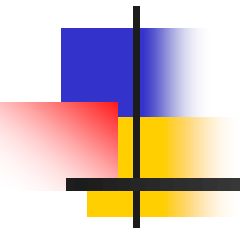
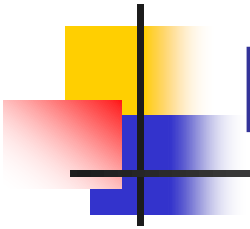


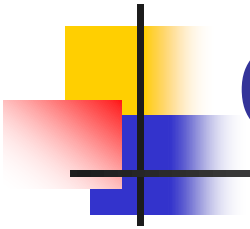
Bài 4

Các Gói & Giao diện (Packages & Interfaces)



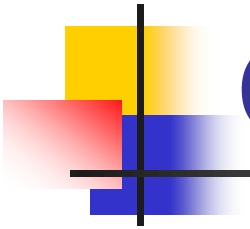
Mục tiêu của bài

- Định nghĩa giao diện (interface)
- Thực thi giao diện (interface)
- Dùng giao diện như một kiểu dữ liệu
- Định nghĩa gói (Package)
- Tạo và sử dụng gói
- Vai trò của gói trong việc điều khiển truy nhập



Giới thiệu

- Những thành phần cơ bản của 1 chương trình Java:
  - Gói (Packages)
  - Giao diện (Interfaces)
- Những phần nội tại của một tập tin nguồn Java:
  - Mệnh đề **package** (khai báo gói)
  - Mệnh đề i**mport** (nhập thêm vào)
  - Khai báo lớp công cộng đơn (single public class)
  - Các lớp riêng của gói (classes private to the package)
- Tập tin nguồn Java có thể chứa tất cả hoặc một vài trong số các phần nội tại nói trên.



Giao diện (Interfaces)

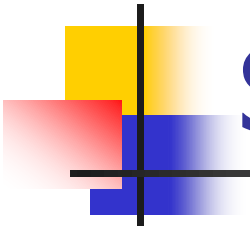
- Cho phép một lớp có thể có nhiều lớp cha (superclasses)
- Chương trình Java chỉ có thể kế thừa từ 1 lớp duy nhất trong cùng một thời điểm, nhưng có thể áp dụng cùng lúc nhiều giao diện (Interfaces)
- Không được phép có những phương thức cụ thể (concrete methods)
- Giao diện cần phải được hiện thực.

Các bước tạo ra giao diện

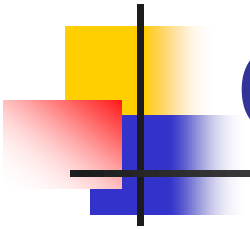
- Định nghĩa giao diện (Interface)
- Biên dịch giao diện (Interface)
- Áp dụng giao diện (Interface)

- Trong khi tạo ra giao diện (interface):
  - Tất cả phương thức trong giao diện phải là kiểu công cộng (**public** type**)**
  - Phương thức phải được định nghĩa trong lớp hiện thực giao diện đó

Sử dụng giao diện (Interface)

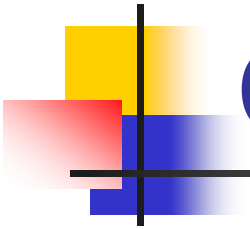


- Không thể mở rộng lớp, nhưng có thể mở rộng những giao diện khác
- Nếu áp dụng việc mở rộng một giao diện sang một giao diện khác, những phương thức trong giao diện mới cũng như trong giao diện cũ phải được ghi đè (overridden)
- Tất cả phương thức trong giao diện phải ở kiểu công cộng (public)
- Khi định nghĩa một giao diện mới thì một kiểu dữ liệu tham chiếu cũng được định nghĩa



Gói (Packages)

- Là các thư mục lưu trữ những lớp và giao diện
- Mỗi một gói bao gồm nhiều lớp và/hoặc giao diện. Đó là những thành viên của gói

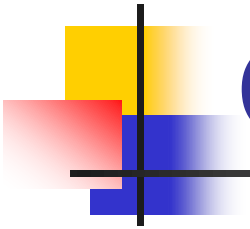


Gói (Packages) (Tiếp theo…)

- Những điểm mạnh của gói (Package):
  - Cho phép tổ chức các lớp đối tượng thành những đơn vị nhỏ hơn
  - Giúp tránh được tình trạng trùng lặp định danh
  - Cho phép bảo vệ các lớp đối tượng
  - Tên gói (Package) có thể được dùng để nhận dạng các lớp đối tượng

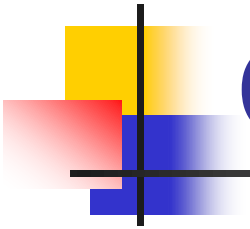
Gói (Packages) (Tiếp theo…)

- Những lưu ý khi tạo gói:
  - Mã nguồn phải bắt đầu bằng mệnh đề 'package'
  - Mã nguồn phải nằm trong cùng thư mục mang tên của gói
  - Tên gói nên bắt đầu bằng ký tự thường (lower case) để phân biệt giữa lớp đối tượng và gói
  - Những mệnh đề khác có thể được viết phía dưới dòng khai báo gói là mệnh đề **import**, kế đến là các mệnh đề định nghĩa lớp đối tượng
  - Những lớp đối tượng trong gói cần phải được biên dịch
  - Để chương trình Java có thể sử dụng những gói này, ta phải **import** gói vào trong mã nguồn



Gói (Packages) (Tiếp theo…)

- Import gói (Importing packages):
  - Xác định tập tin cần được import trong gói
  - Hoặc có thể import toàn bộ gói

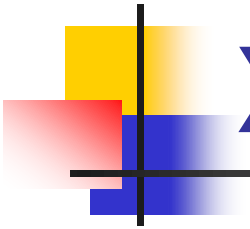


Các bước tạo ra gói (Package)

- Khai báo gói
- Import những gói chuẩn cần thiết
- Khai báo và định nghĩa các lớp đối tượng có trong gói
- Lưu các định nghĩa trên thành tập tin **.java**, và biên dịch những lớp đối tượng đã được định nghĩa trong gói.

Sử dụng những gói do người dùng định nghĩa (user-defined packages)

- Mã nguồn của những chương trình này phải ở cùng thư mục của gói do người dùng định nghĩa.
- Để những chương trình Java khác sử dụng những gói này, import gói vào trong mã nguồn
- Import những lớp đối tượng cần dùng
- Import toàn bộ gói
- Tạo tham chiếu đến những thành viên của gói



Xác lập CLASSPATH

- Là danh sách các thư mục, giúp cho việc tìm kiếm các tập tin lớp đối tượng tương ứng
- Nên xác lập CLASSPATH trong lúc thực thi (runtime), vì như vậy nó sẽ xác lập đường dẫn cho quá trình thực thi hiện hành

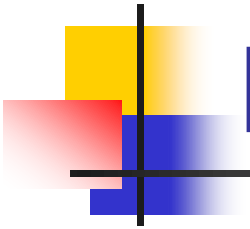
Gói và điều khiển truy xuất (Packages & Access Control)

	public	protected	No modifier	private
Same class	Yes	Yes	Yes	Yes
Same package subclass	Yes	Yes	Yes	No
Same package non-subclass	Yes	Yes	Yes	No
Different package subclass	Yes	Yes	No	No
Different package non-subclass	Yes	No	No	No

Gói java.lang

- Mặc định thì bất cứ chương trình Java nào cũng import gói **java.lang**
- Những lớp Wrapper (bao bọc) cho các kiểu dữ liệu nguyên thủy:

Data type	Wrapper class
boolean	Boolean
byte	Byte
char	Character
double	Double
float	Float
int	Integer
long	Long
short	Short

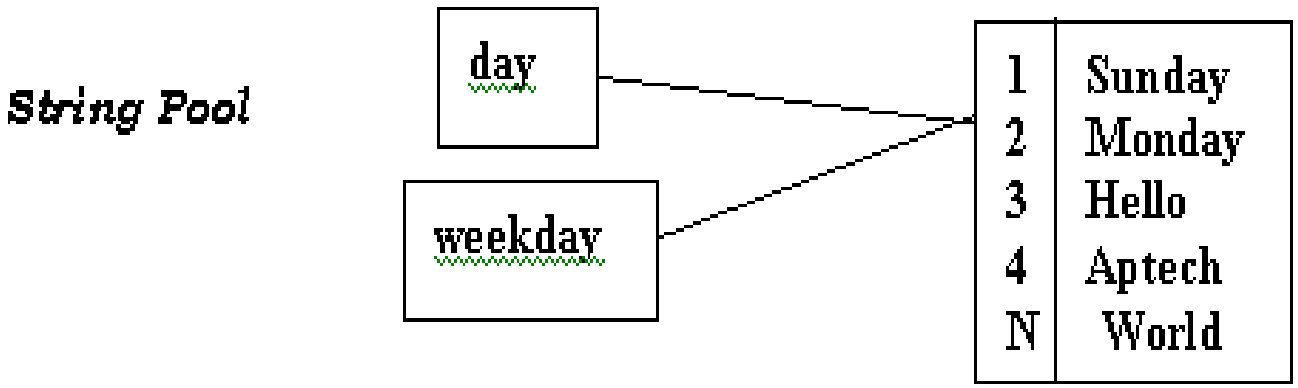


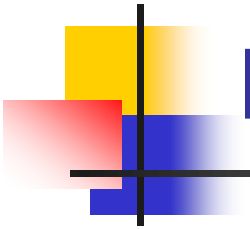
Lớp **String**

- Phương thức khởi tạo (Constructor):
  - **String str1 = new String();**
  - **String str2 = new String(“Hello World”);**
  - **char ch[ ] = {“A”,”B”,”C”,”D”,”E”};**
  - **String str3 = new String(ch);**
  - **String str4 = new String(ch,0,2);**

String Pool

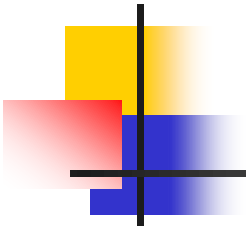
- 'String Pool' đại diện cho tất cả các ký tự được tạo ra trong chương trình
- Khái niệm 'String Pool'





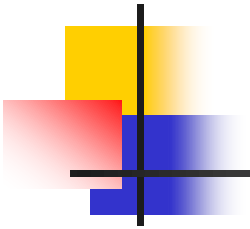
Những phương thức của lớp **String**

- **charAt( )**
- **startsWith()**
- **endsWith( )**
- **copyValueOf( )**
- **toCharArray( )**
- **indexOf( )**
- **toUpperCase( )**
- **toLowerCase( )**
- **trim( )**
- **equals( )**



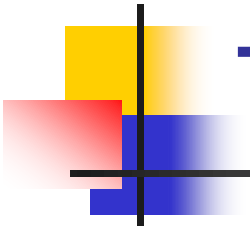
Lớp **StringBuffer**

- Cung cấp những phương thức khác nhau để thao tác trên đối tượng string (chuỗi ký tự)
- Những đối tượng của lớp này khá linh hoạt
- Cung cấp những phương thức khởi tạo (constructor) đã được nạp chồng (overloaded)
- Những phương thức của lớp **StringBuffer**:
  - **append( )**
  - **insert( )**
  - **charAt( )**
  - **setCharAt( )**
  - **setLength( )**
  - **getChars( )**
  - **reverse( )**



Lớp **java.lang.Math**

- **abs()**
- **ceil()**
- **floor()**
- **max()**
- **min()**
- **round()**
- **random()**
- **sqrt()**
- **sin()**
- **cos()**
- **tan()**



Tóm tắt

- Gói là thư mục mà lớp đối tượng và giao diện được tổ chức lưu trữ trong đó
- **CLASSPATH** là danh sách các thư mục giúp cho việc tìm kiếm các tập tin lớp đối tượng tương ứng
- Những lớp đối tượng **Wrapper** (bao bọc)
- '**String Pool**' đại diện cho tất cả các ký tự được tạo trong chương trình
- Lớp đối tượng **String** và **StringBuffer**
- Lớp **java.lang.Math**