



LẬP TRÌNH VIÊN CÔNG NGHỆ JAVA

Module 1

☞ *Click vào phụ lục để chuyển tới bài cần đọc*

Phụ lục

Bài 1 Tổng quan – Môi trường làm việc	2
Bài 2 Kiểu dữ liệu cơ sở	15
Bài 3 Giao diện 1.....	29
Bài 4 Xử lý tập tin	40
Bài 5 Mảng cơ sở	59
Bài 6 Giao diện 2.....	68
Bài 7: Đối tượng	78
Bài 8 Sử dụng dịch vụ	99



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

LTV CÔNG NGHỆ JAVA

Module 1 – Bài 1: *Tổng quan – Môi trường làm việc*

Ngành LT & CSDL

www.t3h.vn



2014

2014



Nội dung



1. Giới thiệu ứng dụng Java
2. Môi trường phát triển ứng dụng
3. Xây dựng ứng dụng đầu tiên



Giới thiệu ứng dụng Java



❑ Chương trình (Program)

- Chương trình là tập hợp các chỉ thị lệnh yêu cầu máy tính thực thi một tác vụ cụ thể



Giới thiệu ứng dụng Java



❑ Ngôn ngữ lập trình (Programming Language)

- Là một ngôn ngữ nhân tạo
- Gồm một tập các ký hiệu và cú pháp được chuẩn hóa để mô tả những xử lý mà người và máy đều có thể hiểu được



Giới thiệu ứng dụng Java



❑ Lập trình (Programming)

- Là quá trình xây dựng các chương trình nguồn được viết bằng một hoặc nhiều ngôn ngữ lập trình
- Các phương pháp lập trình thường gặp
 - Lập trình tuyến tính (Linear Programming)
 - Lập trình cấu trúc (Structured Programming)
 - Lập trình hướng đối tượng (Object Oriented Programming)
 - Lập trình trực quan (Visual Programming)

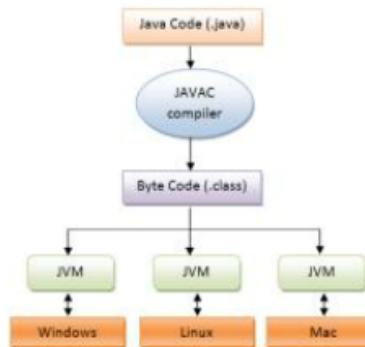


Giới thiệu ứng dụng Java



☐ Ngôn ngữ trung gian của Java - Java bytecode:

- Là định dạng biên dịch chương trình viết bằng Java
- Là các chỉ thị lệnh JVM thực thi
- Các tập tin bytecode có phần mở rộng .class



Giới thiệu ứng dụng Java



☐ Lịch sử phát triển của Java:

- Java được viết bởi James Gosling vào 6/1991 trong dự án *Top Box Project*
- Đầu tiên gọi là *Oak*
- Được đổi tên thành Java 1.0 năm 1995

☐ "Write Once, Run Anywhere" (WORA)

☐ Phiên bản mới nhất Java SE 7, Java EE 7



Giới thiệu ứng dụng Java



□ Phân loại

- J2SE (Java 2 Standard Edition)
 - Ứng dụng Desktop
- J2EE (Java 2 Enterprise Edition)
 - Ứng dụng Enterprise
- J2ME (Java 2 Mobile Edition)
 - Ứng dụng Mobile
- ...



Nội dung



1. Giới thiệu ứng dụng Java
2. Môi trường phát triển ứng dụng
3. Xây dựng ứng dụng đầu tiên



Môi trường phát triển ứng dụng



□ Giới thiệu Eclipse

- Là một môi trường phát triển tích hợp (IDE- integrated development environment)
- Được phát triển ban đầu bởi IBM, và hiện nay là bởi tổ chức Eclipse
- Chứa một workspace cơ bản và một hệ thống plug-in mở rộng để tùy biến các môi trường
- Viết phần lớn trong Java
- Được sử dụng để phát triển các ứng dụng Java và các ứng dụng khác như PHP, C/C++, C#, XML...

□ Phiên bản mới nhất Eclipse Kepler 26/06/2013



LTV Công nghệ Java – Module 1

10

Các thành phần của Eclipse

Môi trường phát triển ứng dụng



❑ Cài đặt Eclipse

- Download phiên bản mới nhất của Eclipse tại: <http://www.eclipse.org/downloads/> gói đầy đủ
- Tạo thư mục chứa các project của Java (mặc định)
 - Ví dụ: Java_Project
- Giải nén gói download và chạy file eclipse.exe

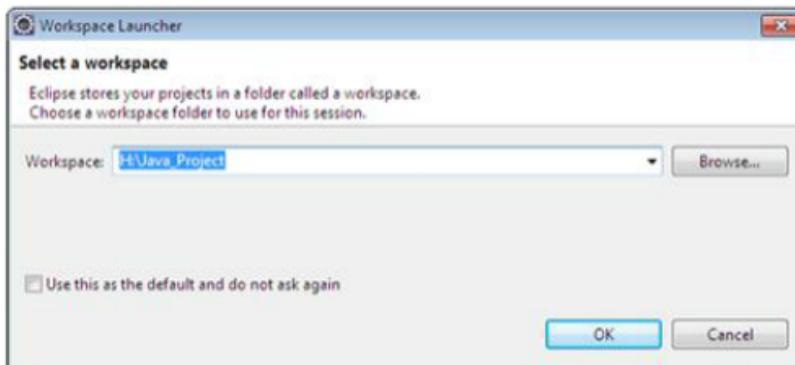


Môi trường phát triển ứng dụng



❑ Cài đặt Eclipse

- Chọn thư mục mặc định > OK



Môi trường phát triển ứng dụng



❑ Cài đặt Plugin

- Là một thành phần được gắn vào Eclipse để hỗ trợ một công việc nào đó
- Ví dụ: Windows Builder Pro do Google cung cấp dùng để thiết kế giao diện .



Môi trường phát triển ứng dụng



❑ Cài đặt Plugin

- Cài đặt
 - Help > Install New Software > Add
 - Nhập đường dẫn chứa plugin vào Location (ví dụ: <https://dl.google.com/eclipse/plugin/4.3>) > Chọn các plugin cần cài > OK



Môi trường phát triển ứng dụng



❑ Cài đặt Plugin

● Thực hành cài đặt Windows Builder Pro

- Help > Install New Software > Add
- Nhập đường dẫn chứa plugin vào Location (ví dụ: <http://dl.google.com/eclipse/inst/d2wbpro/latest/3.Z>) > OK



Nội dung



1. Giới thiệu ứng dụng Java
2. Môi trường phát triển ứng dụng
3. Xây dựng ứng dụng đầu tiên



Xây dựng ứng dụng đầu tiên



□ Cấu trúc chương trình viết bằng Java

```
package test;
public class chao_console {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Chào mừng các bạn đến với \n Ngôn ngữ lập trình Java!");
    }
}
```

Tên lớp

Tên phương thức chính, điểm bắt đầu của chương trình viết bằng Java

In kết quả ra màn hình Console



Xây dựng ứng dụng đầu tiên

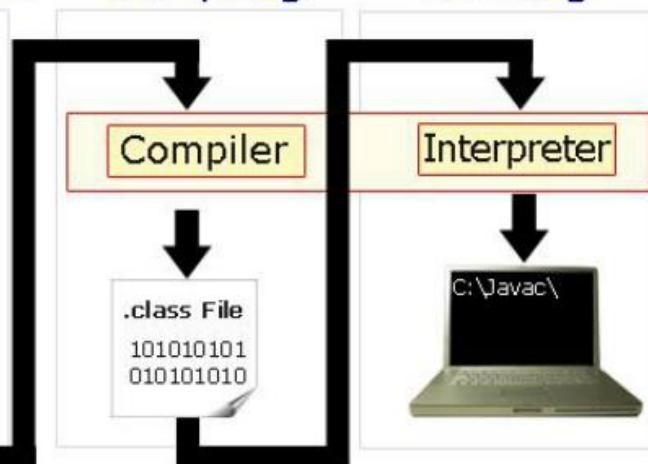


□ Xây dựng và thực thi chương trình

Write a Program



Compiling



Running



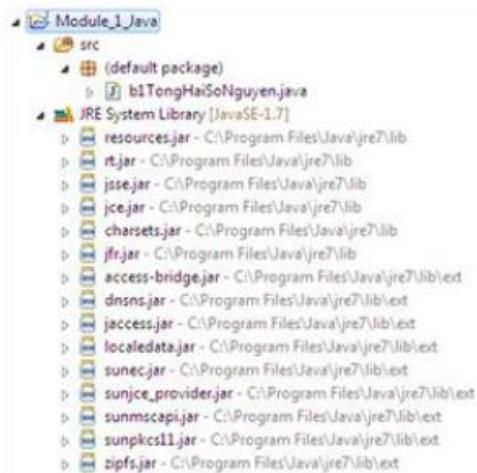
Xây dựng ứng dụng đầu tiên



□ Tạo project:

- File > New > Java Project > nhập tên của Project > chọn vị trí lưu trữ hoặc để vị trí mặc định > Finish

- Cấu trúc Project:



Xây dựng ứng dụng đầu tiên



□ Tạo một ứng dụng trên Console:

- Chọn Project chứa ứng dụng > src > New > Class > nhập tên của class > Finish
- Ví dụ: Tạo ứng dụng xuất câu chào như sau: **Chào mừng các bạn đến với
Ngôn ngữ lập trình Java!**

```
package test;

public class chao_console {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Chào mừng các bạn đến với \n Ngôn ngữ lập trình Java!");
    }
}
```



Xây dựng ứng dụng đầu tiên



□ Tạo một ứng dụng Windows Form:

- Chọn Project chứa ứng dụng > src > New > Others... > chọn JFrame > nhập tên của ứng dụng > Finish
- Ví dụ: Tạo ứng dụng xuất câu chào như sau:



Xây dựng ứng dụng đầu tiên

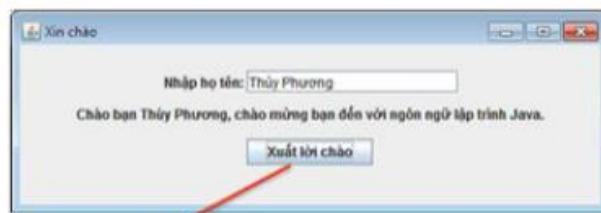


□ Tạo một ứng dụng Windows Form:

- Ví dụ:

- Khai báo biến:

```
private static final String CAU_CHAO = "Chào  
bạn %s, chào mừng bạn đến với ngôn ngữ lập  
trình Java.;"
```



```
public void actionPerformed(ActionEvent arg0) {  
    String hoTen = txtHoTen.getText();  
    lblCauChao.setText(String.format(CAU_CHAO, hoTen));  
}
```







Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

LTV CÔNG NGHỆ JAVA

Module 1 – Bài 2: **Kiểu dữ liệu cơ sở**

Ngành LT & CSDL

www.t3h.vn



2014

5014



Nội dung



1. Kiểu dữ liệu và chuyển đổi kiểu dữ liệu
2. Biến và hằng số
3. Các xử lý trên kiểu String



Kiểu dữ liệu và chuyển đổi kiểu dữ liệu



Kiểu dữ liệu

Kiểu cơ sở (Primitive data types)

Kiểu tham chiếu (Reference data types)

Kiểu dữ liệu	Kích thước	Miền giá trị	Kiểu dữ liệu	Mô tả
byte	8 bit	-2 ⁷ .. 2 ⁷ -1	Array	Kiểu mảng
short	16 bit	-2 ¹⁵ .. 2 ¹⁵ -1	Class	Kiểu lớp đối tượng
int	32 bit	-2 ³¹ ..2 ³¹ -1	Interface	Kiểu giao diện lập trình
long	64 bit	-2 ⁶³ ..2 ⁶³ -1	String	Chuỗi ký tự
float	32 bit	1.40129846432481707e-45 .. 3.40282346638528860e+38		
double	64 bit	1.40129846432481707e-45 .. 3.40282346638528860e+38		
Boolean		true hoặc false		
char	16 bit	Ký tự Unicode 16 bit		



Kiểu dữ liệu và chuyển đổi kiểu dữ liệu



❑ Cơ chế chuyển đổi kiểu dữ liệu

- Tại sao cần chuyển đổi kiểu dữ liệu ?



```
c:\Windows\system32\cmd.exe
c:\BTJU>javac Tong2SoNguyen.java
c:\BTJU>java Tong2SoNguyen
Nhập số nguyên thứ nhất: 2
Nhập số nguyên thứ hai: 3
2+3=23
c:\BTJU>
```

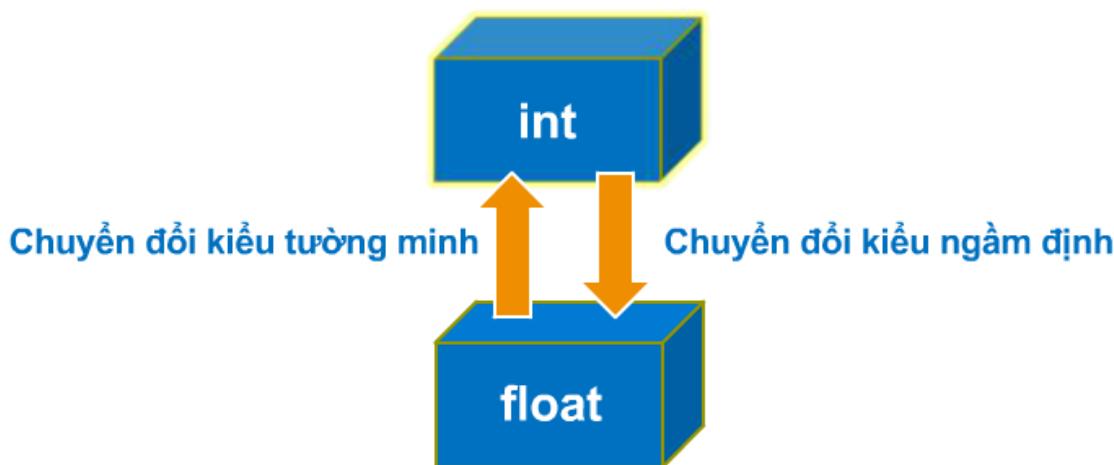


Kiểu dữ liệu và chuyển đổi kiểu dữ liệu



❑ Cơ chế chuyển đổi kiểu dữ liệu

- Trạng thái 2 kiểu của việc chuyển đổi kiểu



Kiểu dữ liệu và chuyển đổi kiểu dữ liệu



□ Cơ chế chuyển đổi kiểu dữ liệu

- Chuyển đổi kiểu ngầm định
 - Hai kiểu phải tương thích
 - Kiểu đích có tầm giá trị lớn hơn kiểu nguồn → qui luật của sự phát triển


```
int i=1000;
long l= i;
```
- Chuyển đổi kiểu tường minh
 - Chuyển từ kiểu có độ chính xác cao sang kiểu có độ chính xác thấp hơn


```
long l=1000;
int i=(int)l;
```



Nội dung



1. Kiểu dữ liệu và chuyển đổi kiểu dữ liệu
2. Biến và hằng số
3. Các xử lý trên kiểu String



Biến và hằng số



Khái niệm biến

- Là một đơn vị lưu trữ trên bộ nhớ của máy tính, lưu trữ các giá trị có thể được dùng để tính toán xử lý
- Biến có thể lưu trữ dữ liệu dạng chuỗi, dạng số, dạng ngày giờ... tùy thuộc vào kiểu dữ liệu của biến
- Biến phải được khai báo trước khi dùng



Biến và hằng số



Khai báo biến

- Cú pháp: **KiểuDữLiệu TênBiến;**

- Ví dụ:

```
int tuoi;
String ten;
double luong;
```



Biến và hằng số



□ Hằng số

- Hằng là những giá trị không thay đổi trong suốt quá trình hoạt động của ứng dụng
- Khai báo hằng
 - Cú pháp: **final** KiểuDữLiệu TênHằng;
 - Ví dụ:
 - **final double PI=3.14;**



Biến và hằng số



□ Quy ước đặt tên biến và hằng số

- Tên biến có thể gồm các ký tự chữ, ký tự số, dấu gạch dưới ‘_’, và dấu ‘\$’
- Tên biến phải bắt đầu bằng ký tự chữ
- Tên biến không được trùng với từ khóa và từ dành riêng của Java
- Tên biến có phân biệt chữ hoa – thường
- Nếu tên biến chỉ gồm một từ đơn, tên biến nên viết chữ thường
- Nếu tên biến gồm nhiều từ, ký tự đầu của từ đầu viết thường, ký tự đầu của mỗi từ kế tiếp viết hoa



Nội dung



1. Kiểu dữ liệu và chuyển đổi kiểu dữ liệu
2. Biến và hằng số
3. Các xử lý trên kiểu String



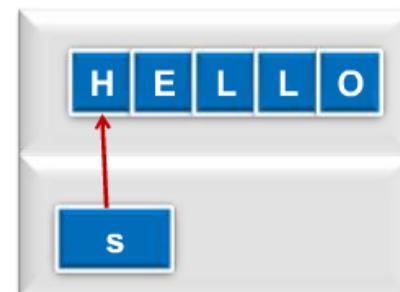
Các xử lý trên kiểu String



❑Kiểu String

- String: là chuỗi các ký tự
- Trong Java, String là lớp quản lý dữ liệu văn bản
- Khai báo:

```
String s1 = new String();  
String s2 = "Hello";
```



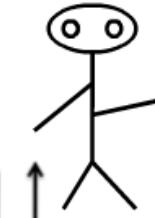
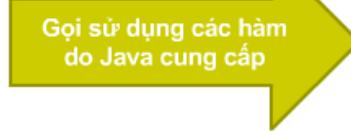
Các xử lý trên kiểu String



□ Ý nghĩa sử dụng



Thực hiện các nghiệp vụ có xử lý chuỗi



Thư viện hàm của Java

Có những công việc được thực hiện đi thực hiện lại nhiều lần liên quan đến việc xử lý chuỗi => Thư viện hàm chuỗi của Java được xây dựng và cung cấp cho developer để giải quyết những công việc đó nhằm tiết kiệm thời gian



Các xử lý trên kiểu String



□ Các phương thức xử lý trên kiểu String

Vấn đề:

Cần có các giá trị để phục vụ cho việc hiển thị và tính toán

Ví dụ:

- Tính chiều dài của chuỗi s
- Nối chuỗi s1 vào chuỗi s
- Lấy một ký tự tại vị trí index trong chuỗi s
- So sánh hai chuỗi s1 và s2
- Tìm vị trí xuất hiện đầu tiên của chuỗi s2 trong chuỗi s

Giải quyết:

Sử dụng hàm chuỗi trong thư viện hàm của Java

```
String s = "Happy ";
```

```
String s1 = "New Year";
```

// Tính chiều dài chuỗi

```
int len = s.length(); // 6
```

// Nối chuỗi s1 vào chuỗi s: tương đương s + s1

```
s.concat(s1); // Happy New Year
```

// Lấy một ký tự tại vị trí số 8 của s

```
char result = s.charAt(8); // e
```

// So sánh hai chuỗi s1 và s2

```
String s2 = "New Year";
```

```
s2.compareTo(s1); // 0 (trả về 0, <0 hoặc >0)
```

// Tìm vị trí xuất hiện đầu tiên của chuỗi s2 trong chuỗi s

```
s.indexOf(s2); // 6 (vị trí đầu tiên)
```





Các xử lý trên kiểu String

❑ Các phương thức xử lý trên kiểu String

Vấn đề:

Cần có các giá trị để phục vụ cho việc hiển thị và tính toán

Ví dụ:

- Tìm vị trí xuất hiện cuối cùng của chuỗi s2 trong chuỗi s
- Thay thế chuỗi s1 bằng chuỗi s2 trong chuỗi s
- Loại bỏ các khoảng trắng thừa của chuỗi s3
- Tạo chuỗi con s4 từ chuỗi s từ vị trí số 6

Giải quyết:

Sử dụng hàm chuỗi trong thư viện hàm của Java

```
String s = "Happy New Year";
```

```
String s1 = "Happy";
```

```
String s2 = "New";
```

```
// Tìm vị trí xuất hiện cuối cùng  
// của chuỗi s2 trong chuỗi s
```

```
s.lastIndexOf(s2); // 6 (trả về -1  
nếu không tìm thấy)
```

```
// Thay thế chuỗi s1 bằng chuỗi s2  
// trong chuỗi s
```

```
s.replace(s1, s2); // New New Year
```

```
// Loại bỏ các khoảng trắng thừa của  
// chuỗi s3
```

```
String s3 = "Hello Bi";
```

```
s3.trim(); // Hello Bi
```

```
// Tạo chuỗi con s4 từ chuỗi s từ vị  
// trí số 6
```

```
String s4 = s.substring(6); // New  
Year
```



Các xử lý trên kiểu String

❑ Lớp StringBuilder

- Quản lý một chuỗi có thể thay đổi kích thước và nội dung

```
// Khởi tạo
StringBuilder()
StringBuilder(int capacity)
StringBuilder(String s)
```

```
// Phương thức
append()
insert()
delete()
reverse()
```



Các xử lý trên kiểu String



□ Lớp StringBuilder

- Các hàm khởi tạo của lớp
 - **StringBuilder()** : mặc định tạo ra một đối tượng
StringBuilder có thể lưu giữ được 16 ký tự
 - **StringBuilder(int capacity)** : tạo ra một đối tượng
StringBuilder có thể lưu giữ được capacity ký tự
 - **StringBuilder(String s)** : tạo một đối tượng
StringBuilder lấy thông tin từ chuỗi s



Các xử lý trên kiểu String



□ Lớp StringBuilder

- Ví dụ:

```
StringBuilder sb = new StringBuilder();
sb.append("Wellcome to ");
sb.append("Java ");
sb.append("world");
System.out.println(sb);
// Wellcome to Java world
```



Các xử lý trên kiểu String



□ Lớp StringBuilder

- `append()`
 - `append(char c)`: gắn thêm chuỗi đại diện của ký tự c vào chuỗi
 - `append(int i)`: gắn thêm chuỗi đại diện của số nguyên i vào chuỗi
 - `append(Object obj)`: gắn thêm chuỗi đại diện của đối tượng obj vào chuỗi
 - `append(String s)`: gắn thêm chuỗi cụ thể s vào chuỗi



Các xử lý trên kiểu String



□ Lớp StringBuilder

- `insert()`
 - `insert(int offset, char c)`: chèn chuỗi đại diện của ký tự c vào chuỗi
 - `insert(int offset int i)`: chèn chuỗi đại diện của số nguyên i vào chuỗi
 - `insert(int offset, Object obj)`: chèn chuỗi đại diện của đối tượng obj vào chuỗi
 - `append(int offset, String s)`: chèn chuỗi cụ thể s vào chuỗi



Các xử lý trên kiểu String



□ Lớp StringBuilder

- `delete()`
 - `delete(int start, int end)`: xóa các ký tự từ `start` tới `end` ra khỏi chuỗi
 - `deleteCharAt(int index)`: xóa ký tự tại vị trí `index` ra khỏi chuỗi
- `reverse()`: đảo ngược chuỗi trong đối tượng đang có, có kết quả trả về là một tham chiếu đến đối tượng này



Các xử lý trên kiểu String



□ Lớp StringTokenizer

- Chia chuỗi thành các chuỗi con

```
// Khởi tạo
StringTokenizer(String str)
StringTokenizer(String str, String delim)
StringTokenizer(String str, String delim, boolean returnDelims)
```

```
// Phương thức
countTokens()
hasMoreTokens()
nextToken()
hasMoreElements()
nextElement()
```



Các xử lý trên kiểu String



□ Lớp StringTokenizer

- Các hàm khởi tạo của lớp
 - **StringTokenizer(String str)**: xây dựng một chuỗi tokenizer cho một chuỗi cụ thể str. Sử dụng các **delim** mặc định là: “\t\n\r\f”
 - **StringTokenizer(String str, String delim)**: xây dựng một chuỗi tokenizer cho một chuỗi cụ thể str. Các ký tự trong delim là ký tự để phân tách cách token.
 - **StringTokenizer(String str, String delim, boolean returnDelims)**: xây dựng một chuỗi tokenizer cho một chuỗi cụ thể str. Nếu returnDelims = true thì mỗi delim được trả về là một chuỗi có chiều dài =1, ngược lại thì delim sẽ được bỏ qua và xem như là một dấu phân cách giữa các token



Các xử lý trên kiểu String



□ Lớp StringTokenizer

● Ví dụ

```
String s = "Lập trình Java";
StringTokenizer st = new StringTokenizer(s);
while (st.hasMoreTokens())
    System.out.println(sr.nextToken());

s = "Lập/trình/Java";
st = new StringTokenizer(s,"/");
while (st.hasMoreTokens())
    System.out.println(sr.nextToken());
```



Các xử lý trên kiểu String



❑ Lớp StringTokenizer

- countTokens(): tính số lượng token trong chuỗi còn lại khi sử dụng delim mặc định
- hasMoreTokens(): kiểm tra xem có còn token trong chuỗi các token hay không. (*)
- nextToken(): trả về token tiếp theo trong chuỗi các token (**)
- hasMoreElements(): tương tự như (*), tuy nhiên nó ở trong Enumeration<Object>
- nextElement(): tương tự như (**), tuy nhiên giá trị trả về là Object





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

LTV CÔNG NGHỆ JAVA

Module 1 – Bài 3: **Giao diện 1**

Ngành LT & CSDL

www.t3h.vn



2014

5014



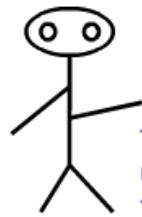
Nội dung

1. Giới thiệu
2. Các điều khiển - thể hiện

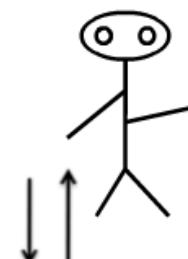
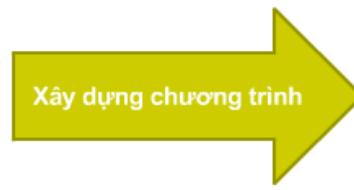


Giới thiệu

□ Ý nghĩa sử dụng



Thực hiện
nhiệm vụ f
trong thực tế
có dùng các
biểu mẫu



Các điều khiển
(thể hiện)

Các điều khiển (thể hiện) là các đối tượng dành cho người dùng tương tác với chương trình để thực hiện một công việc nào đó, hoặc xem một nội dung



Nội dung



1. Giới thiệu
2. Các điều khiển và thể hiện



Các điều khiển – thể hiện



- ❑ Là một lớp điều khiển nằm trong thư viện Swing – GUI của Java

❑ Thuộc tính chung

- Variable: tên điều khiển
- Class: điều khiển thuộc lớp ...
- Text: giá trị mặc định
- Icon: hình đại diện



Các điều khiển – thẻ hiện



❑ Phương thức chung

- setBounds(int x, int y, int width, int height): Đặt kích thước cho điều khiển
- setLocation(int x, int y): Đặt vị trí cho điều khiển
- setFont(Font font): Đặt font chữ cho điều khiển
- setSize(int width,int height): Đặt kích thước cho điều khiển



Các điều khiển – thẻ hiện



❑ JFrame



- Được dùng để chứa các điều khiển

khác như TextField, Label, Button...

- Thuộc tính:

- title: tiêu đề của khung
- iconImage: hình đại diện
- type: loại khung



Các điều khiển – thẻ hiện



❑ JFrame

- Khởi tạo

- `JFrame():` Khởi tạo một frame mới invisible
- `JFrame(String title):` Tạo một frame mới invisible có tiêu đề được chỉ định

- Phương thức cơ bản

- `setVisible(boolean b):` Đặt JFrame ẩn/hiện
- `setTitle(String title):` Đặt tiêu đề cho JFrame
- `setResizable(boolean b):` Cho phép JFrame có được thay đổi kích thước hay không

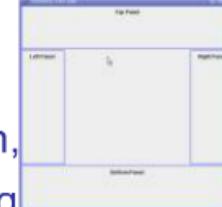


Các điều khiển – thẻ hiện



❑ JPanel

- Được dùng để chứa các điều khiển Button, TextField... trong một GUI phức tạp. Trong 1 Frame có thể chứa nhiều Panel



- Khởi tạo

- `JPanel():` Khởi tạo một panel mới invisible

- Phương thức cơ bản

- `add(Component comp):` Đưa một điều khiển vào Panel



Các điều khiển – thẻ hiện



❑ JLabel

- Được dùng để hiển thị một dòng đơn dữ liệu, thông tin chỉ đọc, hình ảnh hoặc cả thông tin và hình ảnh.

- Khởi tạo

- JLabel(): Tạo thẻ hiện JLabel là chuỗi rỗng
- JLabel(String text): Tạo thẻ hiện JLabel là chuỗi
- JLabel(String text, int horizontalAlignment): Tạo thẻ hiện JLabel là chuỗi và có canh lề horizontal alignment.



Các điều khiển – thẻ hiện



❑ JLabel

- Phương thức cơ bản

- setText(String text): Đặt nội dung hiển thị cho JLabel
- getText(): Lấy nội dung hiển thị của JLabel
- setToolTipText(String text): Đặt Tooltip cho JLabel (Khi di chuột trên Label sẽ hiển thị text tip)
- setForeground(Color fg): Đặt màu chữ
- setIcon(Icon icon): Đặt icon cho JLabel



Các điều khiển – thẻ hiện



❑ JLabel – ImageIcon

- Hiển thị hình ảnh, hoặc cả hình ảnh và thông tin
- Khởi tạo
 - JLabel(Icon image, int horizontalAlignment): Tạo thẻ hiện JLabel chỉ định một hình ảnh và horizontal alignment
 - JLabel(String text, Icon icon, int horizontalAlignment): Tạo thẻ hiện JLabel chỉ định nội dung, hình ảnh và horizontal alignment



Các điều khiển – thẻ hiện



❑ JLabel – ImageIcon

- Ví dụ: Hiển thị hình ảnh: C:/canh_cut.jpg vào Jlabel

```
ImageIcon image = new
ImageIcon("C:/canh_cut.jpg");
```

```
JLabel lblShowImage = new JLabel(image,
JLabel.CENTER);
```



Các điều khiển – thẻ hiện

❑ JTextField

- Được dùng để nhập liệu hoặc chỉnh sửa/ hiển thị nội dung
- Khởi tạo
 - JTextField(): Tạo thẻ hiện JTextField rỗng
 - JTextField(String text): Tạo thẻ hiện JTextField có nội dung mặc định
 - JTextField(String text, int columns): Tạo thẻ hiện JTextField có nội dung mặc định và có độ rộng



Các điều khiển – thẻ hiện

❑ JTextField

- Phương thức cơ bản
 - setText(String text): Đặt nội dung cho JTextField
 - getText(): Lấy nội dung đang có của JTextField
 - toString(): Chuyển nội dung thành chuỗi
 - setEditable(boolean b): Cho phép nhập/ hiệu chỉnh dữ liệu



Các điều khiển – thẻ hiện

❑ JButton

- Cho phép người dùng click vào để kích hoạt một hành động được chỉ định
- Khởi tạo
 - JButton(): Tạo một JButton không có nội dung hoặc icon
 - JButton(Action a): Tạo một Button có thuộc tính được lấy từ một Action
 - JButton(Icon icon): Tạo một Button có icon
 - JButton(String text): Tạo một Button có nội dung
 - JButton(String text, Icon icon) : Tạo một JButton có nội dung và icon



16

Các điều khiển – thẻ hiện

❑ JButton

- Phương thức cơ bản
 - addActionListener(ActionListener l): Thêm một ActionListener cho Button
 - getAction(): Lấy hành động hiện tại của JButton



Các điều khiển – thẻ hiện

❑ JFileChooser



- Được dùng để chọn một hay nhiều tập tin để xử lý (đọc, ghi, hiển thị...)

- Khởi tạo
 - JFileChooser(): Tạo thẻ hiện JFileChooser
 - JFileChooser(String currentDirectoryPath): Tạo thẻ hiện JFileChooser với đường dẫn được thiết lập.
 - JFileChooser(File currentDirectory, FileSystemView fsv): Tạo thẻ hiện JFileChooser với thư mục chỉ định và FileSystemView



Các điều khiển – thẻ hiện

❑ JFileChooser



- Phương thức cơ bản

- setSelectedFile(File f): Thiết lập tập tin được chọn mặc định
- showOpenDialog(): Thiết lập cơ chế hiển thị hộp thoại mở
- setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY) : Chỉ hiển thị thư mục trong hộp thoại
- setFileHidingEnabled(true/false): Hiển thị/ không hiển thị file ẩn
- setControlButtonsAreShown(true/false): Hiển thị/ không hiển thị các nút điều khiển



Các điều khiển – thẻ hiện



❑ JFileChooser

- Ví dụ: Tạo một file chooser

```
JButton button = new JButton("Select File");
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        JFileChooser fileChooser = new JFileChooser();
        int returnValue = fileChooser.showOpenDialog(null);
        if (returnValue == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();
            System.out.println(selectedFile.getName());
        }
    }
});
```





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

LTV CÔNG NGHỆ JAVA

Module 1 – Bài 4: *Xử lý tập tin*

Ngành LT & CSDL

www.t3h.vn



2014

5014



Nội dung



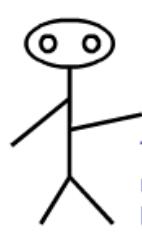
1. Tổng quan về I/O Stream
2. Tổng quan về làm việc với tập tin (File I/O)



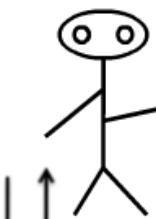
Tổng quan về I/O Stream



□ Ý nghĩa sử dụng



Thực hiện các
nhiệm vụ cần
lưu trữ và xử
 lý dữ liệu
được lưu trữ



Tập tin

Có những dữ liệu cần phải được lưu trữ và xử lý thường
xuyên => Sử dụng tập tin



Tổng quan về I/O Stream



Khái niệm về I/O Stream



❑ Tất cả các xử lý nhập/xuất dữ liệu đều được gọi chung là luồng nhập xuất (I/O Stream)

- Nhập dữ liệu từ bàn phím
- Lấy dữ liệu từ một chương trình khác
- Ghi dữ liệu ra đĩa
- Xuất dữ liệu ra màn hình, máy in,
- ...

❑ Luồng hỗ trợ nhiều kiểu dữ liệu khác nhau

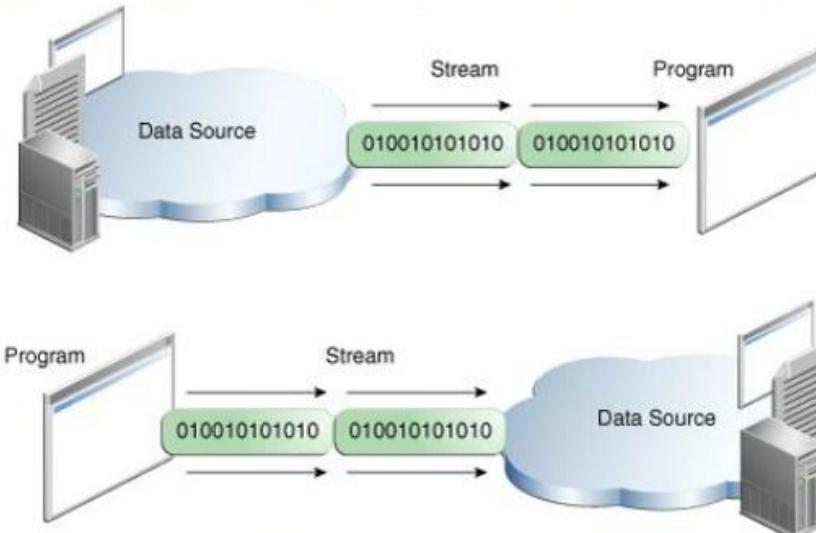
❑ Một số luồng chỉ truyền dữ liệu, một số luồng có thể xử lý và chuyển đổi dữ liệu



Khái niệm về I/O Stream



- ☐ Tất cả các luồng đều có chung mô hình hoạt động: luồng là một dãy dữ liệu tuần tự



LTV Công nghệ Java – Module 1

6

Byte Stream



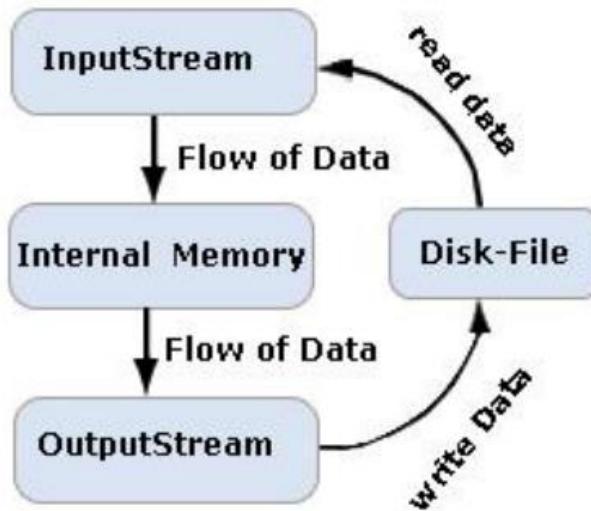
- ☐ Luồng byte (Byte Stream) hỗ trợ việc xuất nhập dữ liệu trên byte, thường được dùng khi đọc ghi dữ liệu nhị phân
- ☐ Tất cả các lớp xử lý luồng byte đều là lớp con của InputStream và OutputStream



LTV Công nghệ Java – Module 1

7

Byte Stream



Đọc dữ liệu với Byte Stream



```

int i; s = "";
try(FileInputStream in = new FileInputStream("C:/BTJV/Test.txt")){
    while ((i = in.read()) != -1)
        s += (char)i;
    ta1.setText(s);
}
catch (FileNotFoundException ex1){
    JOptionPane.showMessageDialog(null,"Khong tim thay tap tin");
}
catch (IOException ex2){
    JOptionPane.showMessageDialog(null,"Loi IO: " +
        ex2.getMessage());
}
catch (Exception ex){
    JOptionPane.showMessageDialog(null,"Loi: " + ex.getMessage());
}
  
```



Ghi dữ liệu với Byte Stream



```

try(FileOutputStream out = new FileOutputStream("C:/BTJV/Text.txt")){
    s = ta1.getText();
    for(int i = 0; i < s.length(); i++)
        out.write(s.charAt(i));
    JOptionPane.showMessageDialog(null,"Ghi file thành công");
}
catch (IOException ex2){
    JOptionPane.showMessageDialog(null,"Loi IO: " + ex2.getMessage());
}
catch(Exception ex){
    JOptionPane.showMessageDialog(null,"Loi: " + ex.getMessage());
}
}

```



Character Stream



- Luồng ký tự (Character Stream) hỗ trợ đọc ghi luồng ký tự Unicode**
- Tất cả các lớp luồng ký tự đều là lớp con của Reader và Writer**



Đọc dữ liệu với Character Stream

```

int i; s = "";
try(FileReader in = new FileReader("C:/BTJV/Test1.txt")){
    while ((i = in.read()) != -1)
        s += (char)i;
    tal.setText(s);
}
catch (FileNotFoundException ex1){
    JOptionPane.showMessageDialog(null,"Khong tim thay tap tin");
}
catch (IOException ex2){
    JOptionPane.showMessageDialog(null,"Loi IO: " + ex2.getMessage());
}
catch(Exception ex){
    JOptionPane.showMessageDialog(null,"Loi: " + ex.getMessage());
}

```



Ghi dữ liệu với Character Stream

```

//String[] ss = {"Xin chào","Chào mừng đến với lập trình Java"};
s = tal.getText();
try(FileWriter out = new FileWriter("C:/BTJV/Test1.txt")){
    //for(i=0;i<ss.length;i++)
    //    out.write(ss[i]+"\r\n");
    out.write(s);
    JOptionPane.showMessageDialog(null,"Ghi file thành công");
}
catch (IOException ex2){
    JOptionPane.showMessageDialog(null,"Loi IO: " + ex2.getMessage());
}
catch(Exception ex){
    JOptionPane.showMessageDialog(null,"Loi: " + ex.getMessage());
}

```



Sử dụng vùng đệm trong đọc ghi dữ liệu



❑ Các thao tác đọc/ghi thông thường tác động trực tiếp đến đĩa cứng

- Không hiệu quả
- Tốn chi phí truy xuất đĩa cứng
- Dễ tổn hại tập tin và đĩa cứng

❑ Sử dụng vùng đệm:

- Đọc dữ liệu từ vùng đệm
- Ghi dữ liệu vào vùng đệm

❑ Hiệu quả và an toàn hơn cho các thao tác đọc/ghi tập tin



Sử dụng vùng đệm trong đọc ghi dữ liệu



❑ Chuyển từ không sử dụng vùng đệm sang sử dụng vùng đệm: dùng buffered stream

- BufferedInputStream
- BufferedOutputStream
- BufferedReader
- BufferedWriter

❑ Ví dụ:

- BufferedReader in = new BufferedReader(new FileReader("Text.txt "));
- BufferedWriter out = new BufferedWriter(new FileWriter("Text.txt "));



Sử dụng vùng đệm trong đọc ghi dữ liệu



```
s = "";
try(BufferedReader in = new BufferedReader(new
    FileReader("C:/BTJV/Test2.txt"))){
    String chuoi = "";
    while ((chuoi = in.readLine())!=null)
        s += chuoi;
    tal.setText(s);
}
catch (FileNotFoundException ex1){
    JOptionPane.showMessageDialog(null,"Khong tim thay tap tin");
}
catch (IOException ex2){
    JOptionPane.showMessageDialog(null,"Loi IO: " + ex2.getMessage());
}
catch(Exception ex){
    JOptionPane.showMessageDialog(null,"Loi: " + ex.getMessage());
}
```



Sử dụng vùng đệm trong đọc ghi dữ liệu



```
//String[] ss={"Xin chào","Chào mừng đến với lập trình Java",
//Chúc bạn học tốt";
s = tal.getText();
try(BufferedWriter out = new BufferedWriter(new
    FileWriter("C:/BTJV/Test2.txt"))){
    //for(i=0;i<ss.length;i++)
    //    out.write(ss[i]+\r\n");
    out.write(s);
    JOptionPane.showMessageDialog(null,"Ghi file thành công");
}
catch (IOException ex2){
    JOptionPane.showMessageDialog(null,"Loi IO: " + ex2.getMessage());
}
catch(Exception ex){
    JOptionPane.showMessageDialog(null,"Loi: " + ex.getMessage());
}
```



Đọc ghi dữ liệu theo định dạng



❑ Chuyển đổi định dạng dữ liệu trong quá trình đọc ghi dữ liệu

- **Scanner:** Phân tách dữ liệu thành các thẻ bài
- **Formatting:** Định dạng dữ liệu theo dạng người dùng có thể xử lý



Đọc ghi dữ liệu theo định dạng



```
s = "";

try (Scanner sc = new Scanner(new BufferedReader(new
    FileReader("C:/BTJV/Test3.txt")))){
    while (sc.hasNext())
        s += sc.next();
    ta1.setText(s);
}

catch (FileNotFoundException ex){
    JOptionPane.showMessageDialog(null,"Khong tim thay tap tin");
}

catch (Exception ex){
    JOptionPane.showMessageDialog(null,"Loi: " + ex.getMessage());
}
```



Đọc ghi dữ liệu theo định dạng



```
//String[] ss={"Xin chào","Chào mừng đến với lập trình Java",
// "Chúc bạn học tốt";
s = ta1.getText();
try(PrintWriter pw = new PrintWriter(new
FileWriter("C:/BTJV/Test4.txt"))){
    //for(int i=0;i<ss.length;i++)
    //    pw.println(ss[i]);
    pw.println(s);
    JOptionPane.showMessageDialog(null,"Ghi file thành công");
}
catch (IOException e) {
    JOptionPane.showMessageDialog(null,e.getMessage());
}
```



Đọc ghi dữ liệu với Data Stream



- Hỗ trợ đọc ghi các giá trị thuộc kiểu cơ sở và String theo định dạng nhị phân
- Tất cả các lớp Data Stream đều là lớp con của DataInputStream và DataOutputStream



Đọc ghi dữ liệu với Data Stream



```

double[] gia = { 19.99, 9.99, 15.99 };
int[] soLuong = { 12, 8, 13};
String[] moTa = {"Áo thun", "Túi xách", "Balo"};
try(DataOutputStream out = new DataOutputStream(
        new BufferedOutputStream(new FileOutputStream("C:/BTJV/Test5.txt"))){
    for (int i = 0; i < gia.length; i++) {
        out.writeDouble(gia[i]);out.writeInt(soLuong[i]);out.writeUTF(moTa[i]);
    }
    JOptionPane.showMessageDialog(null,"Ghi file thành công");
}
catch (IOException e) {
    JOptionPane.showMessageDialog(null,e.getMessage());
}

```



Đọc ghi dữ liệu với Data Stream



```

s = "";
try(DataInputStream in = new DataInputStream(
        new BufferedInputStream(new FileInputStream("C:/BTJV/Test5.txt"))){
    while (true){
        double gia = in.readDouble(); int soLuong = in.readInt(); String
        moTa = in.readUTF();
        s += String.format("Ban dat %d %s voi gia %.2f\n",soLuong,moTa,gia);
    }
}
catch (FileNotFoundException ex1){
    JOptionPane.showMessageDialog(null,"Khong tim thay tap tin");
}
catch (EOFException ex){ }
catch (IOException e) {
    JOptionPane.showMessageDialog(null,e.getMessage());
}
finally{         tal.setText(s);          }

```



Đọc ghi đối tượng với Object Stream



- ❑ Hỗ trợ đọc ghi các đối tượng
- ❑ Các đối tượng phải thực thi interface Serializable
- ❑ Tất cả các lớp ObjectOutputStream đều là lớp con của ObjectInputStream và ObjectOutputStream



Đọc ghi đối tượng với Object Stream



```

try(ObjectOutputStream out = new ObjectOutputStream(
        new BufferedOutputStream(new
        FileOutputStream("C:/BTJV/Test6.txt"))){
    NhanVien nv = new NhanVien("Văn Phú Cường",
    2.67,5000000,1);
    out.writeObject(nv);

    nv = new NhanVien("Nguyễn Nam",2.34,3000000,0);
    out.writeObject(nv);

    JOptionPane.showMessageDialog(null,"Ghi file thành công");
}
catch (IOException e) {
    JOptionPane.showMessageDialog(null,e.getMessage());
}

```



Đọc ghi đối tượng với Object Stream



```
s = "";
try(ObjectInputStream in = new ObjectInputStream(
        new BufferedInputStream(new
        FileInputStream("C:/BTJV/Test6.txt"))){
    NhanVien nv = (NhanVien)in.readObject();
    s += nv.xuat();
    nv = (NhanVien)in.readObject();
    s += nv.xuat();
    tal.setText(s);
}
catch (ClassNotFoundException ex){
    JOptionPane.showMessageDialog(null,ex.getMessage());
}
catch (IOException e) {
    JOptionPane.showMessageDialog(null,e.getMessage());
}
```



Nội dung



1. Tổng quan về I/O Stream
2. Tổng quan về làm việc với tập tin (File I/O)



Tổng quan về làm việc với tập tin, thư mục



- Làm việc với đường dẫn tập tin, thư mục**
- Kiểm tra tồn tại tập tin, thư mục**
- Xóa, di chuyển tập tin, thư mục**
- Giám sát sự thay đổi của hệ thống tập tin, thư mục**



Đường dẫn tập tin, thư mục



- Đường dẫn tuyệt đối**
 - Bắt đầu từ thư mục gốc của ổ đĩa
 - Ví dụ: **C:\BTJAVA\XinChao.java**
- Đường dẫn tương đối**
 - Không bắt đầu từ thư mục gốc của ổ đĩa
 - Cần thêm thông tin để xác định vị trí tập tin
 - Ví dụ: **XinChao.java**



Lớp Path



- ❑ **Thể hiện đường dẫn của hệ thống tập tin, thư mục**
- ❑ **Cung cấp các phương thức hỗ trợ thu thập thông tin các thành phần trên đường dẫn tập tin, thư mục**



Lớp Path



❑ Tạo Path

```
Path p1 = Paths.get("XinChao.java");
Path p2 = Paths.get("/ThuMuc/XinChao.java");
```

❑ Rút trích thông tin từ Path

```
Path p = Paths.get("C:\\BTJAVA\\XinChao.java");
 JOptionPane.showMessageDialog(null, p.toString());
 JOptionPane.showMessageDialog(null, p.getFileName());
 JOptionPane.showMessageDialog(null, p.getName());
 JOptionPane.showMessageDialog(null, p.getParent());
 JOptionPane.showMessageDialog(null, p.getRoot());
```



Lớp Files



- Cung cấp các phương thức static hỗ trợ đọc, ghi, xử lý tập tin, thư mục
- Lớp Files làm kết hợp với 1 thẻ hiện của Path để xác định vị trí của tập tin, thư mục



Kiểm tra tồn tại tập tin, thư mục



```
Path p = Paths.get("1.jpg");
if (Files.exists(p))
    JOptionPane.showMessageDialog(null,
        "Có tập tin");
else
    JOptionPane.showMessageDialog(null,
        "Không Có tập tin");
```



Kiểm tra truy xuất tập tin, thư mục



☐ Kiểm tra truy xuất đến tập tin

- Tập tin cho phép đọc:

```
Files.isReadable(path)
```

- Tập tin cho phép ghi:

```
Files.isWritable(path)
```



Xóa tập tin, thư mục



```
try {
    Files.delete(path);
}

catch (NoSuchElementException x) {
    JOptionPane.showMessageDialog(null,x.getMessage());
}

catch (DirectoryNotEmptyException x) {
    JOptionPane.showMessageDialog(null,x.getMessage());
}

catch (IOException x) {
    JOptionPane.showMessageDialog(null,x.getMessage());
}
```



Di chuyển tập tin, thư mục



❑ Sao chép tập tin, thư mục

```
Files.copy(fileNguon, fileDich,  
          REPLACE_EXISTING);
```

❑ Di chuyển tập tin, thư mục

```
Files.move(fileNguon, fileDich,  
          REPLACE_EXISTING);
```





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

LTV CÔNG NGHỆ JAVA

Module 1 – Bài 5: *Mảng cơ sở*

Ngành LT & CSDL

www.t3h.vn



2014

5014



Nội dung

1. Khái niệm mảng
2. Mảng một chiều
3. Các hàm xử lý khác trên mảng



Khái niệm mảng

□ Ý nghĩa sử dụng



Mảng là một **loại biến đặc biệt**, bao gồm một **dãy các ô nhớ có nhiều ô nhớ con** cho phép biểu diễn thông tin dạng danh sách trong thực tế

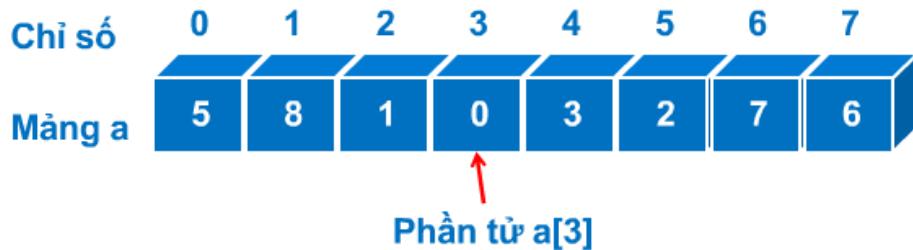
Các phần tử trong mảng có **cùng kiểu dữ liệu với nhau**



Khái niệm mảng



❑ Ví dụ



Khái niệm mảng



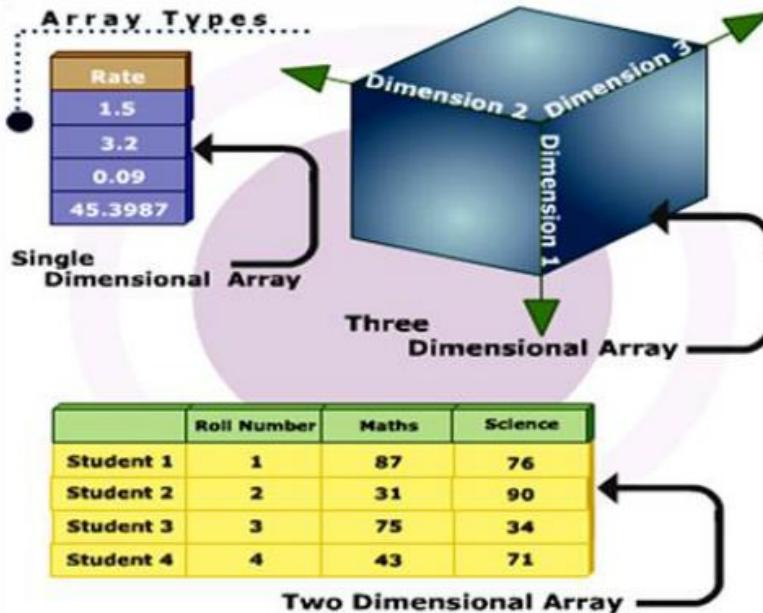
❑ Lợi ích của việc sử dụng mảng

- Mảng là cách tốt nhất cho phép quản lý nhiều phần tử dữ liệu có cùng kiểu tại cùng một thời điểm
- Mảng tạo ra sự tối ưu trong việc quản lý bộ nhớ so với việc sử dụng nhiều biến cho cùng một chức năng
 - Bộ nhớ có thể được gán cho mảng chỉ khi mảng thực sự được sử dụng. Do đó, bộ nhớ không bị tiêu tốn cho mảng ngay khi bạn khai báo mảng.



Khái niệm mảng

❑ Các kiểu mảng



Nội dung

1. Khái niệm mảng
2. Mảng một chiều
3. Các hàm xử lý khác trên mảng



Mảng một chiều



□ Khai báo và khởi tạo

- Khai báo không khởi tạo kích thước và giá trị

KiểuDữLiệu[] TênMảng;

KiểuDữLiệu TênMảng[] ;

- Ví dụ: int[] a; int a[] ;



Mảng một chiều



□ Khai báo và khởi tạo

- Khai báo khởi tạo kích thước nhưng không có giá trị ban đầu

KiểuDữLiệu[] TênMảng = new KiểuDữLiệu[SốPhầnTử];

KiểuDữLiệu TênMảng[] = new KiểuDữLiệu[SốPhầnTử];

- Ví dụ:

int[] a = new int[5];

int a[] = new int[5];



Mảng một chiều



□ Khai báo và khởi tạo

- Khai báo có khởi tạo kích thước và khởi tạo giá trị ban đầu:

```
KiểuDữLiệu[] TênMảng = new KiểuDữLiệu[]
{giá trị 1, giá trị 2, giá trị 3, ...};
hoặc
KiểuDữLiệu[] TênMảng = {giá trị 1, giá trị 2, giá trị 3,
...};
```

- Ví dụ:

```
int[] a = new int[]{2,10,4,8,5};
int[] a = {2, 10, 4, 8, 5};
```



Mảng một chiều



□ Thao tác cơ bản

- Truy xuất giá trị 1 phần tử:

TênMảng[vị_trí_i]

- Vị trí của 1 phần tử trong mảng bắt đầu từ 0
- Vị_trí_i có giá trị từ 0 đến (số phần tử - 1)

Chỉ số	0	1	2	3	4	5	6	7
Mảng a	5	8	1	0	3	2	7	6

- Lấy chiều dài của mảng: thuộc tính **length**

TênMảng.length



Mảng một chiều



□ Duyệt mảng

- Duyệt và xử lý từng phần tử của mảng:

```
for (int i = 0; i < TênMảng.length; i++)
{
    // Xử lý trên phần tử TênMảng[i]
}
```

- Ví dụ: duyệt và xuất mảng

```
int[] a = {1,2,3,4,5};
for (int i = 0; i < a.length; i++)
    System.out.println(a[i]);
```



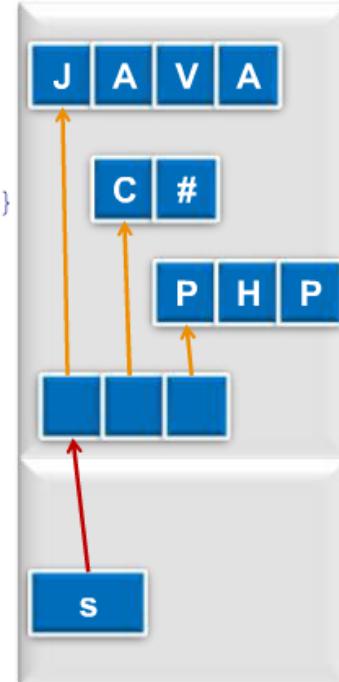
Mảng một chiều



□ Mảng String

- Ví dụ: Duyệt và xuất mảng

```
String[] list = { "JAVA", "C#", "PHP" }
for (int i = 0; i < list.length; i++)
    System.out.println(list[i]);
```



Nội dung



1. Khái niệm mảng
2. Mảng một chiều
3. Các hàm xử lý khác trên mảng



Các hàm xử lý khác trên mảng



Vấn đề:

Cần xử lý mảng một cách nhanh chóng

Ví dụ:

- Sắp xếp mảng số nguyên tăng dần
- So sánh hai mảng số nguyên array1 và array
- Gán giá trị cho các phần tử trong mảng array1
- Sao chép mảng array1 sang mảng array2

Giải quyết:

Sử dụng hàm mảng trong thư viện hàm của Java

```
// import java.util.Arrays;
int array[] = { 2, 5, -2, 6, -3, 8, 0,
7, -9, 4 };
int array1[] = { 2, 5, 6, -3, 8};
// Sắp xếp mảng số nguyên
Arrays.sort(array);
// So sánh hai mảng số nguyên array1 và
array
array1.equals(array);
// Gán giá trị cho các phần tử trong
mảng array1
Arrays.fill(array1, 10); // 10, 10, 10,
10, 10
// Sao chép mảng array1 sang array2
int[] arr2 = Arrays.copyOf(arr1, 6); // 10 10 10 10 10 0
```







Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

LTV CÔNG NGHỆ JAVA

Module 1 – Bài 6: **Giao diện 2**

Ngành LT & CSDL

www.t3h.vn



2014

5014



Nội dung

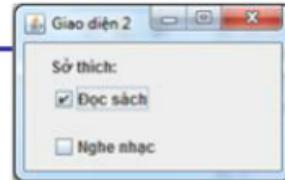


1. Các điều khiển - thể hiện

1. JCheckBox
2. JRadioButton
3. ButtonGroup
4. JComboBox
5. JTable
6. JMenu
7. JMenuItem



Các điều khiển – thể hiện



❑ JCheckBox

- Là một điều khiển có hai trạng thái cho phép người dùng lựa chọn (selected)/ hoặc không (unselected)
- Khởi tạo
 - JCheckBox(): Tạo thể hiện JCheckBox rỗng, unselected
 - JCheckBox(String text): Tạo thể hiện JCheckBox có nội dung, unselected



Các điều khiển – thẻ hiện



● Khởi tạo

- `JCheckBox(Icon icon)`: Tạo thẻ hiện JCheckBox có icon, unselected
- `JCheckBox(String text, Icon icon)`: Tạo thẻ hiện JCheckBox có nội dung và icon, unselected
- `JCheckBox(String text, Icon icon, boolean selected)`: Tạo thẻ hiện JCheckBox có nội dung và icon, selected



Các điều khiển – thẻ hiện



JCheckBox

● Phương thức cơ bản

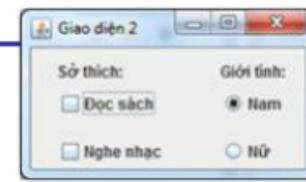
- `setRolloverIcon(Icon rolloverIcon)`: Thiết lập rollover icon cho JCheckBox (tương tự cho selected icon)
- `addItemListener(ItemListener l)`: Thêm icon listener cho JCheckBox
- `setSelected(boolean selected)`: Thiết lập trạng thái cho JCheckBox
- `isSelected()`: Kiểm tra xem checkbox có được chọn hay không



Các điều khiển – thẻ hiện

JButton

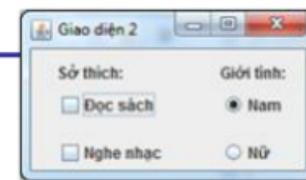
- Tương tự như JCheckBox
- Khởi tạo
 - Tương tự như JCheckBox
- Phương thức cơ bản
 - Tương tự như JCheckBox



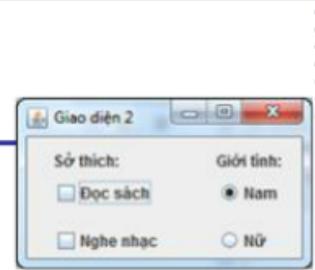
Các điều khiển – thẻ hiện

JButtonGroup

- Được sử dụng để chứa các JRadioButton hoặc JCheckBox
- Khởi tạo
 - ButtonGroup(): Tạo thẻ hiện JButtonGroup
- Phương thức cơ bản
 - add(JRadioButton rb): Thêm một JRadioButton vào JButtonGroup (tương tự cho JCheckBox)



Các điều khiển – thẻ hiện



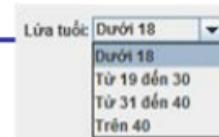
☐ JButtonGroup

- Kiểm tra xem điều khiển nào trong JButtonGroup đang được chọn

```
Enumeration elements = group.getElements();
while (elements.hasMoreElements()) {
    AbstractButton button = (AbstractButton)
elements.nextElement();
    if (button.isSelected()) {
        System.out.println("Điều khiển đang
được chọn là: " + button.getText());
    }
}
```



Các điều khiển – thẻ hiện



☐ JComboBox

- Là một điều khiển cho phép người dùng lựa chọn một hoặc nhiều mục trong danh sách
- Khởi tạo
 - JComboBox(): Tạo thẻ hiện JComboBox rỗng
 - JComboBox(Object listData[]): Tạo thẻ hiện JComboBox chứa danh sách đối tượng



Các điều khiển – thẻ hiện



❑ JComboBox

- Phương thức cơ bản

- add(Object item): thêm một item vào JComboBox
- getSelectedItem(): lấy item được chọn
- setSelecteltem(Object item): thiết lập item được chọn mặc định
- getItemCount(): đếm số lượng item trong JComboBox
- removeAllItems(): xóa tất cả các item trong JComboBox



Các điều khiển – thẻ hiện



❑ JTable

- Là một điều khiển dùng để hiển thị thông tin dưới dạng bảng

- Khởi tạo

- JTable(): Tạo thẻ hiện JTable rỗng
- JTable(int rows, int columns): Tạo thẻ hiện JTable có dòng và cột
- JTable(Object rowData[][], Object columnNames[]): Tạo thẻ hiện JTable có tên của các cột và mỗi dòng là một đối tượng



Các điều khiển – thẻ hiện



❑ JTable

- Phương thức cơ bản

- `getValueAt(int rowIndex, int CollIndex)`: Lấy giá trị từ một ô cụ thể trong JTable
- `setValueAt(String value, int rowIndex, int CollIndex)`: Đặt giá trị cho một ô cụ thể trong JTable
- `setAutoResizeMode(JTable.AUTO_RESIZE_OFF)`: Tắt chức năng tự động thay đổi kích cỡ của Jtable
- `setCellSelectionEnabled(boolean flag)`: Đặt chức năng cho phép chọn ô
- `setCellRenderer(new ImageTableCellRenderer(int width, int height))`: Tạo ô chứa hình ảnh



Các điều khiển – thẻ hiện



❑ JMenu



- Là một điều khiển dùng để chứa các menu item

- Khởi tạo

- `JMenu()`: Tạo thẻ hiện JMenu rỗng
- `Jmenu(String text)`: Tạo thẻ hiện Jmenu có tên



Các điều khiển – thẻ hiện



❑ JMenu

- Phương thức cơ bản

- add(JMenuItem item): Thêm một menu item vào JMenu
- addSeparator(): Thêm separator vào Jmenu



Các điều khiển – thẻ hiện



❑ JMenuItem

- Là một điều khiển nằm trong menu item. Để thực hiện một công việc nào đó

- Khởi tạo

- JMenuItem(): Tạo thẻ hiện JMenuItem rỗng
- JMenuItem(Icon icon): Tạo thẻ hiện JMenuItem có icon
- JMenuItem(String text): Tạo thẻ hiện JMenuItem có tên
- JMenuItem(String text, Icon icon): Tạo thẻ hiện JMenuItem có tên & icon
- JMenuItem(String text, int mnemonic): Tạo thẻ hiện JMenuItem có tên & phím tắt



Các điều khiển – thẻ hiện



❑ JMenuItem

- Phương thức cơ bản

- `setEnabled(boolean flag)`: Thiết lập chế độ ẩn/hiện menu item
- `setMnemonic(KeyEvent.VK_KyTuVietTat)`: Thiết lập phím tắt cho menu item



Các điều khiển – thẻ hiện



❑ JMenuItem

- Ví dụ: tạo menu item có thẻ xác định sự thay đổi trạng thái

```
JMenuItem item = new JMenuItem("Label") {
    public void menuSelectionChanged(boolean isSelected) {
        super.menuSelectionChanged(isSelected);
        if (isSelected) {
            //System.out.println("Menu item is selected");
        } else {
            //System.out.println("Menu item is unselected");
        }
    }
};
```







Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

LTV CÔNG NGHỆ JAVA

Module 1 – Bài 7: *Đối tượng*

Ngành LT & CSDL

www.t3h.vn



2014

5014



Nội dung

1. Tổng quan về Đối tượng
2. Xây dựng lớp đối tượng
3. Package



Tổng quan đối tượng

□ Ý nghĩa sử dụng



Đối tượng (Object) là những **thực thể tồn tại trong **thế giới thực**.**



Tổng quan đối tượng



❑ Ví dụ

- Con người
 - Sinh viên Nguyễn Văn An
 - Nhân viên Trần Thị Thảo
- Đồ vật
 - Phòng học C41
 - Máy in Laser Jet 4300
- Chứng từ
 - Hóa đơn HD01
 - Đơn đặt hàng DDH_14022008_01
- Một chiếc xe hơi cụ thể với các thông tin về chiếc xe: Biển số xe, Hiệu xe, Màu sơn, Hãng sản xuất, Năm sản xuất



Tổng quan đối tượng



❑ Khái niệm lập trình hướng đối tượng

- Là phương pháp thiết kế và phát triển phần mềm dựa trên kiến trúc lớp (class) và đối tượng (object)
- Mô hình hóa các đối tượng trong thế giới thực thành các đối tượng trong chương trình



Tổng quan đối tượng



□ Đặc điểm của LT HĐT

- Tập trung vào dữ liệu thay cho các hàm
- Chương trình được chia thành các đối tượng độc lập.
- Cấu trúc dữ liệu được thiết kế sao cho đặc tả được các đối tượng.
- Dữ liệu được che giấu, bao bọc.
- Các đối tượng trao đổi với nhau thông qua các hàm
- Chương trình được thiết kế theo hướng tiếp cận từ dưới lên



Tổng quan đối tượng



□ Ưu điểm của LT HĐT

- Loại bỏ được những dư thừa, trùng lặp trong việc xây dựng ứng dụng.
- Cài đặt đối tượng giúp xúc tiến việc sử dụng lại, trao đổi giữa các đối tượng với nhau do đó sẽ giảm kích thước, thời gian xử lý,... thời gian phát triển hệ thống, tăng năng xuất lao động.
- Dễ bảo trì, nâng cấp, giảm lỗi.

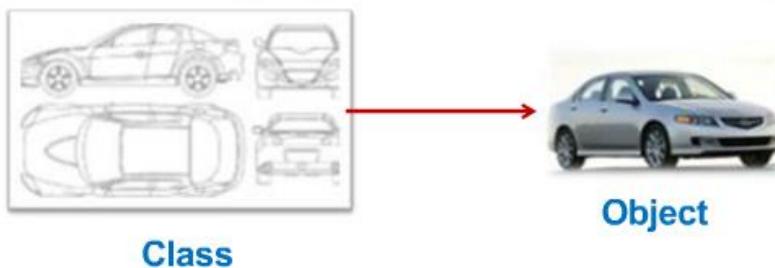


Tổng quan đối tượng



□ Đặc trưng – Tính trừu tượng Abstraction

- Lớp (Class) là một khái niệm trừu tượng, đối tượng là một thể hiện cụ thể của lớp
- Ví dụ:
 - Bản thiết kế của chiếc xe hơi là lớp
 - Chiếc xe hơi được tạo ra từ bản thiết kế là đối tượng



Tổng quan đối tượng



□ Đặc trưng – Tính trừu tượng Abstraction

- Từ những đối tượng giống nhau: có thể trừu tượng hóa thành một lớp.
 - Loại bỏ tính chất phức tạp của đối tượng bằng cách chỉ đưa ra các thuộc tính và phương thức cần thiết của đối tượng trong lập trình.



Tổng quan đối tượng



□ Đặc trưng – Tính đóng gói (Encapsulation)

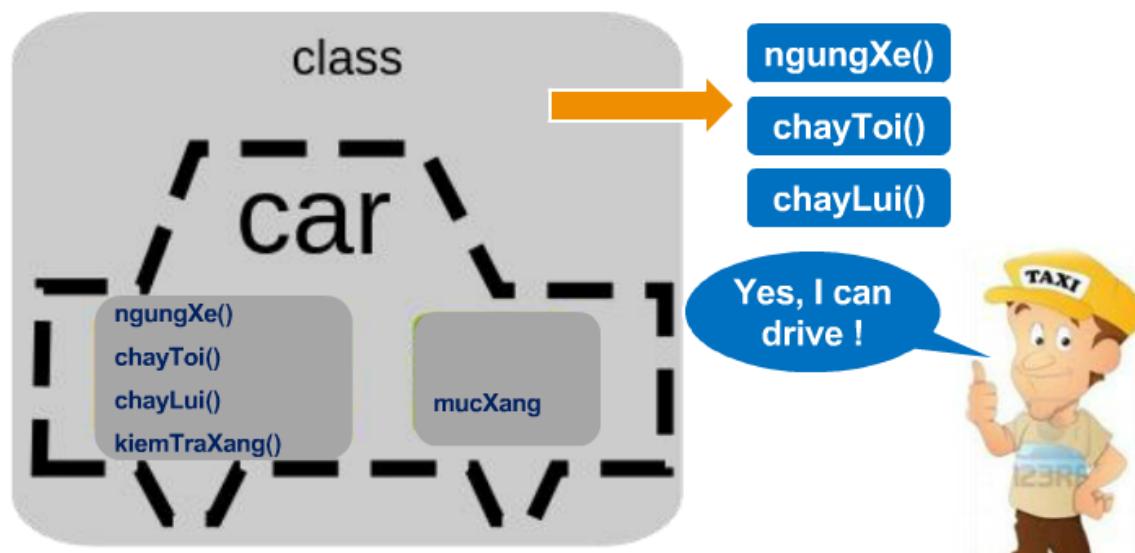
- Mỗi lớp được xây dựng để thực hiện một nhóm chức năng đặc trưng của riêng lớp đó
 - Tất cả mọi thao tác truy xuất vào thành phần dữ liệu từ đối tượng này qua đối tượng khác phải được thực hiện bởi các phương thức (method) của chính đối tượng chứa dữ liệu.
 - Tính đóng gói cho phép dấu thông tin của đối tượng bằng cách kết hợp thông tin và các phương thức liên quan đến thông tin trong đối tượng.



Tổng quan đối tượng



□ Đặc trưng – Tính đóng gói (Encapsulation)



Tổng quan đối tượng



□ Đặc trưng – Tính kế thừa (Inheritance)

- Cho phép xây dựng một lớp mới dựa trên các định nghĩa của một lớp đã có.
 - Lớp đã có gọi là lớp Cha, lớp mới phát sinh gọi là lớp Con
 - Lớp con kế thừa tất cả các thành phần của lớp Cha, có thể mở rộng các thành phần kế thừa và bổ sung thêm các thành phần mới.



Nội dung



1. Tổng quan về Đối tượng
2. Xây dựng lớp đối tượng
3. Package



Xây dựng lớp đối tượng



□ Tạo lớp

- Một lớp dùng định nghĩa một kiểu dữ liệu mới
- Cú pháp khai báo lớp

```
class <TênLớp>
{
    ...
}
```



Xây dựng lớp đối tượng



□ Tạo lớp

- Một lớp dùng định nghĩa một kiểu dữ liệu mới
- Cú pháp khai báo lớp:

```
class <TênLớp>
{
    ...
}
```

- Ví dụ: khai báo lớp tam giác

```
class TamGiac
{
    ...
}
```



Xây dựng lớp đối tượng



□ Quy tắc đặt tên

- Tên lớp nên là một danh từ
- Tên lớp có thể gồm nhiều từ, ký tự đầu tiên của mỗi từ nên viết hoa
- Tên lớp nên đặt đơn giản, dễ nhớ, và có ý nghĩa
- Tên lớp không được trùng với từ khóa của Java
- Tên lớp không thể bắt đầu bằng số, nhưng có thể bắt đầu bằng dấu '\$' và dấu gạch dưới '_'

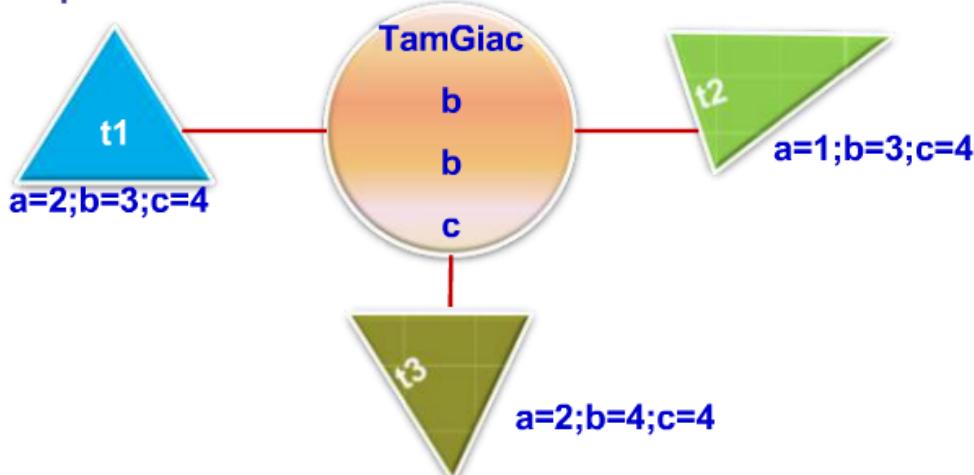


Xây dựng lớp đối tượng



□ Biến thực thể

- Mô tả trạng thái của đối tượng
- Mỗi đối tượng đều có 1 bản sao của biến thực thể



Xây dựng lớp đối tượng



☐ Biến thực thể

- Ví dụ: khai báo biến thực thể mô tả thông tin lớp TamGiac

```
class TamGiac{

    // a, b, c là 3 biến thực thể mô tả thông tin 1
    //đối tượng thuộc lớp tam giác

    public double a,b,c;

}
```



Xây dựng lớp đối tượng



☐ Bổ tử truy xuất

Cấp độ truy xuất

Tùy khóa

Áp dụng cho

Truy xuất tự do

public

interface



protected

class

Không thể truy xuất từ bên ngoài

private

Thành viên của

interface/class



Không có
chỉ định từ
truy xuất

(default)



Xây dựng lớp đối tượng



☐ Bổ tử truy xuất

Chỉ định từ truy xuất	Các thành viên sẽ thấy trong			
	Class	Package	Lớp con (trong package khác)	Ngoài
public	yes	yes	yes	yes
protected	yes	yes	yes	no
private	yes	no	no	no
Không có chỉ định từ truy xuất	yes	yes	no	no



Xây dựng lớp đối tượng



☐ Phương thức khởi tạo

- Là phương thức (method) đặc biệt dùng để khởi tạo các biến thành viên của lớp
- Tên của phương thức trùng với tên lớp và không có kiểu trả về
- Được gọi tự động khi đối tượng được tạo ra
- Có hai loại phương thức:
 - Phương thức mặc định
 - Phương thức có tham số



Xây dựng lớp đối tượng



□ Phương thức khởi tạo

- Ví dụ: Khai báo phương thức mặc định cho lớp tam giác vừa tạo

```
class TamGiac{

    public double a,b,c;
    // hàm dựng mặc định

    public TamGiac(){
        // xử lý khởi tạo giá trị mặc định
        // cho các biến thành viên
        a = b = c = 0;
    }

}
```



Xây dựng lớp đối tượng



□ Phương thức khởi tạo

- Ví dụ: Khai báo hàm dựng có tham số cho lớp tam giác vừa tạo

```
class TamGiac{

    public double a,b,c;
    public TamGiac(){

        // xử lý khởi tạo giá trị mặc định cho biến thành viên
    }

    public TamGiac(int _a,int _b,int _c){
        a = _a; b = _b; c = _c;
    }

}
```



Xây dựng lớp đối tượng



□ Phương thức khởi tạo

- Đối tượng (ĐT) được khai báo tương trưng cho lớp
- Quá trình tạo ĐT cho lớp gồm hai bước:
 - Đầu tiên, một biến có kiểu của lớp được khai báo, biến này chưa được định nghĩa là một ĐT. Nó chỉ là biến tham chiếu đến ĐT.
 - Tiếp theo, một bản sao chép ĐT của lớp trên bộ nhớ được tạo ra và được gán cho biến. Điều này được thực hiện bằng toán tử new
- Toán tử new cấp phát bộ nhớ động cho ĐT và trả về tham chiếu tới nó
- Tất cả các ĐT của lớp phải được cấp phát động



Xây dựng lớp đối tượng

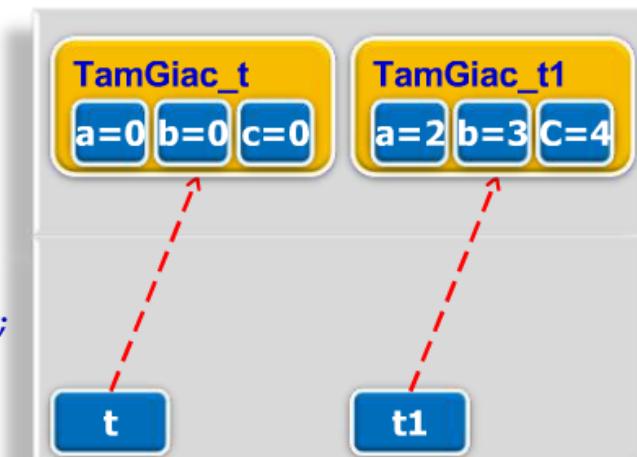


□ Phương thức khởi tạo

- Ví dụ: Khai báo đối tượng có kiểu TamGiac

```

TamGiac t;
t = new TamGiac();
TamGiac t1;
t1 = new TamGiac(2,3,4);
  
```



Xây dựng lớp đối tượng



□ Phương thức khởi tạo

- Ví dụ: Truy xuất các thành phần của đối tượng

```
TamGiac t;  
  
t = new TamGiac();  
  
t.a = 2;  
t.b = 3;  
t.c = 4;  
  
System.out.printf("Tam giac (%d, %d, %d)",  
                  t.a, t.b, t.c);
```



Xây dựng lớp đối tượng



□ Phương thức xử lý

- Một phương thức được định nghĩa để cài đặt cho một hành động của đối tượng
- Cú pháp:

```
BắtTừTruyXuất KiểuDữLiệu  
  
TênPhươngThức (DanhSáchThamSố) {  
    //xử lý của phương thức  
}
```

Phương thức

TamGiac
a
b
c
tinhChuVi()
tinhDienTich()



Xây dựng lớp đối tượng



□ Phương thức xử lý

- Ví dụ: Khai báo phương thức tính chu vi cho lớp tam giác

```
class TamGiac{
    // biến thực thể
    public double a, b, c;
    // hàm dựng mặc định
    public TamGiac() {}
    // phương thức tính chu vi
    public double tinhChuVi()
    {
        return a + b + c;
    }
}
```

LTV Công nghệ Java – Module 1

28

Xây dựng lớp đối tượng



□ Phương thức xử lý

- Ví dụ: tính chu vi tam giác

```
TamGiac t = new TamGiac();
t.a = 2;
t.b = 3;
t.c = 4;
System.out.printf("Tam giac (%f, %f, %f)",
                   t.a, t.b, t.c);
System.out.printf("Chu vi tam giac: %f",
                   t.tinhChuVi());
```



LTV Công nghệ Java – Module 1

29

Xây dựng lớp đối tượng



□ Truyền tham trị

- Được sử dụng cho các kiểu dữ liệu cơ bản, mọi thay đổi diễn ra bên trong phương thức không ảnh hưởng đến giá trị truyền vào

□ Truyền tham chiếu

- Thay đổi bên trong phương thức sẽ làm thay đổi giá trị của tham số truyền vào
- Tham số của phương thức có kiểu dữ liệu là tham chiếu sẽ được truyền theo kiểu tham trị chứ không phải kiểu tham chiếu. Ví dụ: khi phương thức kết thúc, tham chiếu này vẫn trả đến cùng đối tượng khi truyền vào



Xây dựng lớp đối tượng



□ Truyền tham số cho phương thức

- Ví dụ: Truyền tham trị

```
public static void Swap(int a,int b){
    int temp = a; a = b; b = temp;
}

public static void main(String args[]){
    int a = 1, b = 2;
    System.out.printf("a = %d, b = %d", a, b);
    Swap(a,b);
    System.out.printf("a = %d, b = %d", a, b);

}
```



Xây dựng lớp đối tượng



□ Truyền tham số cho phương thức

- Ví dụ: Truyền tham chiếu

```
public static void Swap(MyClass a,MyClass b){
    MyClass temp = a; a = b; b = temp;
}

public static void main(String args[]){
    MyClass a = new MyClass(1);
    MyClass b = new MyClass(2);
    System.out.printf("a = %d, b = %d", a.x, b.x);
    Swap(a,b);
    System.out.printf("a = %d, b = %d", a.x, b.x);
}
```

```
class MyClass{
    public int x;
    public MyClass(int _x){
        x=_x;
    }
}
```



Xây dựng lớp đối tượng



□ Truyền tham số cho phương thức

- Ví dụ: truyền tham chiếu

```
public static void Swap(MyClass a,MyClass b){
    MyClass temp = new MyClass(a);
    a.x = b.x; b.x = temp.x;
}

public static void main(String args[]){
    MyClass a = new MyClass(1);
    MyClass b = new MyClass(2);
    System.out.printf("a = %d, b = %d", a.x, b.x);
    Swap(a,b);
    System.out.printf("a = %d, b = %d", a.x, b.x);
}
```

```
class MyClass{
    public int x;
    public MyClass(int _x){
        x=_x;
    }
}
```



Xây dựng lớp đối tượng



□ Phương thức có tham số thay đổi

- Tham số thay đổi cho phép gọi phương thức với số tham số khác nhau.
- Cú pháp

KiểuDữLiệu TênPhươngThức

(**KiểuDữLiệu ... TênThamSố**)

```
{
    // các lệnh
}
```



Xây dựng lớp đối tượng



□ Phương thức có tham số thay đổi

- Ví dụ:

```
public static void Test(int ... a) {
    for (int i : a)
        System.out.println(i);
}

public static void main(String args[]) {
    Test(1, 2, 3, 4, 5, 6);
    Test(10, 20);
}
```



Nội dung



1. Tổng quan về Đối tượng
2. Xây dựng lớp đối tượng
3. Package



Package



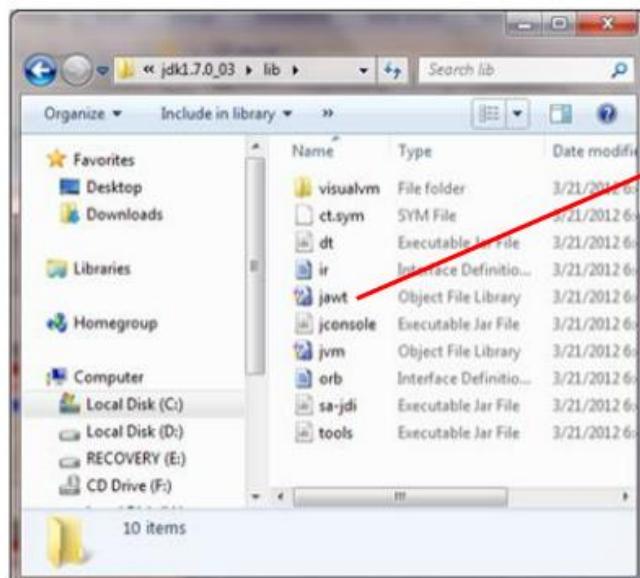
- ❑ Java package là một nhóm các lớp và giao diện có liên hệ với nhau được tổ chức thành một đơn vị để quản lý.
- ❑ Package có thể do người dùng tạo ra hoặc do Java tạo sẵn.
- ❑ Lệnh package, nếu sử dụng, phải đặt ở đầu chương trình
- ❑ Lệnh import được dùng để import một hoặc nhiều lớp từ package vào chương trình
- ❑ Chỉ định từ truy xuất điều khiển việc truy xuất các lớp và sự nhìn thấy các thành viên của lớp
- ❑ Chỉ định từ truy xuất cho biến và phương thức là các từ khóa được dùng để xác định các biến và phương thức cần được khai báo để điều khiển việc truy xuất từ người dùng



Package



☐ Package được định nghĩa trước



Bạn có thể xem nội dung của các package
Bằng WINZIP/WINRAR

```
import java.util.*;
import java.io.*;

class A
{
    .....
}
```



Package do người dùng định nghĩa



☐ Package được định nghĩa trước

- Tạo package

```
package myPackage;
class A
{
    .....
}
```

- Sử dụng package

```
import myPackage.A;
class B
{
    .....
    void method3()
    {
        A obj = new A();
        .....
    }
}
```







Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

LTV CÔNG NGHỆ JAVA

Module 1 – Bài 8: *Sử dụng dịch vụ*

Ngành LT & CSDL

www.t3h.vn



2014

5014



Nội dung



1. Giới thiệu
2. Sử dụng dịch vụ - Web Service



Dịch vụ - Web Service



□ Giới thiệu: Web service (www.phpvn.org)

- Dịch vụ Web (Web Service) được coi là một công nghệ mang đến cuộc cách mạng trong cách thức hoạt động của các dịch vụ B2B (Business to Business) và B2C (Business to Customer).
- Giá trị cơ bản của dịch vụ Web dựa trên việc cung cấp các phương thức theo chuẩn trong việc truy nhập đối với hệ thống đóng gói và hệ thống kế thừa.
- Các phần mềm được viết bởi những ngôn ngữ lập trình khác nhau và chạy trên những nền tảng khác nhau có thể sử dụng dịch vụ Web để chuyển đổi dữ liệu thông qua mạng Internet theo cách giao tiếp tương tự bên trong một máy tính.



Dịch vụ - Web Service



□ Giới thiệu: Web service (www.phpvn.org)

- Tuy nhiên, công nghệ xây dựng dịch vụ Web không nhất thiết phải là các công nghệ mới, nó có thể kết hợp với các công nghệ đã có như XML, SOAP, WSDL, UDDI...
- Với sự phát triển và lớn mạnh của Internet, dịch vụ Web thật sự là một công nghệ đáng được quan tâm để giảm chi phí và độ phức tạp trong tích hợp và phát triển hệ thống.



Dịch vụ - Web Service



□ Giới thiệu: Web service

(www.w3schools.com)

- Là các thành phần của ứng dụng có thể được xuất bản, tìm thấy và sử dụng trên trang web
- Giao tiếp bằng cách sử dụng các giao thức mở
- Khép kín và tự mô tả
- Có thể được tìm thấy bằng cách sử dụng UDDI
- Có thể được sử dụng bởi nhiều ứng dụng khác nhau
- HTTP và XML là nền tảng cho các dịch vụ Web



Dịch vụ - Web Service



❑ Ví dụ

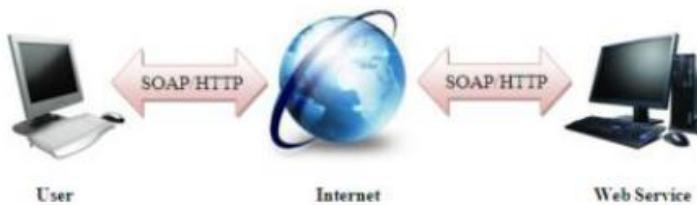
- Dịch vụ của Google: Google Weather, Google Map...
- Dịch vụ của Yahoo: YM!, Sports...
- Dịch vụ của các website khác:
<http://quotes.vinatrading.com/gia-vang.html>,
<http://dongabank.com.vn/exchange/export...>



Giới thiệu



❑ Hoạt động



Nội dung

1. Giới thiệu
2. Sử dụng dịch vụ - Web Service



Sử dụng dịch vụ

- Để sử dụng một Web Service, Client cần phải biết Web Service đó hỗ trợ những phương thức nào, phương thức cần có những tham số nào, kết quả trả về ra sao...
- Những thông tin này của một Web Service được mô tả bởi tài liệu WSDL (Web Service Description Language).



Sử dụng dịch vụ



❑ Sử dụng Web Service miễn phí

- Ví dụ: Sử dụng dịch vụ “Tỷ giá” trên Internet để xây dựng ứng dụng như sau:

Mã	Quốc kỳ	Giá mua	Giá bán
	JPY	200.7	203.0
	SGD	16430.0	16600.0
	HKD	2700.0	2730.0
	PNJ_DAB	3276000.0	3316000.0
	CAD	19100.0	19320.0
	AUD	18400.0	18600.0
	NZD	0.0	0.0

Nhập số tiền:

Loại ngoại tệ:

Thành tiền VND: Mua
Bán



Sử dụng dịch vụ



❑ Sử dụng Web Service miễn phí

- Đường dẫn:

<http://dongabank.com.vn/exchange/export>



Sử dụng dịch vụ



❑ Sử dụng Web Service miễn phí

▪ Phương thức hỗ trợ

```
private String layTyGia() throws IOException {  
    URL tyGiaUrl = new  
        URL("http://dongabank.com.vn/exchange/export");  
    URLConnection yc = tyGiaUrl.openConnection();  
    BufferedReader in = new BufferedReader(new  
        InputStreamReader(  
            yc.getInputStream()));  
    String input = in.readLine();  
    in.close();  
    return input;  
}
```

