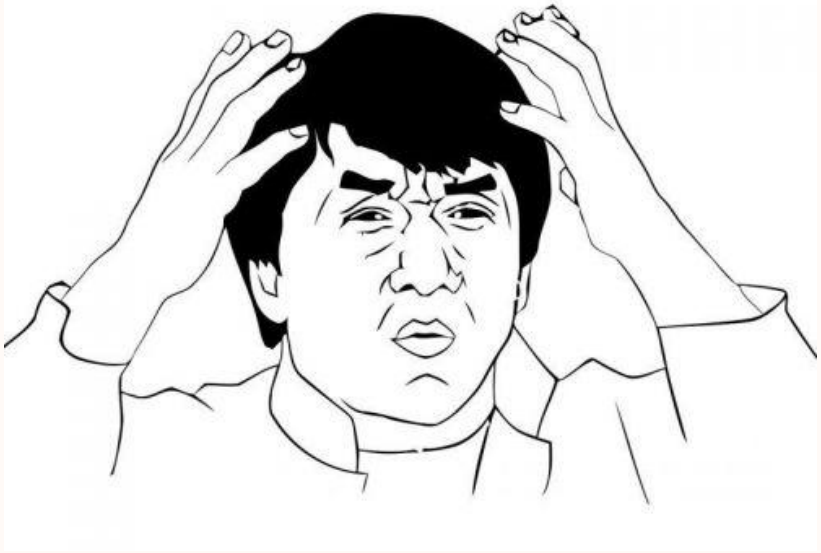
So sánh giữa Interface và Abstract class

Nội dung so sánh	Interface class	Abstract class
Vai trò	Cung cấp một khung chức năng cho các lớp khác thực hiện. Mối quan hệ **“can-do”.**	Xác định danh tính cơ bản của một lớp. Mối quan hệ **“is-a”.**
Kế thừa	Một lớp con có thể thực hiện nhiều lớp Interface.	Một class chỉ có thể kế thừa 1 lớp Abstract.
Phương thức	Interface bao gồm phương thức trừu tượng nên sẽ không khai báo phần thân cho phương thức, trừ phương thức **default** và phương thức **static** thì có thể khai báo.	Abstract bao gồm phương thức trừu tượng và phương thức cụ thể, do đó có thể khai báo phần thân cho phương thức.
Phạm vi truy cập	Tất cả các phương thức trong interface đều phải được định nghĩa là public.	Có thể có bất kỳ phạm vi truy cập nào.

Trả lời câu hỏi

- 1 interface class có thể kế thừa từ interface class khác không? Và có thể kế thừa bao nhiêu interface class khác?

- 1 abstract class có thể kế thừa từ abstract class khác không? Và có thể kế thừa bao nhiêu abstract class khác?



Phương thức, chức năng

Phương thức, chức năng

❖ **Định nghĩa:**

➢ Phương thức (method) dùng để thực hiện các nhiệm vụ, hành động của đối tượng (dùng trong lập trình hướng đối tượng).

➢ Chức năng (function) trong phạm vi Java (ngôn ngữ hướng đối tượng) có thể hiểu là phương thức (method).

➢ Ví dụ:

```
package com.example.animals;
public class Cat {
        public void makeSound() {
                // đây là phương thức tạo ra tiếng động, ví dụ tiếng kêu meo meo
                // được định nghĩa trong lớp Cat
        }
}
```

Phương thức có chứa tham số

❖ **Tham số kiểu dữ liệu nguyên thủy:**

```
public static void infomation(String name, int year) {
        System.out.println("Môn học " + name + " được Oracle mua lại vào năm " + year);
}

public static void main(String[] args) {
        information("java", 2010);
}
```

Phương thức có chứa tham số

❖ **Tham số kiểu dữ liệu đối tượng:**

```
public void present(Person person) {
        System.out.println(“Tôi tên là “ + person.getName());
        System.out.println(“Năm nay tôi “ + person.getAge() + “ tuổi”);
}

public static void main(String[] args) {
        Person person = new Person();
        present(person);
}
```

Phương thức có giá trị trả về

❖ **Giá trị trả về kiểu dữ liệu nguyên thủy:**

```
public class Person {
    private String name;
    private int age;

    public void setName(String name) { this.name = name; }
    public String getName() {
        return name;
    }

    public void setAge(int age) { this.age = age; }
    public int getAge() {
        return age;
    }
}
```

Phương thức có giá trị trả về

❖ **Giá trị trả về kiểu dữ liệu đối tượng:**

```
public static void present(Person person) {
        System.out.println(“Tôi tên là “ + person.getName());
        System.out.println(“Năm nay tôi “ + person.getAge() + “ tuổi”);
}

public static Person editPerson(Person person) {
        person.setName(“Hải Triều”);
        person.setAge(30);
        return person;
}

public static void main(String[] args) {
        Person person = new Person();
        present(editPerson(person));
}
```

ROAD TO KOREA

Nếu có bất kỳ thắc mắc nào, hãy đặt câu hỏi qua

mail@mail.com hoặc Zalo 0xxx xxx xxx