CHƯƠNG 8. PACKAGES VÀ INTERFACE

GV. TS. Hồ Thị Hương Thơm
SĐT: 0989567488

8.1. Packages

Khái niệm, cách tạo và sử dụng package

Khái niệm Package

- Khái niệm là một nhóm các lớp (class) và giao diện (interface) có liên quan, được tổ chức chung trong một khối logic, giống như một thư mục chứa mã nguồn.

- Có thể là package có sẵn (built-in) hoặc tự định nghĩa (user-defined).

Lợi ích của việc sử dụng Package

- **Tổ chức rõ ràng, dễ quản lý**: Gom nhóm các lớp liên quan, hỗ trợ tìm kiếm, bảo trì và phát triển dễ dàng.

- **Tránh xung đột tên**: Các package khác nhau có thể **chứa lớp trùng tên** mà không gây lỗi.

- **Kiểm soát truy cập** (Encapsulation): Kết hợp từ khóa public, protected,... để che giấu thông tin không cần thiết, tăng bảo mật.

Gói có sẵn (Built-in packages)

- Java cung cấp nhiều gói thư viện có sẵn trong Java API, tổ chức theo nhóm chức năng.

+java.lang: tự động nhập, chứa các lớp lõi (String, System, Math,...)

+java.util: lớp tiện ích (ArrayList, Scanner,...)

+java.io: xử lý nhập/xuất dữ liệu

Gói có sẵn (Built-in packages)

- Ví dụ: Sử dụng class Scanner trong package java.util

```java
// File: Main.java
import java.util.Scanner;   // import class Scanner từ package java.util


public class Main {
    public static void main(String[] args) {
        // tạo đối tượng Scanner để đọc từ bàn phím
        Scanner scanner = new Scanner(System.in);
        System.out.print("Nhập tên của bạn: ");
        String name = scanner.nextLine();
        System.out.println("Xin chào, " + name + "!");
        scanner.close();  // đóng scanner sau khi sử dụng
    }
}
```

Cách tạo và sử dụng Package

- **Bước 1: Khai báo package**

Cú pháp: `package <tên_gói>`

Tên gói thường viết thường, có thể nhiều mức ví dụ: vimaru.edu.vn

```
// File: mypackage/MyClass.java
package mypackage;

public class MyClass {
    public void hello() {
        System.out.println("Xin chào từ MyClass trong gói mypackage");
    }
}
```

Cách tạo và sử dụng Package

- Bước 2: Lưu trữ tệp đúng thư mục

  Nếu khai báo là package mypackage;

  → đặt file vào thư mục: mypackage/

```
<project-root>/
├── …
└── mypackage/
    └── MyClass.java
```

Nếu khai báo là package com.myapp.util;

→ thư mục là: com/myapp/util/

Cách tạo và sử dụng Package

- **Bước 3: Sử dụng lớp từ package**

```
<project-root>/
├── …
├── TestPackage.java
└── mypackage/
    └── MyClass.java
```

```java
// Cách 1: Import class
import mypackage.MyClass;

public class TestPackage {
    public static void main(String[] args) {
        MyClass obj = new MyClass();
        obj.hello(); //Gọi hàm hello() có trong MyClass.java
    }
}


// Cách 2: Gọi trực tiếp tên đầy đủ (ít dùng hơn)
public class TestPackage {
    public static void main(String[] args) {
        mypackage.MyClass obj = new mypackage.MyClass();
        obj.hello();
    }
}
```

Phạm vi truy cập mặc định (default)

- Nếu một thuộc tính/phương thức không khai báo phạm vi truy cập là private, public hay protected
- → được coi là truy cập mặc định (default, package-private) nghĩa là chỉ truy cập được trong cùng package

Phạm vi truy cập mặc định (default)

- Ví dụ: truy cập các phần tử có phạm vi truy cập mặc định, trong cùng một package

```
//File Dog.java
package animals;

class Dog {
    String name = "Chó";   // phạm vi: default

    void bark() {          // phạm vi: default
        System.out.println("Gâu gâu");
    }
}


//File Main.java
package animals;
public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog();
        System.out.println(dog.name);   // Truy cập biến default
        dog.bark();                     // Gọi method default
    }
}
```

Tất cả phạm vi truy cập trong Java

Phạm vi truy cập	Trong cùng lớp	Trong cùng package	Lớp con khác package	Bất kỳ nơi nào (ngoài package)
public	✅	✅	✅	✅
protected	✅	✅	✅	❌
(default)	✅	✅	❌	❌
private	✅	❌	❌	❌

✅ : có thể truy cập ; ❌ : không thể truy cập

8.2. Interface

Khái niệm, đặc điểm và ví dụ minh họa Interface

Khái niệm Interface (giao diện)

- Là một tập hợp các phương thức trừu tượng (abstract methods) và hằng số, đóng vai trò như một bản thiết kế cho các lớp

- Cú pháp khai báo:

```
class TênLớp extends LớpCha (nếu có) implements
TênInterface, ... {
    //nội dung lớp, bao gồm triển khai các phương thức của interface
}
```

Đặc điểm của Interface

1. Tất cả các phương thức trong Interface đều mặc định là public abstract, và không có thân hàm

```
interface Animal {
    void sound();  // tự hiểu là public abstract
}
```

2. Biến trong interface là public static final (hằng số)

```
interface Config {
    int MAX_SIZE = 100;   // tương đương public static final
}
```

Đặc điểm của Interface

3. Không tạo được đối tượng từ interface:

Không có constructor → không thể tạo new Interface().

4. Lớp implements phải override toàn bộ phương thức (trừ khi lớp đó là abstract)

```
interface Animal {
    void sound();
}

class Dog implements Animal {
    @Override
    public void sound() {
        System.out.println("Woof");
    }
}
```

Đặc điểm của Interface

5. Một lớp có thể implements nhiều interface

```
class SmartPhone implements Camera, GPS {
    // phải ghi đè và triển khai các phương
thức có trong interface Camera và GPS
}
```

6. Interface có thể kế thừa interface khác

```
interface A { void methodA(); }
interface B { void methodB(); }
interface C extends A, B {}
```

//C kế thừa methodA + methodB

Ví dụ minh họa về Interface (1)

- Tạo interface Animal với phương thức trừu tượng makeSound()

  Lớp Cat và Dog triển khai Animal

```
public interface Animal {
    void makeSound();
}


public class Dog implements Animal {
    @Override
    public void makeSound() {
        System.out.println("Woof!"); }
}


public class Cat implements Animal {
    @Override
    public void makeSound() {
        System.out.println("Meow!");
    }
}
```

Ví dụ minh họa về Interface

- Lớp Main tạo đối tượng và gọi hàm

```
public class Main {
    public static void main(String[] args) {
        Animal a1 = new Dog();
        Animal a2 = new Cat();

        a1.makeSound();  // Gọi phương thức makeSound() -> in: "Woof!"
        a2.makeSound();  // Gọi phương thức makeSound() -> in: "Meow!"
    }
}
```

Làm các bài tập và bài trắc nghiệm

- Theo tài liệu bài tập thực hành
- Học trực tuyến: hoctructuyen.vimaru.edu.vn