Bài 4:
Hiểu thêm về Java

Trường ĐH Công nghệ, ĐHQG Hà Nội

Tài liệu tham khảo

- *Giáo trình Lập trình HĐT*, Chương 3, 4
- *Java How to Program*, Chapters 3-8

Nội dung

- Phương thức và thuộc tính tĩnh
- Gói và kiểm soát truy cập
- Kiểu hợp thành (composition)
- Vào ra với luồng dữ liệu chuẩn

Phương thức và thuộc tính tĩnh

- Phương thức và thuộc tính có thể được khai báo là thành phần tĩnh (static)
  - Độc lập với đối tượng
  - Có thể được truy cập không qua đối tượng
    - Sử dụng tên lớp
- Thuộc tính tĩnh
  - Thuộc về lớp
  - Được chia sẽ giữa các đối tượng của lớp

```
public class Dummy {
   // number of Dummy objects
   static int counter = 0;
   static int count() {
      return counter;
   }

   private String name;

   public Dummy(String name) {
      counter++;
      this.name = name;
   }

   //main function to test Dummy class
   public static void main(String args[]) {
      System.out.println(Dummy.count());
      Dummy d1 = new Dummy ("First Dummy");
      System.out.println(d1.count());
      Dummy d2 = new Dummy ("Second Dummy");
      System.out.println(d1.count());
   }
}
```

Dummy.counter được chia sẻ và được cập nhật bởi các đối tượng Dummy.

```
0
1
2
```

Dummy.count() có thể được gọi qua lớp, không cần đối tượng.

Phương thức và thuộc tính tĩnh

- Phương thức tĩnh
  - Không thể truy cập các thành phần thông thường khác (non-static)
  - Không thể gọi các phương thức non-static
- Tại sao?

```
class Hello{
    public void sayHello(String msg){
        System.out.println(msg);
    }
}
```

```
public class HelloTestDrive{
    public static void main(String[] args){
        Hello h=new Hello();//tạo đối tượng
        h.sayHello("Hello, world");
    }
}
```

```
class Hello{
    public static void sayHi(String msg) {
        System.out.println(msg);
    }
    public void sayHello(String msg){
        System.out.println(msg);
    }
}
```

```
public class HelloTestDrive{
    public static void main(String[] args){
        Hello.sayHi("Hi, there!");

        Hello h=new Hello();
        h.sayHello("Hello");
        h.sayHi("Hello");
    }
}
```

```
class Hello{
    public static String message="Hello, world";
    public static void sayHi(msg) {
        System.out.println(msg);
    }
}
```

```
public class HelloTestDrive{
    public static void main(String[] args){
        Hello.sayHi("Hi!");

        System.out.println(Hello.message);

    }
}
```

Gói các lớp đối tượng (package)

- Các lớp đối tượng được chia thành các gói
  - nếu không khai báo thì các lớp thuộc gói default
  - các lớp trong cùng một tệp mã nguồn luôn thuộc cùng một gói
- Tồn tại mức truy cập gói (package)
  - mức package là mặc định (nếu không khai báo tường minh là public hay private)
  - các đối tượng của các lớp thuộc cùng gói có thể truy cập đến thành phần non-private của nhau
  - chỉ có thể tạo (new) đối tượng của lớp được khai báo là public của gói khác

Sử dụng Gói như thế nào?

```
//Hello.java:

class HelloMsg {
    void sayHello() {
        System.out.println(”Hello, world!”);
    }
}

public class Hello {
    public static void main(String[] args) {
        HelloMsg msg = new HelloMsg();
        msg.sayHello();
    }
}
```

Làm thế nào để đưa HelloMsg vào một gói?

Khai báo Gói

- Lệnh package xuất hiện ngay ở dòng đầu tiên trong tệp

khai báo gói bằng lệnh package với tên gói. Phần còn lại của tệp sẽ thuộc về cùng một gói.

```
// HelloMsg.java
package dse;

public class HelloMsg {
   public void sayHello() {
      System.out.println("Hello, world!");
   }
}
```

Được khai báo public nên có thể được sử dụng ngoài phạm vi gói dse.

Hai cách sử dụng Gói

1. Sử dụng lệnh import để cung cấp các tên (names) trong gói và chỉ cần một lần.

```
//Hello.java
import dse.HelloMsg;

public class Hello {
    public static void main(String[] args) {
        HelloMsg msg = new HelloMsg();
        msg.sayHello();
    }
}
```

2. Chỉ rõ gói cho mỗi lần gọi.

```
//Hello.java
public class Hello {
    public static void main(String[] args) {
        dse.HelloMsg msg = new dse.HelloMsg();
        msg.sayHello();
    }
}
```

dse/hello/Hello.java:

```
package dse.hello;
public class Hello {
    public void sayHello() {
        System.out.println("Hello, world!");
    }
}
```

dse/test/HelloTestDrive.java:

```
import dse.hello.Hello;

public class Hello {
    public static void main(String[] args) {
        Hello h = new Hello();
        h.sayHello();
    }
}
```

Biên dịch và thực thi

- Biên dịch
  - javac HelloMsg.java –d
  - javac Hello.java
- javac dse/test/HelloTestDrive.java
- Thực thi
  - java Hello
  - java dse.test.HelloTestDrive

Hợp thành (composition)

- Các đối tượng có thể chứa các đối tượng khác ở dạng thuộc tính
- Các thuộc tính kiểu tham chiếu này phải được khởi tạo sử dụng toán tử new hoặc phép gán

Hợp thành

Dữ liệu có kiểu không nguyên thủy

```
public class Person {
   private String name;
   private Date birthDate;

   public Person(String name, Date birthDate) {
      this.name = name;
      this.birthDate = birthDate;
   }

   public String toString() {
      return String.format("%s: Birthday: %s",name,birthDate);
   }
}
```

Đối tượng được chứa phải được tạo.

```
...
Date d = new Date (31, 12, 1999);
Person p1 = new Person ("Bob", d);

Person p2 = new Person ("Alice", new Date (1, 1, 2000));
```

Get/Set thuộc tính tham chiếu

```
class Person {
…
    public MyDate getBirthday() {
        return birthday;
    }
}

Person p = new Person(...);
MyDate d = p.getBirthday();
d.setYear(1900);
```

Get/Set bằng khởi tạo sao chép

```
class Person {
    private String name;
    private MyDate birthday;
    public Person(String s, MyDate d) {
        name = s;
        birthday = new MyDate(d);
    }
    public MyDate getBirthday() {
        return new MyDate(birthday);
    }
    public void setBirthday(MyDate d) {
        birthday = new MyDate(d);
    }
...
}
```

Cách nào đúng???

Vào ra từ luồng dữ liệu chuẩn

- Ba đối tượng luồng được tạo tự động khi thực thi chương trình
  - **System.out**: đối tượng luồng ra chuẩn
    - thường cho phép chương trình xuất dữ liệu ra màn hình
  - **System.err**: đối tượng luồng lỗi chuẩn
    - thường cho phép chương trình xuất thông báo lỗi ra màn hình
  - **System.in**: đối tượng luồng vào chuẩn
    - thường cho phép chương trình nhận dữ liệu (bytes) từ bàn phím
- Có thể được điều hướng để được gửi hoặc đọc từ các nguồn khác, chẳng hạn từ tệp trên đĩa
  - Sử dụng phương thức setIn(), setOut(), setErr()
  - Tại cửa sổ lệnh (chỉ input và output):
    C:\> input.dat > java AJavaProgram > output.dat

Nhập dữ liệu từ luồng vào chuẩn

- InputStream: lớp đối tượng ứng với luồng vào chuẩn
  - System.in: đối tượng tương ứng
  - chưa có phương thức nhập dữ liệu
- Scanner: nhập dữ liệu kiểu nguyên thủy và xâu ký tự
  - next(): nhập xâu ký tự
  - nextType(): nhập dữ liệu kiểu Type
  - hasNext(), hasNextType(): kiểm tra xem còn dữ liệu không

Ví dụ: Nhập liệu từ luồng vào chuẩn


```
// import the wrapper class
import java.util.Scanner;
...
// create Scanner to get input from keyboard
Scanner sc = new Scanner( System.in );

// read a word
System.out.println(sc.next());

// read an integer
int i = sc.nextInt();

// read a series of big intergers
while (sc.hasNextLong()) {
   long aLong = sc.nextLong();
}
```

Tham số dòng lệnh

CmdLineParas.java:

```
public class CmdLineParas {
public static void main(String[] args) {
    for (int i=0; i<args.length; i++)
        System.out.println(args[i]);
    }
}
```

Ví dụ:

```
#java CmdLineParas hello world
hello
world
```

Tổng kết

- Phương thức và thuộc tính tĩnh
- Gói và kiểm soát truy cập
- Kiểu hợp thành (composition)
- Vào ra với luồng dữ liệu chuẩn