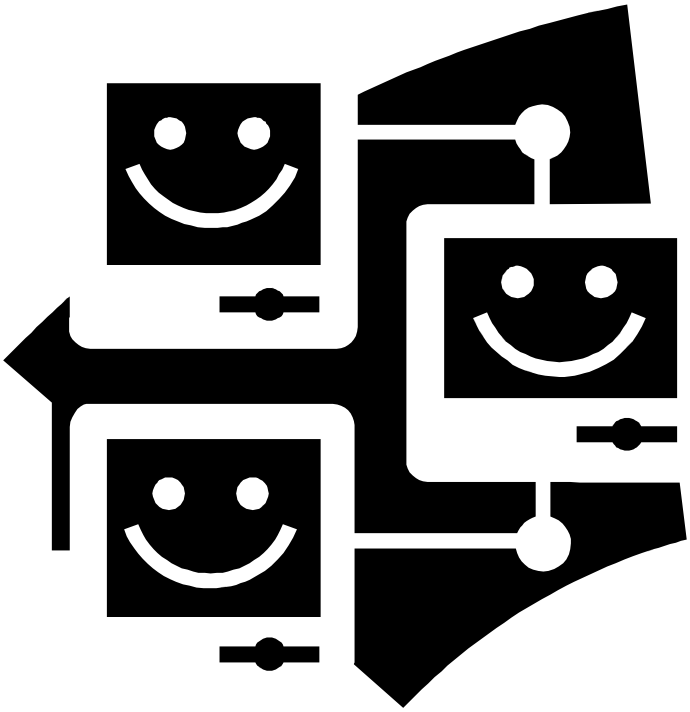
Tài liệu học tập:

Lập trình Java cơ bản



Giáo viên biên soạn: Nguyễn Tấn Thành

Năm 2007

http://www.ebook.edu.vn

Chương 1 TỔNG QUAN VỀ JAVA

Sau bài học này, học viên có thể:

- Giải thích được kiến trúc Java
- Hiểu được các công nghệ hiện có.
- Xác định được các môi trường hỗ trợ lập trình Java
- Viết mã và thi hành 1 chương trình Java đầu tay

I. LICH SỬ HÌNH THÀNH VÀ PHÁT TRIỂN JAVA:

- Java là một ngôn ngữ lập trình cấp cao theo hướng đối tượng do James Gosling và một số đồng nghiệp ở Sun Microsystems phát triển (với tên gọi ban đầu là Oak). Đây cũng là một phần trong dự án Green (các phần mềm điều khiển thiết bị điện tử dân dụng) của Sun.
- Năm 1995 Oak trở thành Java với phiên bản 1.0. Sau đó, Java không ngừng được phát triển và lần lượt các phiên bản mới được Sun phát hành. Năm 2005, Sun phát hành Java 1.5.0.

II. ĐẶC ĐIỂM NGÔN NGỮ JAVA:

Đơn giản:

- Java phát triển trên nền tảng C++, nhưng đơn giản hơn C++ rất nhiều như: không kế thừa bội, không sử dụng biến con trỏ, cấu trúc "struct" và "union" cũng được loại bỏ khỏi Java,...

Hướng đối tượng:

- Java được thiết kế xoay quanh mô hình hướng đối tượng. Vì vậy trong Java, tiêu điểm là dữ liệu và các phương pháp thao tác lên dữ liệu đó. Dữ liệu và các phương pháp mô tả trạng thái và cách ứng xử của một đối tượng trong Java.

Phân tán (Distributed):

- Java là ngôn ngữ thông dụng trong việc xây dựng các ứng dụng trên mạng nói chung và ứng dụng web nói riêng.

Trung lập kiến trúc hệ thống:

- Đây là khả năng một chương trình được viết tại một máy nhưng có thể chạy được bất kỳ đâu.

Bảo mật cao:

- Java cung cấp một số lớp để kiểm tra bảo mật và an toàn hệ thống.

Khả năng đa tuyến:

- Chương trình Java sử dụng kỹ thuật đa tiến trình (Multithread) để thực thi các công việc đồng thời. Chúng cũng cung cấp giải pháp đồng bộ giữa các tiến trình.

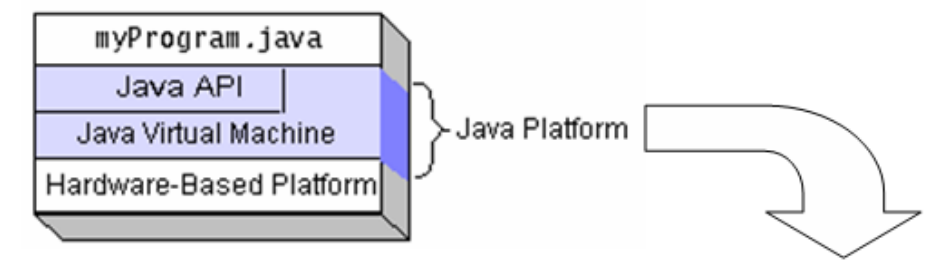
Mạnh mẻ:

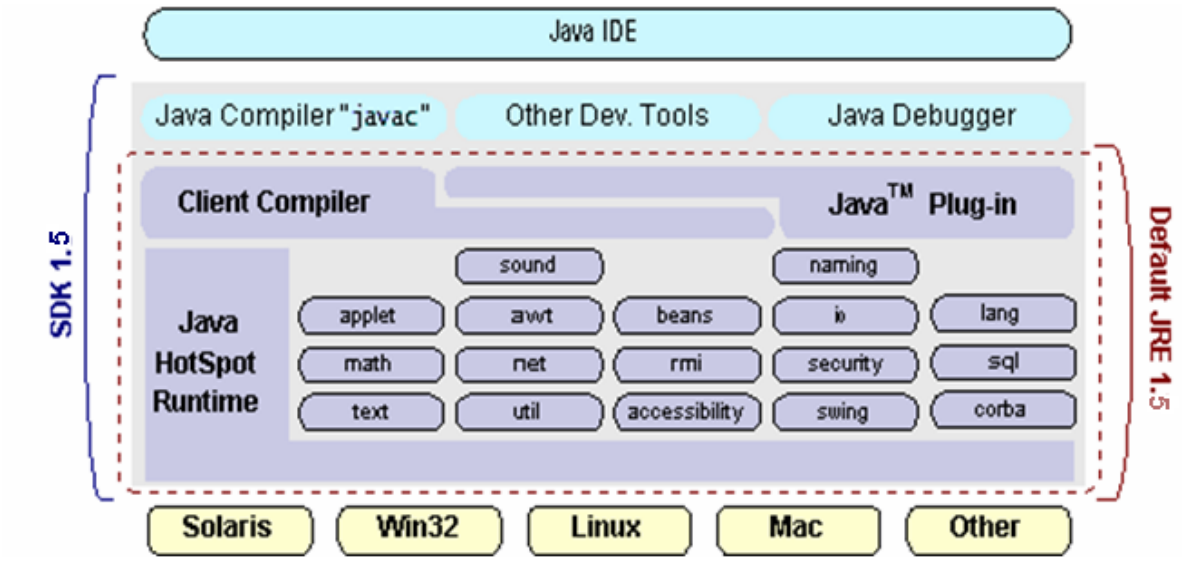
http://www.ebook.edu.vn

- Java yêu cầu chặt chẽ về kiểu dữ liệu và phải mô tả rõ ràng khi viết chương trình. Chúng sẽ kiểm tra lúc biên dịch và cả trong thời gian thông dịch vì vậy Java loại bỏ các kiểu dữ liệu dễ gây ra lỗi.

III. KIẾN TRÚC JAVA (JAVA PLATFORM) – CÁC CÔNG NGHỆ HIỆN CÓ:

1) Kiến trúc java:





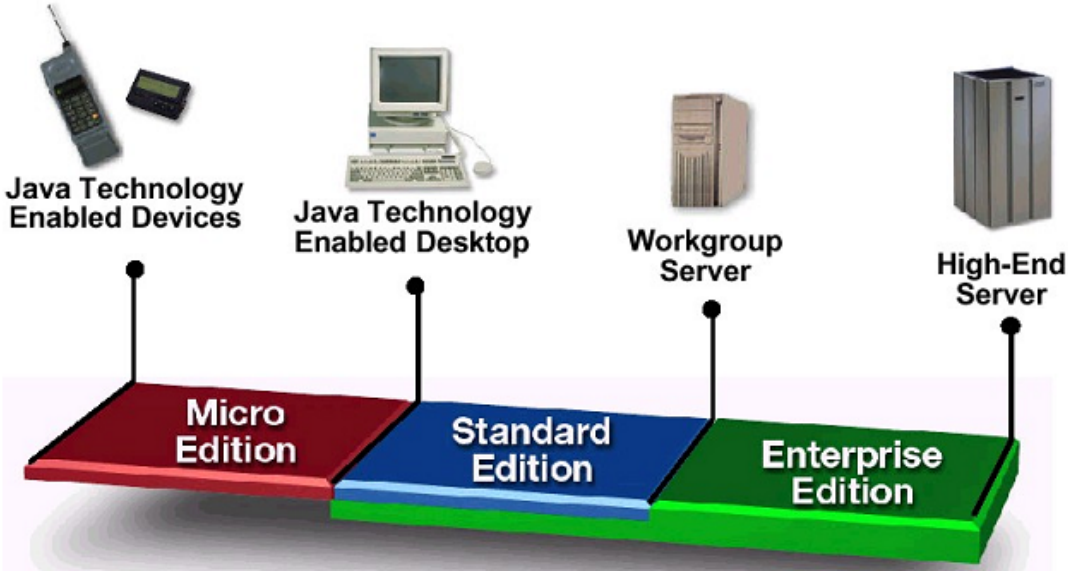
- J2SE (Java 2 Standard Edition) vừa là một đặc tả, cũng vừa là một nền tảng thực thi (bao gồm cả phát triển và triển khai) cho các ứng dụng Java. Nó cung cấp các API, các kiến trúc chuẩn, các thư viện lớp và các công cụ cốt lõi nhất để xây dựng các ứng dụng Java.
- J2SE gồm 2 bộ phận chính là:
  - *Java 2 Runtime Environment, Standard Edition (JRE)*
  - *Java 2 Software Development Kit, Standard Edition (SDK).*
- Môi trường thực thi hay JRE cung cấp các Java API, máy ảo Java và các thành phần cần thiết khác để chạy các Applet và ứng dụng viết bằng ngôn ngữ lập trình Java. Môi trường thực thi Java không có các công cụ và tiện ích như là các trình biên dịch hay các trình gỡ lỗi để phát triển các applet và các ứng dụng.
- Java 2 SDK là một tập mẹ của JRE, và chứa mọi thứ nằm trong JRE, bổ sung thêm các công cụ như là trình biên dịch và các trình gỡ lỗi cần để phát triển applet và các ứng dụng.

http://www.ebook.edu.vn

2) *Công nghệ Java:*

Hiện nay, Java có 3 công nghệ:

- ***J2SE (Java 2 Standard Edition):*** công nghệ Java chuẩn dành cho hầu hết môi trường phát triển ứng dụng Java.
- ***J2EE (Java 2 Enterprise Edition):*** công nghệ Java dành cho môi trường xí nghiệp (Enterprise) hỗ trợ kiến trúc Web, EJB, Transaction, Database...
- ***J2ME (Java 2 Micro Edition):*** công nghệ phát triển các ứng dụng cho thiết bị điều khiển như đồ gia dụng, điện thoại di động,...



IV. CÀI ĐẶT VÀ SỬ DỤNG JDK:

1) *Cài đặt:*

- Download trình biên dịch JDK 1.5.0 trên Windows tại địa chỉ:

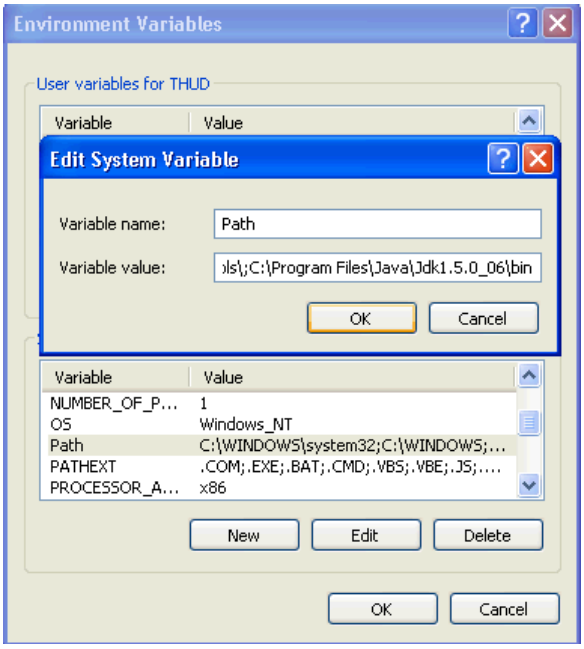
http://java.sun.com/javase/downloads/index.jsp

- Chạy chương trình cài đặt JDK. Thư mục sau cài đặt mặc định là (được gọi là JAVA_HOME):

C:\Program File\Java\Jdk1.5.0

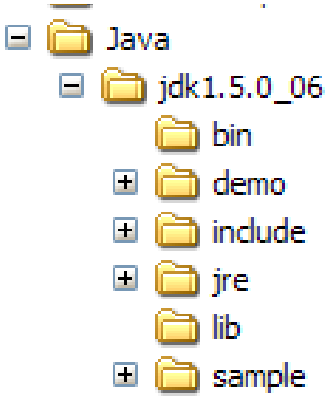
- Cập nhật biến đường dẫn trong hệ điều hành Windows 2000/XP/2003.
  - *Start\ Settings\ Control Panel\ System*
  - *Click Advanced Tab -> Click Environment Variables*
  - *Chọn biến Path trong System Variables -> Click Edit.*
  - *Cập nhật lại biến Path cho trình biên dịch JDK*

http://www.ebook.edu.vn



- Click OK.

2) *Cấu trúc thư mục của trình biên dịch JDK*



- Chứa trình biên dịch và các công cụ hỗ trợ.
- Chứa các chương trình mẫu.
- Chứa các tập tin biên dịch native code
- Môi trường thực thi ứng dụng java.
- Chứa những tập tin thư viện .jar
- Các chương trình mẫu đơn giản.

3) *Các tập tin biên dịch thường dùng:*

a) Javac:

- Dùng để biên dịch chương trình mã nguồn (.java) thành tập tin byte code (.class)

```
$javac filename.java
```

b) Java:

- Trình thông dịch java, dùng để thi hành chương trình java application

```
$java  filename
```

c) Appletviewer:

- Trình duyệt applet, dùng để thi hành chương trình java applet

```
$appletview  filename.html
```


http://www.ebook.edu.vn

d) Jdb:

- Dùng để debug chương trình Java.

V. CÁC KIỂU CHƯƠNG TRÌNH JAVA:

1) Applets:

Đây là chương trình ký sinh chạy trên Internet thông qua các trình duyệt Web hỗ trợ Java như Internet Explorer (IE) hay Netscape Navigator.

2) Ứng dụng dòng lệnh (console):

Các chương trình này chạy từ dấu nhắc lệnh và không sử dụng giao diện đồ họa. Các thông tin nhập xuất được thể hiện tại dấu nhắc lệnh.

3) Ứng dụng đồ họa (graphics):

Đây là các chương trình Java chạy độc lập cho phép người dùng tương tác qua giao diện đồ họa.

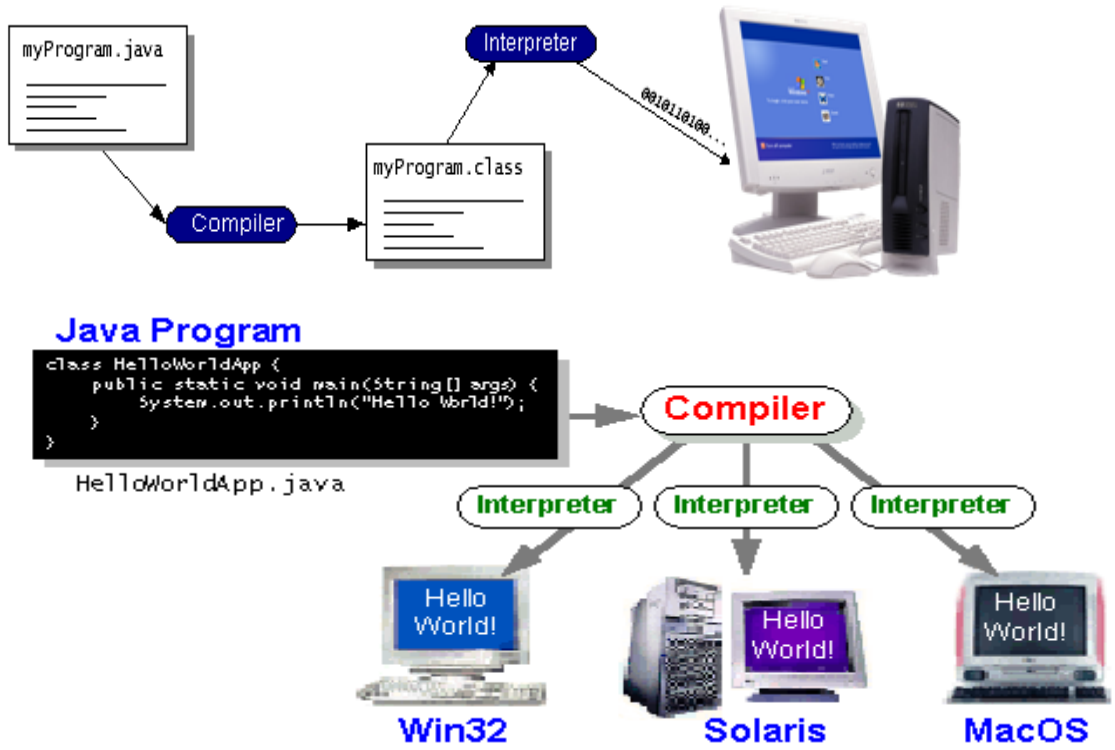
4) Servlet:

Các chương trình Java API chạy trên máy chủ, giám sát các quá trình tại máy chủ và trả lời các yêu cầu của máy trạm. Chúng có thể được dùng để xử lý dữ liệu, thực thi các transaction và thường được thực thi qua máy chủ Web.

5) Ứng dụng cơ sở dữ liệu (Database):

Các ứng dụng này sử dụng JDBC API để kết nối và lập trình với cơ sở dữ liệu.

VI. CHƯƠNG TRÌNH JAVA ĐẦU TIÊN:



http://www.ebook.edu.vn

1) Chương trình ứng dụng console:

Yêu cầu:

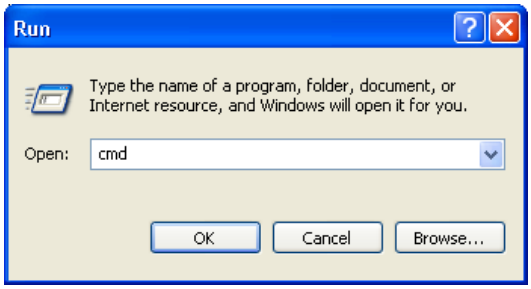
Chương trình hiển thị dòng chữ "Welcome to Java program".

Thực hiện:

- Bước 1: Tạo tập tin java nguồn Welcome.java (sử dụng chương trình NotePad)

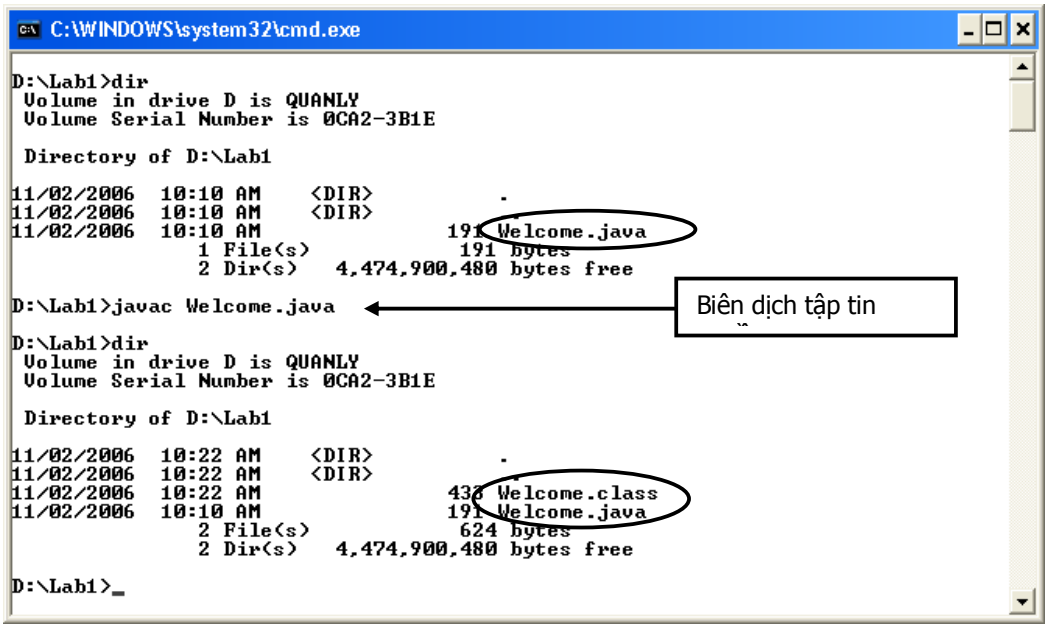
```
class Welcome{
   //main method
   public static void main(String args[]){
     //display the string
     System.out.println("\n Welcome to Java program");
   }
}
```

- Bước 2: Biên dịch tập tin nguồn.
  - *Chọn Start -> Run và nhập lệnh* ***cmd****.*



  - *Chuyển đến thư mục chứa tập tin nguồn cần biên dịch.*
  - *Dùng lệnh javac để biên dịch tập tin nguồn.*

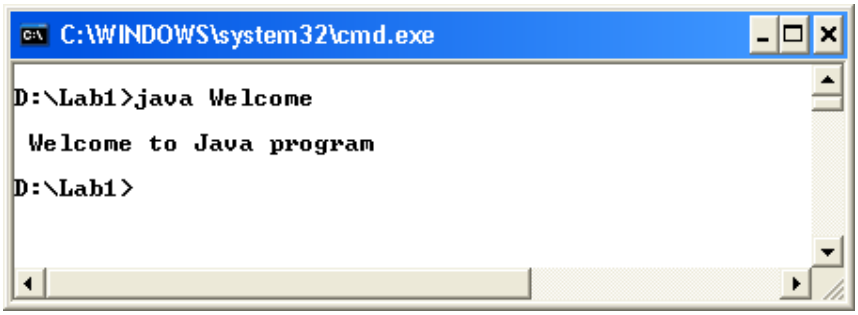
```
D:\Lab1\javac Welcome.java
```



http://www.ebook.edu.vn

- Bước 3: Chạy chương trình.
  - *Dùng lệnh java để chạy chương trình.*

```
D:\Lab1\java Welcome
```



Kết quả chương trình console

2) *Chương trình Java Applet:*

Yêu cầu:

- Chương trình vẽ chuỗi "Welcome to Java Programming" ra cửa sổ applet.

Thực hiện:

- Bước 1:
  - *Tạo tập tin java nguồn WelcomeApplet.java.*

```
import java.awt.Graphics;
import javax.swing.JApplet;
public class WelcomeApplet  extends JApplet{
   public void paint(Graphics g){
     super.paintComponents(g);
     g.drawString( "Welcome to Java Programming!",25,25);
   }
}
```

  - *Tạo tập tin WelcomeApplet.html (dùng để nhúng applet)*

```
<HTML>
<HEAD>
<TITLE>A Simple Program</TITLE>
</HEAD>
<BODY>
 Here is the output of my program:
<APPLET CODE="WelcomeApplet.class" WIDTH=300 HEIGHT=50>
</APPLET>
</BODY>
</HTML>
```

- Bước 2: Biên dịch tập tin nguồn thành tập tin .class.

```
D:\Lab1\javac WelcomeApplet.java
```


http://www.ebook.edu.vn

- Bước 3: Chạy chương trình:

```
D:\Lab1\appletviewer WelcomeApplet.html
```

Kết quả chương trình sử dụng applet

- *Ta có thể xem applet bằng trình duyệt Web.*

http://www.ebook.edu.vn

Chương 2 CẤU TRÚC LẬP TRÌNH CƠ BẢN TRONG JAVA

Sau bài học này, học viên có thể:

- Xác định được cấu trúc chung của một chương trình viết bằng Java
- Nhận dạng các kiểu dữ liệu
- Nhận dạng các toán tử
- Nhập/xuất trong java
- Nhận biết các cấu trúc lập trình cơ bản trong java

I. CẤU TRÚC CHUNG CỦA 1 CHƯƠNG TRÌNH JAVA:

1) Cấu trúc chung:

```
public class Tên_Lớp
{
  public static void main(String[] args)
{
      // các câu lệnh
  }
}
```

2) Câu chú thích (comment):

- Chú thích 1 dòng:

```
// lời chú thích
```

- Chú thích trên nhiều dòng:

```
/*  chú thích  dòng 1
chú thích  dòng 2 */
```

- Chú thích dạng tài liệu:

```
/**
document ...*/
```

II. KIỂU DỮ LIỆU:

Java là ngôn ngữ có kiểu rõ ràng, nghĩa là một biến phải có khai báo kiểu. Java cung cấp 2 loại kiểu dữ liệu:

- Các kiểu dữ liệu nguyên thủy (primitive)
- Các kiểu dữ liệu tham chiếu (reference)

Lưu ý:

- Trong C và C++, kiểu dữ liệu bị phụ thuộc vào hệ nền. Thí dụ khi thực thi chương trình trên hệ MS-DOS hay Win 3.1, kiểu int là 2 bytes, trên Windows chế độ 32 bit

http://www.ebook.edu.vn

thì kiểu int là 4 bytes. Nhưng đối với Java, kích thước của tất cả kiểu dữ liệu là độc lập hệ nền.

1) *Các kiểu dữ liệu nguyên thủy (primitive):*

a) Kiểu số (nguyên, thực):

Kiểu	Kích thước	Miền giá trị
int	4 bytes	–2.147.483.648 → 2.147.483. 647
short	2 bytes	–32.768 → 32.767
long	8 bytes	–9.223.372.036.854.775.808 → 9.223.372.036.854.775.807
byte	1 byte	–128 → 127
float	4 bytes	±3.40282347E+38F
double	8 bytes	±1.79769313486231570E+308

b) Kiểu kí tự (char):

- Kích thước 2 byte.

c) Kiểu luận lý (boolean):

- Kiểu boolean có 2 giá trị true. false

Chú ý:

- Trong java không có kiểu unsigned

2) *Các kiểu dữ liệu tham chiếu (reference):*

- Trong Java có 3 kiểu dữ liệu tham chiếu:

Kiểu dữ liệu	Miền giá trị
Mảng (Array)	Tập hợp các dữ liệu cùng loại.
Lớp (Class)	Tập hợp các biến và các phương thức.
Giao diện (Interface)	Là một lớp trừu tượng được tạo ra để bổ sung cho các kế thừa đa lớp trong Java.

III. BIẾN VÀ HẰNG TRONG JAVA:

1) *Biến:*

- Vùng nhớ lưu trữ tạm thời dữ liệu nhập vào hay tính toán để xử lý. Giá trị của biến có thể thay đổi.

Khai báo 1 biến:

```
<kiểu dữ liệu> <tên_biến>;
```

Khai báo nhiều biến:

```
<kiểu dữ liệu> <biến 1>, <biến 2>,...;
```


http://www.ebook.edu.vn

Khai báo biến:

```
<kiểu dữ liệu> <tên_biến>;
```

Ví dụ:

```
int  x;
float  a. b=6;
```

2) Hằng:

- Vùng nhớ lưu trữ tạm thời dữ liệu để xử lý. Hằng có giá trị không đổi.

Khai báo hằng:

```
final <kiểu dữ liệu> <tên hằng>=<giá trị>;
```

Ví dụ:

```
final  int MAX=100; //khai báo hằng MAX có giá trị 100.
```

IV. CÁC PHÉP TOÁN TRONG JAVA:

1) Phép toán số học:

Phép toán số học	Ý nghĩa	Phép toán so sánh	Ý nghĩa
+	Phép cộng	>	Lớn hơn
-	Phép trừ	>=	Lớn hơn hay bằng
*	Phép nhân	<	Nhỏ hơn
/	Phép chia (nguyên,thực)	<=	Nhỏ hơn hay bằng
%	Phép lấy số dư (nguyên)	==	Bằng
++	Tăng một đơn vị	!=	Khác
--	Giảm một đơn vị	!	Phủ định

Phép toán luận lý	Ý nghĩa	Phép toán khác	Ý nghĩa
&&	Và	=	Gán
||	Hay	(;)	Thay đổi độ ưu tiên
?	Điều kiện	[;]	Truy xuất phần tử của mảng

Đặc biệt: phép toán + còn dùng với ý nghĩa nối chuỗi.

- Number + String -> String
- Boolean + String -> String

Ví dụ:

```
System.out.println(" Hello " + "World!");
// Kết quả in trên màn hình:
Hello World!
```


http://www.ebook.edu.vn

Ví dụ:

```
int x=5;
System.out.println("Value x =" + x);
// Kết quả in trên màn hình:
Value x=5
```

2) *Chuyển kiểu trong java:*

- Chuyển số nguyên sang số thực và ngược lại:

Ví dụ:

```
int n=5;
double x;
x=n;
System.out.println(d);
//kết quả in trên màn hình: 5.0
```

Ví dụ:

```
double  x=15.7;
int n;
n=(int) x;
System.out.println(n);
// Kết quả in ra màn hình: 15
```

- Chuyển dữ liệu chuỗi sang số: sử dụng các phương thức của các lớp bao kiểu số (Integer. Long. Float. Double...)

```
Integer.parseInt(chuỗi dạng số nguyên)
Float.parseFloat(chuỗi dạng số thực)
Double.parseDouble(chuỗi dạng số thực)
```

Ví dụ:

```
Int n;
n= Integer.parseInt("12");
// n có giá trị là 12
```

Ví dụ:

```
double x;
x= Double.parseDouble("12.25");
// n có giá trị là 12.25
```


http://www.ebook.edu.vn

V. NHẬP XUẤT TRONG JAVA:

Ta dễ dàng xuất một giá trị ra thiết bị chuẩn bằng cách gọi lệnh System.out.println ().

- Trước JDK1.5. không có phương pháp thuận lợi để đọc 1 giá trị từ console window.
- Trong JDK1.5. java cung cấp luồng nhập Scanner để đọc 1 giá trị bất kỳ từ console window.

1) Lớp Scanner:

Dùng để đọc giá trị từ một console. Các bước đọc dữ liệu từ console:

- Tạo luồn nhập chuẩn.
- Sử dụng các phương thức đọc dữ liệu tương ứng.

a) Tạo luồng nhập chuẩn:

Cú pháp:

```
Scanner in = new Scanner(System.in);
```

b) Các phương thức đọc dữ liệu:

```
in.nextLine()       đọc dòng dữ liệu
in.nextInt()        đọc một số nguyên
in.nextDouble()     đọc một số thực
```

Ví dụ:

```
import java.util.*;
  public class InputTest{
    public static void main(String[] args){
      Scanner in = new Scanner(System.in);
      System.out.print("What is your name? ");
      String name = in.nextLine();
      System.out.print("How old are you? ");
      int age = in.nextInt();
      // display output on console
     System.out.println("Hello. "+ name + ". Next year.
     you'll be " +(age + 1));
    }
  }
```


http://www.ebook.edu.vn

Chú ý:

- Nếu sử dụng phiên bản trước JDK1.5.0. ta sẽ rất khó khăn để đọc giá trị từ người dùng ở chế độ console. Phương pháp đơn giản là nên sử dụng hộp thoại (JOptionPane). Với lệnh như sau:

```
String input = JOptionPane.showInputDialog("Thông báo");
```

- Kết quả nhận được là kiểu chuỗi. Để chuyển dữ liệu sang kiểu số ta dùng phương thức chuyển kiểu: **Integer.parseInt** hoặc **Double.parseDouble**

Ví dụ:

```
import javax.swing.*;
  public class InputTest{
    public static void main(String[] args){
      String name = JOptionPane.showInputDialog("What is
your name?");
      String input = JOptionPane.showInputDialog("How old
are you?");
      int age = Integer.parseInt(input);
      JOptionPane.showMessageDialod(null."Hello. " + name
+ ". Next year. you'll be " +(age +   1));
      System.exit(0);
    }
  }
```

VI. CẤU TRÚC ĐIỀU KHIỂN:

1) Ý nghĩa:

Dùng làm thay đổi trật tự thi hành lệnh của chương trình.

Các cấu trúc gồm:

- Câu lệnh điều kiện: if. switch.
- Câu lệnh lặp: for. while. do-while.
- Câu lệnh: break. continue. return.

2) Lệnh và khối lệnh:

a) Lệnh:

- Là một chỉ thị được kết thúc bằng dấu chấm phẩy ";"

http://www.ebook.edu.vn

b) Khối lệnh:

- Là tập hợp nhiều lệnh nằm trong một cặp móc "{" và "}"

Ví dụ:

```
{
   int  x =5;  // lệnh 1
System.out.println("Giá trị x=" +x); // lệnh 2
}
```

3) Câu lệnh điều kiện:

a) Câu lệnh if ..else:

Dạng 1 (if thiếu):

```
if(<biểu thức điều kiện>)
{    <lệnh 1>
     <lệnh 2>
}
```

Dạng 2 (if thiếu):

```
if(<biểu thức điều kiện>)
{    <lệnh 1-1>
     <lệnh 1-2>
}
else
{    <lệnh 2-1>
     <lệnh 2-2>
}
```

Dạng 3 (if mở rộng):

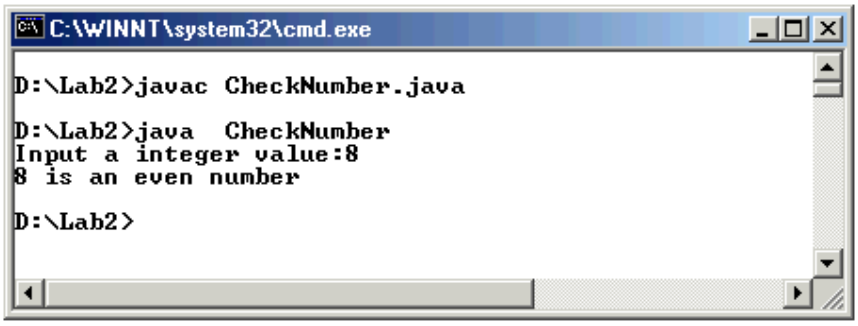
```
if(<biểu thức điều kiện 1>)
{    <lệnh 1-1>
     <lệnh 1-2>
}
elseif(<biểu thức điều kiện 2>)
{    <lệnh 2-1>
     <lệnh 2-2>
}
else
{    <lệnh 3-1>
     <lệnh 3-2>
...
}
```


http://www.ebook.edu.vn

Ví dụ:

- Chương trình sau kiểm tra xem số nguyên được nhập vào từ bàn phím là số chẵn hay lẻ và hiển thị thông báo phù hợp.

```
import java.util.Scanner;
  public class CheckNumber{
    public static void main(String args[]){
      Scanner in = new Scanner(System.in);
      System.out.print("Input a integer value:");
      int num = in.nextInt();
      if(num %2 == 0)
        System.out.println(num + " is an even number");
      else
        System.out.println(num + " is an odd number");
    }
  }
```



Kết quả của chương trình CheckNumber

b) Câu lệnh switch:

Cú pháp:

```
switch(<biểu thức>)
{
  case <giá trị 1>:  lệnh 1;
                     break;
  case <giá trị 2>:  lệnh 2;
                     break;
  default:           lệnh n;
```

Lưu ý:

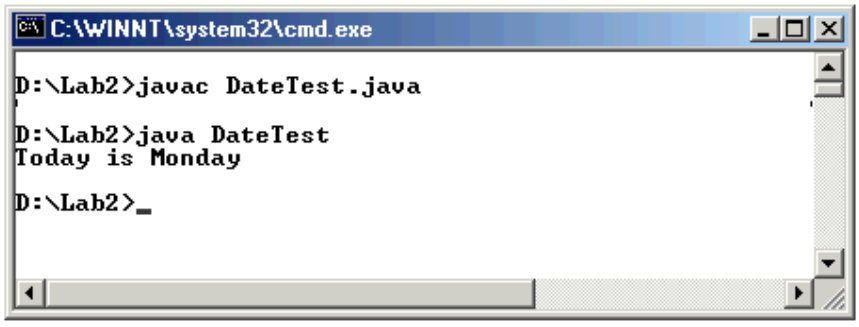
- Biểu thức trong câu lệnh switch phải có giá trị kiểu nguyên (byte,char)

Ví dụ:

- Chương trình sau kiểm tra ngày hiện tại là thứ mấy trong tuần.

http://www.ebook.edu.vn

```
import java.util.Date;
class DateTest{
  public static void main(String agrs[]){
    Date today = new Date();
    int day = today.getDay();
    switch(day){
      case 0: System.out.println("Today is Sunday");
              break;
      case 1: System.out.println("Today is Monday");
              break;
      case 2: System.out.println("Today is Tuesday");
              break;
      case 3: System.out.println("Today is Wednesday");
              break;
      case 4: System.out.println("Today is Thursday");
              break;
      case 5: System.out.println("Today is Friday");
              break;
      case 6: System.out.println("Today is Satuday");
              break;
      default: System.out.println("Invalid day of week");
    }
  }
}
```



Kết quả của chương trình DateTest

http://www.ebook.edu.vn

4) Cấu trúc vòng lặp:

a) Vòng lặp while:

Cú pháp:

```
while(<biểu thức điều kiện>){
    <lệnh 1>;
    <lệnh 2>;
}
```

b) Vòng lặp do..while:

Cú pháp:

```
do{
    <lệnh 1>;
    <lệnh 2>;
}while(<biểu thức điều kiện>);
```

c) Vòng lặp for:

Cú pháp:

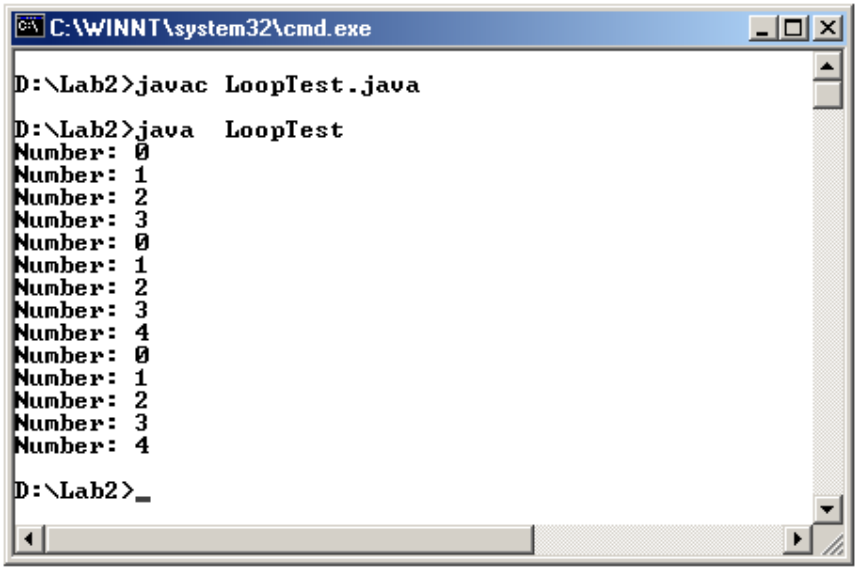
```
for(<khởi tạo>;<Bthức điều kiện>;<Bthức thay đổi>){
   <lệnh 1>;
   <lệnh 2>;
}
```

Ví dụ:

```
public class LoopTest{
  public static void main(String args[]){
    listNums1(3);
    listNums2(4);
    listNums3(5);
  }
 // while loops
  public static void listNums1(int max){
    int i=0;
    while(i<=max){
      System.out.println("Number: " + i);
      i++;   //i=i+1;
    }
  }
 // do loops
```


http://www.ebook.edu.vn

```
  public static void listNums2(int max){
    int i=0;
    do{
      System.out.println("Number: " + i);
      i++;
    }while(i<=max);
  }
 // for loops
  public static void listNums3(int max){
    for(int i=0;i<max;i++){
      System.out.println("Number: " + i);
    }
   }
}
```



Kết quả chương trình LoopTest.java

5) *Câu lệnh break, continue và return:*

a) Lệnh break:

- Khi gặp lệnh break, chuong trình sẽ di chuyển điều khiển ra khỏi khối lệnh.

http://www.ebook.edu.vn

Ví dụ:

```
int i = 1;
while(true){
  if(i == 3)
    break;
  System.out.println(i);
  i++;
}
<Lệnh k>
```

- Khi i = 3 thì chương trình thoát khỏi vòng lặp và thực hiện lệnh k.

b) Lệnh continue:

- Di chuyển điều khiển sang lần lặp kế tiếp và không thực hiện lệnh đứng sau nó.

Ví dụ:

```
for(i=0; i<5; i++){
  if(i%2 == 0)
    continue;
  System.out.println(i);
}
```

- Khi i = số chẵn thì chương trình sẽ không thực hiện lệnh in i mà thực hiện tiếp vòng lặp kế.

c) Lệnh return:

- Chấm dứt sự thực thi của một phương thức và trả về giá trị cho lời gọi phương thức.

Cú pháp:

```
return;
Hoặc
return <giá trị>;
```

VII. MẢNG (Array):

Mảng là một cấu trúc dữ liệu mà ở đó nó lưu trữ tập hợp các phần tử có giá trị cùng kiểu. Ta truy xuất 1 phần tử của mảng thông qua tên mảng và chỉ số (index) của nó.

1) Mảng 1 chiều:

Khai báo:

```
<kiểu dữ liệu> <tên mảng>[];
```


http://www.ebook.edu.vn

```
Int a[];
Int b[]=new int[10];   // vừa khai báo vừa cấp phát
int c[] ={2,3,5,7,11,13}; // vừa khai báo và khởi tạo
```

- Chỉ số phần tử của mảng xuất phát từ 0 → kích_thước - 1
- Để biết số phần tử của mảng, dùng phương thức **length**

Ví dụ:

```
for(int i = 0; i < a.length; i++)
  System.out.println(a[i]);
```

- Trong JDK1.5. ta có thể dùng vòng lặp **for each** để duyệt tập hợp (mảng).

Cú pháp:

```
for(<biến>:<tập_hợp>){
   <lệnh 1>;
   <lệnh 2>;
}
```

Ví dụ:

```
for(int x:a)
  System.out.println(x);
```

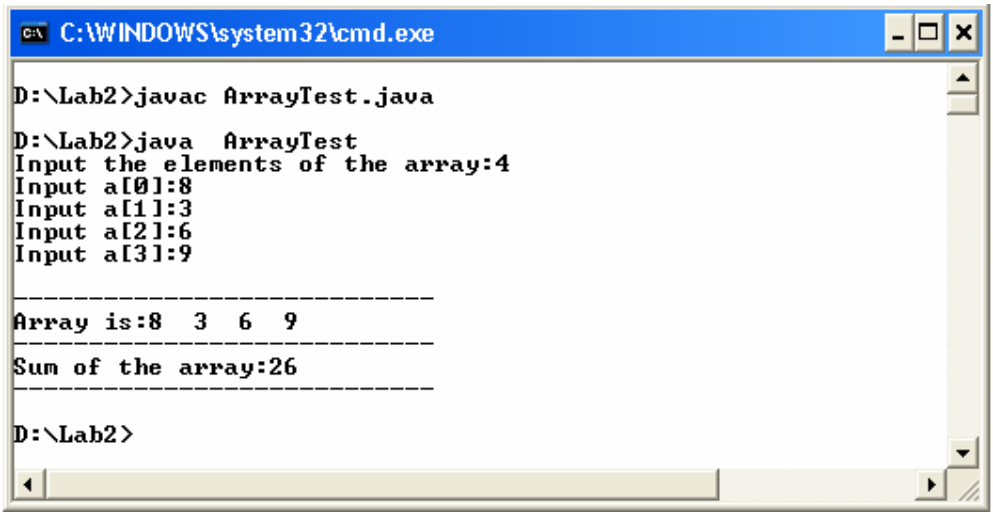
Ví dụ:

- Nhập mảng số nguyên có n phần tử bằng console windows. Tính tổng mảng vừa nhập.

```
import java.util.Scanner;
public class ArrayTest{
  public static void inputArray(int a[], Scanner in){
    for(int i=0;i<a.length;i++){
      System.out.print("Input a["+i+ "]:");
      a[i]=in.nextInt();
    }
  }
  public static void outputArray(int a[]){
    for(int i=0;i<a.length;i++){
      System.out.print(a[i] + "  ");
    }
  }
  public static int sumArray(int a[]){
    int s=0;
    for(int i=0;i<a.length;i++){
```


http://www.ebook.edu.vn

```
        s+=a[i];
      }
      return s;
    }
    public static void main(String args[]){
      int a[],n;
      Scanner in = new Scanner(System.in);
      System.out.print("Input the elements of the array:");
      n= in.nextInt();
      a= new int[n];
      inputArray(a.in);
      System.out.println("\n----------------------------");
      System.out.print("Array is:");
      outputArray(a);
      System.out.println("\n----------------------------");
      System.out.println("Sum of the array:" +sumArray(a));
      System.out.println("----------------------------");
    }
}
```



Kết quả chương trình ArrayTest.java

2) Mảng 2 chiều:

Khai báo:

```
<kiểu dữ liệu> <tên mảng>[][];
```


http://www.ebook.edu.vn

Cấp phát vùng nhớ:

```
<tên mảng> = new <kiểu dữ liệu>[số dòng][số cột];
```

Ví dụ:

```
int a[][];
a= new int [3][3];
int b[][]= new int [3][2]; // khai báo và cấp phát
int  array[] ={{16, 3, 2, 13},
               {5, 10, 11, 8},
               {9, 6, 7, 12}};
```

- Truy xuất phần tử dòng i. cột j của mảng: b[i][j]

Duyệt mảng 2 chiều:

Ví dụ:

```
for(int row = 0;row < array.length;row++){
  for(int col = 0;col < array[row].length;col++){
    System.out.println("  " + array[row][col]);
  }
  System.out.println("\n");
}
```

- Dùng vòng lặp "for each" để duyệt mảng 2 chiều:

```
for(int[] row:array){
  for(int  x:row){
    <lệnh 1>;
    <lệnh 2>;
  }
}
```

VIII. GÓI (package):

1) Khái niệm:

Gói là thư mục chứa một hay nhiều tập tin .class sau khi biên dịch từ tập tin mã nguồn (.java) nhằm mục đích phân phối cho người dùng khác sử dụng lại. Gói có tính phân cấp (gói lồng gói).

Ưu điểm của việc tạo gói:

- Cho phép tổ chức các lớp vào những đơn vị nhỏ hơn
- Giúp tránh được tình trạng trùng lặp khi đặt tên.
- Cho phép bảo vệ các lớp đối tượng
- Tên gói (Package) có thể được dùng để nhận dạng chức năng của các lớp.

http://www.ebook.edu.vn

Một số gói chuẩn của java:

- java.lang
- java.applet
- java.awt
- javax.swing
- java.io
- java.util
- java.net
- java.awt.event
- java.rmi
- java.security
- java.sql

2) *Tạo gói:*

Để đặt một lớp vào gói. đặt lệnh tạo gói vào dòng đầu của tập tin mã nguồn

Cú pháp:

```
package <tên_gói>;
```

Ví dụ:

```
package my.com;
public class Common{
  public static boolean evenCheck(int x){
    if(x%2==0)
      return true;
    return false;
  }
}
```

3) *Sử dụng lại lớp trong gói:*

Một chương trình .java muốn sử dụng một lớp đã cài đặt. ta phải nhập lớp này ở đầu chương trình (giống như include tập tin .h trong C).

Cú pháp:

```
import <tên_gói>.<tên_lớp>;
import <tên_gói>.*;
```

Ví dụ:

```
import my.com.Common;
  public class CommonTest{
    public static void main(String args[]){
```


http://www.ebook.edu.vn

```
        int x=4;
        System.out.println("Even x test:" +
                           Common.evenCheck(x));
    }
}
```

Lưu ý:

- Dòng lệnh khai báo tạo gói phải là dòng đầu tiên trong tập tin khai báo lớp.
- Các tập tin khai báo lớp trong cùng gói phải được lưu trong cùng một thư mục.
- Không nên tham khảo toàn bộ một gói vì sẽ làm tốn bộ nhớ trong khi chỉ truy cập đến một vài lớp trong gói.

http://www.ebook.edu.vn

Chương 3 LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Sau bài học này, học viên có thể:

- Giải thích các đặc điểm của lập trình hướng đối tượng
- Hiểu được mẫu chung của lớp
- Hiểu được phương thức khởi tạo (Constructor)
- Giải thích khái niệm Overloading phương thức và Overriding phương thức
- Hiểu được từ khóa this, super
- Phân biệt được dữ liệu và phương thức thể hiện với dữ liệu và phương thức lớp (static)

I. KHÁI NIỆM:

1) Khái niệm lập trình hướng đối tượng

- Lập trình hướng Đối tượng (OOP) là một phương pháp thiết kế và phát triển phần mềm. Những ngôn ngữ OOP không chỉ bao gồm cú pháp và một trình biên dịch (compiler) mà còn có một môi trường phát triển toàn diện. Môi trường này bao gồm một thư viện được thiết kế tốt, thuận lợi cho việc sử dụng các đối tượng.
- Trong OOP, mỗi vấn đề được chia ra thành nhiều yếu tố, được gọi là các Đối tượng (Objects) hoặc các Thực thể (Entities).
- Chúng ta sử dụng kỹ thuật hướng đối tượng để ánh xạ những thực thể chúng ta gặp phải trong đời sống thực thành những thực thể tương tự trong máy tính.

2) Đặc điểm của lập trình hướng đối tượng:

Sự đóng gói (Encapsulation): Là việc gom chung dữ liệu và các thao tác liên quan đến dữ liệu thành một thể thống nhất, tránh được các thao tác không hợp lệ từ bên ngoài.

Tính kế thừa (Inheritance): Một lớp mới có thể xây dựng từ một lớp cũ thông qua sự kế thừa. Sự kế thừa có thể là kế thừa dữ liệu và các phương thức.

Tính đa hình (Polymorphism): Tính đa hình cho phép mô tả những phương thức có tên giống nhau trong cùng một lớp hoặc trong các lớp khác nhau. Nói cách khác, một thao tác có thể được cài đặt khác nhau trong cùng một lớp hay trong những lớp khác nhau.

II. ĐỐI TƯỢNG – LỚP:

1) Đối tượng (object):

a) Đối tượng trong thế giới thực:

Bao gồm 2 thành phần:

- Các thuộc tính.
- Các hành động.

Ví dụ: Xét đối tượng hình tròn trong mặt phẳng, có

- Thuộc tính: bán kính,..
- Hành động: tính chu vi, tính diện tích,…

http://www.ebook.edu.vn

b) Đối tượng dưới góc độ lập trình:

Đối tượng = thuộc tính + phương thức

2) *Lớp (Class):*

Là cấu trúc mô tả cho các đối tượng cùng các tính chất đặc trưng. Lớp phải chỉ ra thành phần thuộc tính và các phương thức của đối tượng thuộc lớp.

a) Định nghĩa lớp:

Cú pháp:

```
[modifier] class ClassName{
    modifier data-type field1;
    modifier data-type field2;
    ...
    modifier data-type fieldN;

    //Constructor
    modifier ClassName(cparams1){
      // body of constructor
    }
    ...
    modifier ClassName( cparams2){
      // body of constructor
    }
    //Method
    modifier return-type  methodName1(parameters){
      // body of method
    }
    ...
    modifier return-type methodName2(parameters){
      // body of method
    }
}
```

Các thuộc tính lớp

Các phương thức lớp

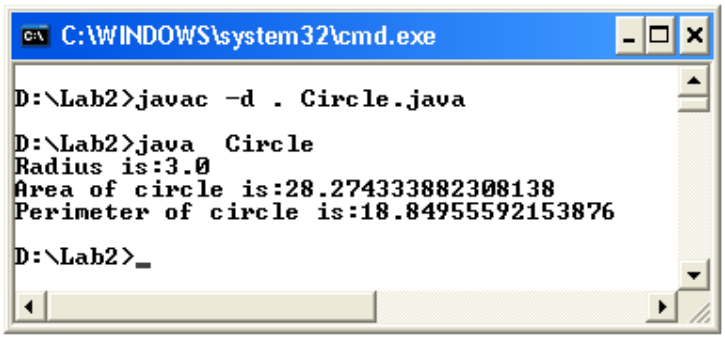
Ví dụ: Định nghĩa lớp Circle (hình tròn)

```
package shape;
public class Circle{
  // khai báo thành phần thuộc tính
  private double radius;
  // định nghĩa Constructor
  public Circle(){
    radius=1.0;
  }
  public Circle(double r){
```


http://www.ebook.edu.vn

```
    radius=r;
  }
  //định nghĩa phương thức
  //cập nhật giá trị cho thuộc tính radius
  public void setRadius(double r){
    radius=r;
  }
  // đọc giá trị của thuộc tính radius
  public double getRadius(){
    return radius;
  }
  // tính diện tích
  public double area(){
    return radius*radius*Math.PI;
  }
  // tính chu vi
  public double perimeter(){
    return 2*radius*Math.PI;
  }
  // phương thức chính
  public static void main(String args[]){
    Circle myCircle = new Circle();
    myCircle.setRadius(3.0);
    System.out.println("Radius is:" +
                       myCircle.getRadius());
    System.out.println("Area of circle is:" +
                       myCircle.area());
    System.out.println("Perimeter of circle is:" +
                       myCircle.perimeter());
   }
}
```


http://www.ebook.edu.vn



Kết quả của Circle.java

3) Khai báo và sử dụng đối tượng:

Một khi lớp đã được định nghĩa, ta có thể dễ dàng khai báo biến (đối tượng) của lớp theo cú pháp sau:

```
<ClassName <objectName>;
```

Ví dụ:

```
Color blue;
Point start;
```

- Giá trị mà đối tượng tham chiếu ban đầu là null.
- Trước khi sử dụng đối tượng, cần phải khởi tạo đối tượng bằng toán tử new

Cú pháp:

```
objectName = new ClassName();
```

Ví dụ:

```
start = new Point(2,3);
```

Truy xuất thành phần của đối tượng:

- Truy xuất thành phần dữ liệu:

```
objectName.fieldName
```

Ví dụ: Lớp Point trong java có 2 thuộc tính x và y. Ta có thể truy xuất như sau:

```
Point p = new Point(2,3);
int xSquared = p.x * p.x; // xSquared có giá trị 4
int xPlusY = p.x + p.y; // xPlusY có giá trị 5
p.x = 7;
xSquared = p.x * p.x; // xSquared có giá trị 49
```

- Truy xuất (gọi) thành phần phương thức:

```
variableName.methodName(param);
```

Ví dụ:

```
String s = new String("Hello Word");
int len = s.length(); // len có giá trị 10
```


http://www.ebook.edu.vn

```
int  index = s.indexOf("Wo"); // index có giá trị 6
```

4) *Phương thức:*

a) Phương thức:

Phương thức là những ứng xử của lớp lên các thành phần dữ liệu hay lên các tác động bên ngoài. Phương thức được định nghĩa theo cấu trúc sau:

```
[modifier] <kiểu_trả_về> <tên_phương_thức>([các tham
số]){
  // thân của phương thức
}
```

Một phương thức có các tính chất bổ từ (modifiers) như:

- public: có thể truy cập từ bên ngoài lớp định nghĩa
- protected: chỉ được truy cập từ lớp định nghĩa và những lớp dẫn xuất của lớp đó.
- private: chỉ được truy cập trong bản thân lớp định nghĩa.
- static: một phương thức của lớp, chung cho mọi phiên bản của lớp.
- abstract: không được cài đặt gì trong lớp.
- final: không bị định nghĩa chồng (overriding) ở các lớp dẫn xuất .
- synchronized: dùng để chỉ một phương thức tới hạn, ngăn các tác động của đối tượng khác lên đối tượng trong khi việc đồng bộ hóa đang thực hiện.

b) Constructor:

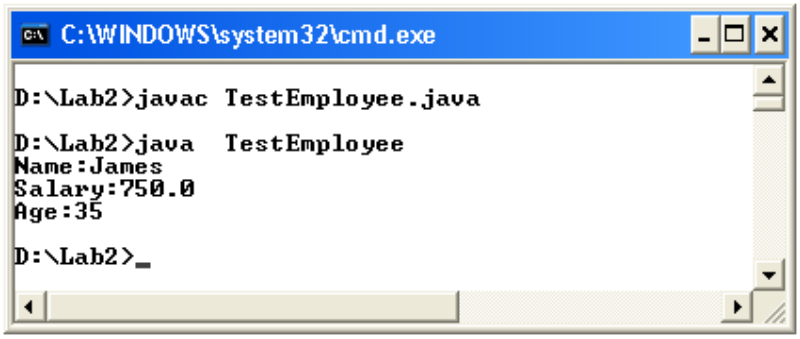
- Còn gọi là phương thức khởi tạo.
- Constructor là một phương thức đặc biệt được gọi tự động khi chương trình tạo một đối tượng mới với toán tử new.
- Phương thức Constructor phải có cùng tên với lớp và không khai báo kiểu trả về.
- Phương thức Constructor gán các giá trị ban đầu cho thuộc tính của đối tượng và các công việc cần thiết trước khi đưa đối tượng vào hoạt động.

Ví dụ: Định nghĩa lớp Employee

```
class Employee{
  // instance fields
  private String name;
  private double salary;
  private int age;
  // constructor
  public Employee(String n, double s, int y){
    name = n;
    salary = s;
    age = y;
  }
  // method
```


http://www.ebook.edu.vn

```
  public String getName(){  return name; }
  public double getSalary(){  return salary; }
  public void raiseSalary(double byPercent){
    double raise = salary * byPercent / 100;
    salary += raise;
  }
  public String toString(){
    return "Name:" + name + "\nSalary:" + salary +
"\nAge:" + age;
  }
} // end class Employee

public class TestEmployee{
  public static void main(String args[]){
    Employee  emp = new Employee("James",750,35);
    System.out.println(emp.toString());
  }
}
```



Kết quả của TestEmployee.java

Chú ý:

- Nếu một lớp không có constructor, Java sẽ gán cho nó một constructor mặc định. Đó là khởi tạo tất cả thành phần dữ liệu của lớp theo các giá trị mặc định của kiểu dữ liệu tương ứng. Tuy nhiên điều đó rất nguy hiểm và thiếu thận trọng. Do đó ta luôn phải xác định một constructor cho mỗi lớp đã tạo.

c) Từ khóa this:

- Mọi lớp trong java luôn tồn tại một biến ẩn this.
- Thành phần này được sử dụng trong khi chạy và tham khảo đến bản thân lớp chứa nó.

http://www.ebook.edu.vn

Ví dụ:

```
// định nghĩa constructor cho lớp Employee
public Employee(String name, double salary, int age){
  this.name = name;
  this.salary = salary;
  this.age = age;
}
```

Trong ví dụ trên do các tên biến tham số cùng tên với các dữ liệu của lớp để dễ hiểu khi sử dụng. Trong trường hợp này, ta phân biệt được dữ liệu của lớp nhờ dùng biến this.

5) Khai báo chồng phương thức (overloaded method):

- Khai báo chồng phương thức là ta tạo ra nhiều phương thức có cùng tên, các phương thức chỉ khác nhau ở các tham số.
- Do đó khi gọi một phương thức có khai báo chồng, ta phải xác định đúng tham số cần gởi theo để phương thức tương ứng thực hiện.

Ví dụ:

- Định nghĩa lớp Point3D

```
class Point3D{
  double x;
  double y;
  double z;
  Point3D(double x){
    this(x,0,0);
  }
  Point3D(double x, double y){
    this(x,y,0);
  }
  Point3D(double x, double y, double z){
    this.x = x;
    this.y = y;
    this.z = z;
  }
```

Các phương thức **Contructor** chồng nhau.

http://www.ebook.edu.vn

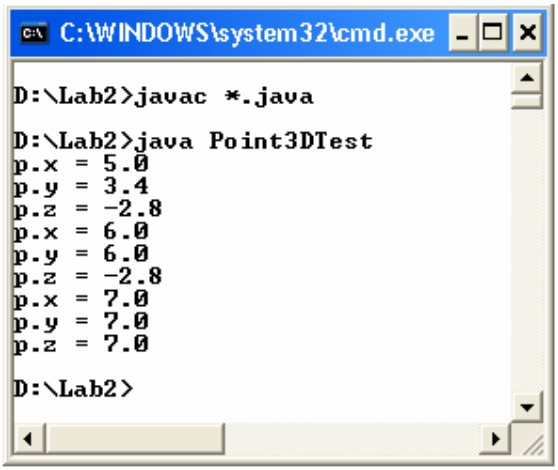
```
void move(double x, double y, double z){
    this.x = x;
    this.y = y;
    this.z = z;
  }
  void move(double x, double y){
    this.x = x;
    this.y = y;
  }
  void move(double x){
    this.x = x;
  }
}
```

Các phương thức **move** chồng nhau.

- Sử dụng lớp Point3D

```
class Point3DTest{
  public static void main(String args[]){
    Point3D p = new Point3D(1.1, 3.4, -2.8);
    p.move(5);
    System.out.println("p.x = " + p.x);
    System.out.println("p.y = " + p.y);
    System.out.println("p.z = " + p.z);
    p.move(6,6);
    System.out.println("p.x = " + p.x);
    System.out.println("p.y = " + p.y);
    System.out.println("p.z = " + p.z);
    p.move(7,7,7);
    System.out.println("p.x = " + p.x);
    System.out.println("p.y = " + p.y);
    System.out.println("p.z = " + p.z);
  }
}
```


http://www.ebook.edu.vn



Kết quả chương trình Point3DTest

6) *Dữ liệu lớp và phương thức lớp (static):*

a) Dữ liệu lớp:

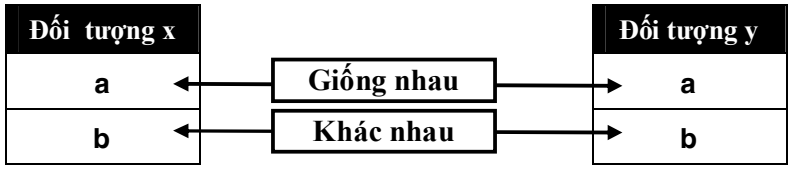
- Là dữ liệu dùng chung (toàn cục) cho tất cả các đối tượng cùng kiểu lớp.

Khai báo:

```
static <kiểu> <tênbiến>;
```

Ví dụ:

```
class Aclass{
  static int a;
  int b;
  ...
}
  Aclass x = new Aclass();
  Aclass y = new Aclass();
```



- Thực hiện gán giá trị:

```
x.a = 10;
x.b = 20;
y.a = 30;
y.b = 40;
```

- Kết thúc 4 câu lệnh trên ta có x.a = 30 , x.b = 20

http://www.ebook.edu.vn

b) Phương thức lớp:

- Là phương thức đặc trưng không cần truy cập dữ liệu phiên bản (non-static) của lớp.
- Phương thức lớp thường tương tự như hàm toàn cục trong các ngôn ngữ khác.
- Phương thức lớp chỉ truy xuất dữ liệu toàn cục (static).

Cú pháp:

```
public static <tên phương thức>([các tham số]){
  // thân của phương thức
}
```

- Để truy xuất phương thức lớp của lớp, ta dùng cú pháp sau:

```
<ClassName>.<methodName>([params);
```

Ví dụ:

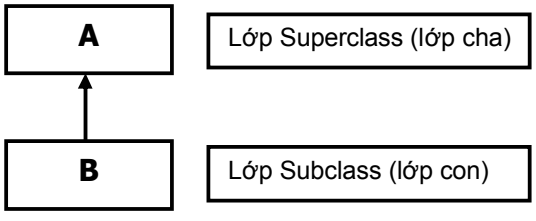
```
class Bclass{
  static int a;
  int b;
  public static void f1(){
         a++; // Truy xuất hợp lệ
         b++; // Phát sinh lỗi
  }
  public void f2(){
    b++;
  }
}
```

- Thực hiện gọi phương thức tĩnh:

```
Bclass.f1();
Bclass.f2(); // Phát sinh lỗi vì f2 không là static
```

III. SỰ KẾ THỪA LỚP:

Trong OOP, rất nhiều khi ta tạo một lớp mới dựa vào một lớp đã biết. Khi đó tất cả tính chất của lớp cũ (dữ liệu và phương thức) sẽ trở thành tính chất của lớp mới. Ta gọi lớp mới là lớp dẫn xuất từ lớp cũ (derived class) hay còn gọi là lớp con (subclass). Còn lớp cũ được gọi là siêu lớp (superclass).



http://www.ebook.edu.vn

1) Định nghĩa lớp kế thừa:

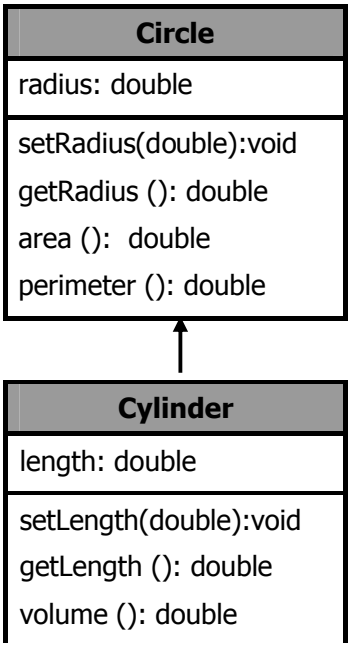
Cú pháp:

```
[modifier] class <Lớp Con> extends <Lớp Cha>{
  //  khai báo dữ liệu mới
  ...
  // định nghĩa phương thức mới
  ...
}
```

- Tất cả các lớp khi định nghĩa trong java đều là lớp dẫn xuất từ lớp Object mặc định của Java. Nhưng ta có thể bỏ qua khai báo **extends Object** trong khi viết chương trình (đây là dạng kế thừa ngầm định).

Ví dụ:

- Xét quan hệ giữa lớp Cylinder (hình trụ) và lớp Circle (hình tròn). Nếu đã có lớp Circle, muốn định nghĩa lớp Cylinder, ta dễ dàng nhận thấy rằng Cylinder là hình ảnh đặc biệt của Circle. Do đó ta có thể dẫn xuất từ lớp Circle để định nghĩa lớp Cylinder.



Mô hình UML

- Lớp Circle.java

```
package shape;
public class Circle{
  // Khai báo thành phần thuộc tính
  private double radius;
  // Constructor
  public Circle(){
```


http://www.ebook.edu.vn

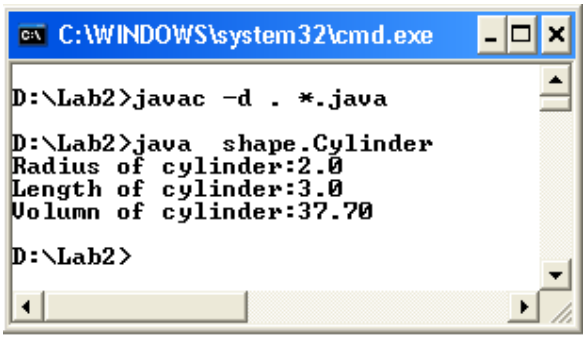
```
    radius=1.0;
  }
  public Circle(double r){
    radius=r;
  }
  // Cập nhật giá trị cho thuộc tính radius
  public void setRadius(double r){
    radius=r;
  }
  // đọc giá trị của thuộc tính radius
  public double getRadius(){
    return radius;
  }
  // tính diện tích
  public double area(){
    return  radius*radius*Math.PI;
  }
  // tính chu vi
  public double perimeter(){
    return 2*radius*Math.PI;
  }
}
```

- Lớp Cylinder.java

```
package  shape;
import java.text.DecimalFormat;
public class Cylinder extends Circle{
  // Khai báo dữ liệu mới
  private double length;
  // Constructors
  public Cylinder(){
    super();
    length=1.0;
  }
  public Cylinder(double radius,double length){
    super(radius);
```


http://www.ebook.edu.vn

```
    this.length=length;
  }
  // cập nhật giá trị thuộc tính length
  public void setLength(double length){
    this.length=length;
  }
  // đọc giá trị thuộc tính length
  public double getLength(){
    return length;
  }
  // tính thể tích
  public double volume(){
    return super.area()*length;
  }
  // phương thức chính
  public static void main(String args[]){
    Cylinder cyl  = new Cylinder(2.0,3.0);
    DecimalFormat fmt = new DecimalFormat("0.00");
    System.out.println("Radius of cylinder:" +
                       cyl.getRadius());
    System.out.println("Length of cylinder:" +
                       cyl.getLength());
    System.out.println("Volumn of cylinder:" +
                       fmt.format(cyl.volume()));
  }
}
```



Kết quả của Cylinder.java

http://www.ebook.edu.vn

2) Từ khóa super:

Từ khóa super dùng để tham chiếu đến lớp cha của lớp đang xét tương tự như từ khóa this là một tham chiếu đến bản thân lớp đang xét.

Cú pháp:

```
super([params]) //gọi Contructor lớp cha
super.<methodName>([params]) //gọi phương thức lớp cha
```

3) Định nghĩa lại phương thức (Orverriding method)

Do tính thừa kế, lớp con nhận tất cả các phương thức public hay protected của lớp cha làm của mình và cũng có thể thay thế các phương thức này nếu thấy cần thiết bằng cách khai báo chồng.

Ví dụ:

- Trong ví dụ trên, do tính kế thừa, lớp Cylinder nhận phương thức area () của lớp Circle là phương thức của mình, nhưng trong ngữ cảnh lớp Cylinder, cách xử lý của phương thức area () không còn phù hợp, khi đó lớp Cylinder có thể khai báo chồng phương thức area () của lớp cha nó như sau:

- Lớp Circle.java

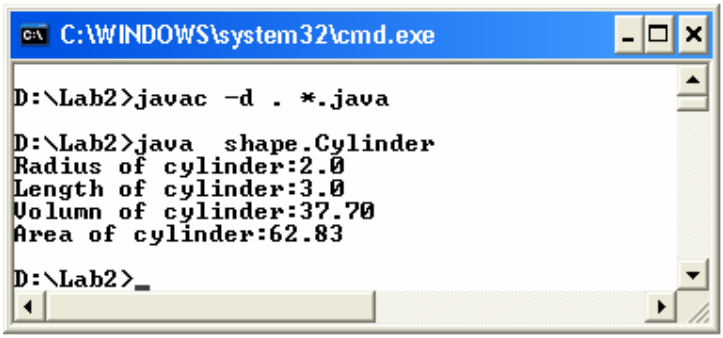
```
package shape;
public class Circle{
  // khai báo thành phần thuộc tính
  private double radius;
  ...
  // tính diện tích
  public double area(){
    return  radius*radius*Math.PI;
  }
  // tính chu vi
  public double perimeter(){
    return 2*radius*Math.PI;
  }
}
```

- Lớp Cylinder.java

```
package shape;
import java.text.DecimalFormat;
public class Cylinder extends Circle{
  // khai báo dữ liệu mới
  private double length;
```


http://www.ebook.edu.vn

```
    ...
    // định nghĩa lại phương thức area()
    public double area(){
      return  2*super.area() + super.perimeter()*length ;
    }
    // tính thể tích
    public double volume(){
      return super.area()*length;
    }
    public static void main(String args[]){
      Cylinder  cyl  = new Cylinder(2.0,3.0);
      DecimalFormat fmt = new DecimalFormat("0.00");
      System.out.println("Radius of cylinder:" +
                         cyl.getRadius());
      System.out.println("Length of cylinder:" +
                         cyl.getLength());
      System.out.println("Volumn of cylinder:" +
                         fmt.format(cyl.volume()));
      System.out.println("Area of cylinder:" +
                         fmt.format(cyl.area()));
    }
}
```



Kết quả của Cylinder.java

4) Các bổ từ của lớp (class modifiers):

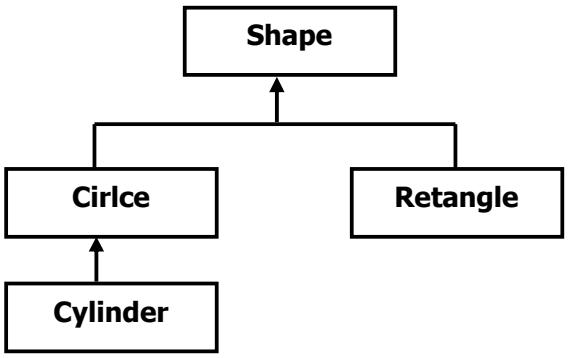
Lớp trong java có 3 tính chất đặc trưng bởi 3 từ khóa bổ từ sau:

- public : lớp có thể được truy cập từ các khối khác (packages).
- final : lớp không thể tạo dẫn xuất (lớp hằng)
- abstract : lớp trừu tượng.

http://www.ebook.edu.vn

5) Lớp trừu tượng

Đôi khi ta định nghĩa một lớp mà không thể biết và định nghĩa các thành phần, phương thức của nó. Ta khai báo lớp đó là một lớp trừu tượng. Các thành phần ở các lớp dẫn xuất sẽ được khai báo cụ thể.



Ta định nghĩa lớp Shape là lớp trừu tượng, do không thể định nghĩa các thành phần của lớp một cách cụ thể.

Cú pháp:

```
public abstract class <tên lớp trừu tượng>{
  ...
  public abstract <kiểu dữ liệu> Phương thức();
  ...
}
```

Ví dụ:

```
public abstract class Shape{
  public abstract double area();
}
// lớp Circle dẫn xuất từ lớp Shape
public class Circle extends Shape{
  public  double area(){
  // chi tiết xử lý tính diện tích
  ...
  }
}
// lớp Rectangle dẫn xuất từ lớp Shape
public class Rectangle extends Shape{
  public double area(){
  // chi tiết xử lý tính diện tích
  ...
```


http://www.ebook.edu.vn

```
    }
}
```

Lưu ý:

- Một phương thức tĩnh (static) hay riêng tư (private) không được định nghĩa là trừu tượng, vì khi được khai báo là private hay static, các phương thức này sẽ không được khai báo chồng (overriding) bởi các lớp dẫn xuất.

6) Lớp trong (Inner class):

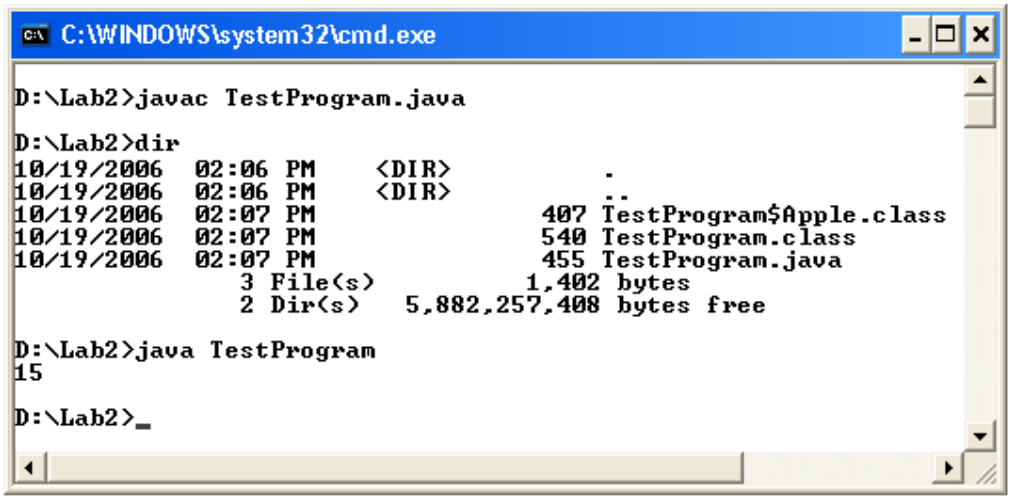
Lớp trong, hay lớp lồng (nest class), là lớp được định nghĩa trong phạm vi một lớp khác.

Ví dụ:

```
public class TestProgram{
  static int currentCount;
  // inner class
  static class Apple{
    private int weight;
    public Apple(int weight){
      this.weight=weight;
      currentCount++;
    }
    public int getWeight(){
      return weight;
    }
  } //end inner class
  public  static void main(String args[]){
    Apple a = new Apple(15);
    System.out.println(a.getWeight());
  }
}
```

Lớp trong Apple

http://www.ebook.edu.vn



Kết quả của TestProgram.java

Đặc tính của lớp trong là:

- Lớp trong có thể tham chiếu dữ liệu và phương thức định nghĩa tại lớp (lồng) ngoài. Vì vậy ta không cần chuyển tham chiếu của lớp ngoài đến phương thức tạo dựng của lớp trong.
- Lớp trong có thể làm cho chương trình trở nên đơn giản và rõ ràng.
- Lớp trong chỉ nhằm để hỗ trợ công việc của lớp ngoài và được biên dịch thành lớp OutClassNam$InnerClassName.class.

IV. GIAO DIỆN (interface)

Giao diện là một khái niệm mới trong ngôn ngữ lập trình hiện đại như Java, C#, C++,... Ở đây đừng nhầm lẫn interface với khái niệm giao diện đồ họa. Interface là giao diện của một lớp đối tượng, nó là các khai báo phương thức hay thuộc tính public để bên ngoài truy xuất.

Giao diện của đối tượng chính là phần đặc tả (chỉ chứa khai báo và không có phần cài đặt) của một lớp. Cài đặt cụ thể là công việc của lớp. Một đối tượng có thể đưa ra cùng lúc nhiều giao diện để chương trình bên ngoài truy xuất. Thay vì sử dụng lớp để truy xuất toàn bộ phương thức thì nhà phát triển có thể giới hạn các phương thức mà người dùng sử dụng thông qua giao diện.

Khi sử dụng giao diện, ta bắt buộc phải tham chiếu đến một lớp cụ thể tương tự như phép toán ép kiểu thông thường.

Cú pháp:

```
[modifier] interface <InterfaceName>{

}
```

Ví dụ: Ta định nghĩa 2 giao diện truy xuất lớp Stack như sau:

- Giao diện IStackOne.java

```
public interface IStackOne{
```


http://www.ebook.edu.vn

```
  public boolean isFull();
  public boolean isEmpty();
}
```

- Giao diện IStackTwo.java

```
public interface  IStackTwo{
  public void push(Object item);
  public Object pop();
}
```

- Cấu trúc khai báo lớp thực thi giao diện:

```
[modifier] class <ClassName> implements <I1>, <I2>{
  // Định nghĩa lớp
}
```

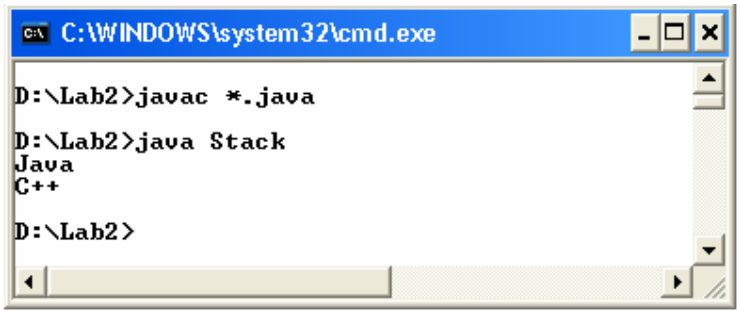
Ví dụ: Ta định nghĩa lớp Stack thực thi 2 giao diện IStackOne và IStackTwo

- Tập tin Stack.java

```
public class Stack implements IStackOne, IStackTwo{
  // khai báo dữ liệu
  private Object data[];
  private int top;
  // constructor
  public Stack(int size){
    data = new Object[size];
    top=-1;
  }
  // phương thức kiểm tra Stack đầy
  public boolean isFull(){
    return top==data.length-1;
  }
  // phương thức kiểm tra Stack rỗng
  public boolean isEmpty(){
    return top==-1;
  }
  //phương thức thêm phần tử vào Stack
  public void push(Object item){
    if(!isFull())
      data[++top]=item;
```


http://www.ebook.edu.vn

```
  }
  //phương thức lấy phần tử khỏi Stack
  public Object pop(){
    return data[top--];
  }
  // phương thức chính
  public static void main(String args[]){
    Stack  s= new Stack(10);
    IStackOne isone =(IStackOne)s;
    IStackTwo istwo =(IStackTwo)s;
    istwo.push("C++");
    istwo.push("Java");
    while(!isone.isEmpty()){
      System.out.println(istwo.pop());
    }
  }
}
```



Kết quả của Stack.java

V. MỘT SỐ LỚP THƯ VIỆN THƯỜNG DÙNG:

1) Lớp String:

Khởi tạo chuỗi:

```
String  s1 = new String("Welcome to java !");
```

Hoặc câu lệnh tạo chuỗi đơn giản:

```
String s1= "Welcome to java !";
```

So sánh bằng giữa 2 nội dung của 2 đối tượng.

```
if(s1.equals(s2))
  System.out.println("s1 same s2") ;
else
```


http://www.ebook.edu.vn

```
System.out.println("s1 different s2") ;
```

So sánh đối chiếu giữa 2 chuỗi:

```
s1.compareTo(String s2)
```

- Phương thức trả về 0 nếu s1 bằng s2
- Phương thức trả về < 0 nếu s1 < s2
- Phương thức trả về > 0 nếu s1>< s2

Ghép hai chuỗi:

```
s1.concat(String s2)
```

- Phương thức trả về chuỗi đã được ghép.

Trích chuỗi con từ 1 chuỗi:

```
s.substring( int begin, int end)
```

- Phương thức trả về chuỗi con bắt đầu từ chỉ số **begin** đến **end-1**

Trả về chiều dài chuỗi.

```
s.length()
```

Truy xuất ký tự tại chỉ số index trong chuỗi.

```
s.charAt(int index)
```

Tìm vị trí chuỗi s2 xuất hiện trong chuỗi s1 bắt đầu từ chỉ số index

```
s1.indexOf( String s2, int index):
```

- Phương thức trả về -1 nếu không có chuỗi s2 trong s1.
- Phương thức trả về 1 nếu không có chuỗi s2 trong s1.

```
String s1= "Welcome to java !";
String s2= "java";
int pos = s1.indexOf(s2,0));  // giá trị của pos là 11
```


http://www.ebook.edu.vn

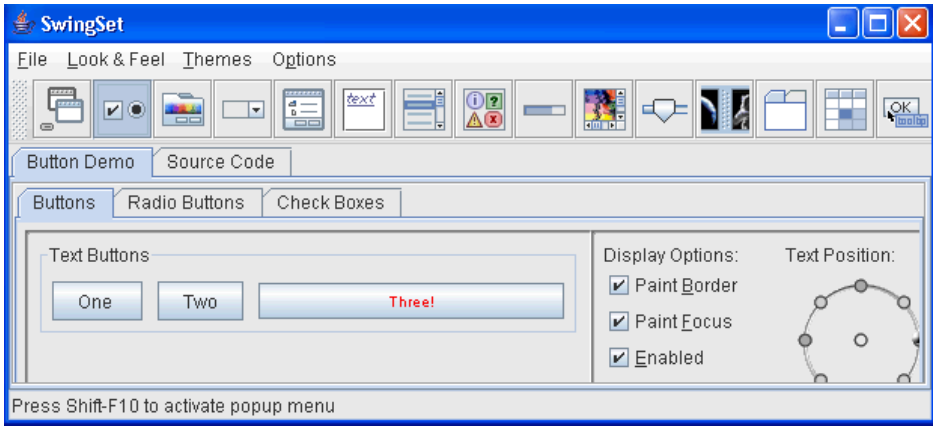
Chương 4 LẬP TRÌNH GIAO DIỆN ĐỒ HỌA

Sau bài học này, học viên có thể:

- Sử dụng được các thành phần GUI (Graphics User Interface) để phát triển ứng dụng đồ họa.
- Hiểu được mô hình biến cố (sự kiện) trên thành phần GUI.
- Xác định được các giao diện lắng nghe và xử lý biến cố cho các thành phần.

I. KHÁI NIỆM:

- Khi Java được phát hành, các thành phần đồ họa tập trung vào thư viện mang tên AWT (Abstract Windows Toolkit). AWT rất hữu ích trong thiết kế chương trình ứng dụng GUI nhưng không dùng để thiết kế những project GUI toàn diện. Ngoài ra, AWT có khả năng mắc lỗi ở hệ nền. Từ Java 2, các thành phần giao diện người dùng thay thế bằng thư viện Swing linh hoạt, đa năng và mạnh mẽ hơn.Thành phần Swing ít phụ thuộc hệ nền hơn. Mặc dù thành phần AWT vẫn được hỗ trợ trong Java 2, nhưng bạn nên lập trình với thành Swing, bởi lẽ thành phần AWT dần dần sẽ lỗi thời.
- Java cung cấp thư viện lớp rất phong phú nhằm trợ giúp thiết lập giao diện người dùng dạng đồ họa. Ta có thể sử dụng nhiều lớp thiết kế GUI khác nhau như khung, panel, nút nhấn, trường văn bản, vùng văn bản, hộp combo, ô chọn, nút radio, menu, thanh cuốn,... để thiết kế giao diện người dùng.



II. TẠO KHUNG (FRAME):

Công việc đầu tiên ta muốn thực hiện với lập trình đồ họa là hiển thị cửa sổ. Lớp dùng để tạo khung cửa sổ là JFrame (javax.swing).

- Khởi tạo cửa sổ:

  JFrame (): tạo khung cửa sổ không có tiêu đề.
- JFrame (String title): tạo khung cửa sổ có tiêu đề chỉ định.

Các phương thức thường dùng trên Jframe:

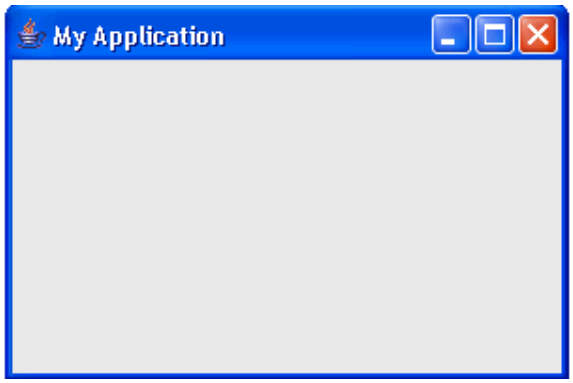
- void setSize (int w, int h): qui định kích thước cửa sổ theo chiều rộng và chiều cao.
- void setTitle (String title): đặt lại tiêu đề cho cửa sổ.
- void setResizable (boolean b): cho phép cửa sổ co giản hay không

http://www.ebook.edu.vn

- void setIconImage (Image img): gán lại biểu tượng cho cửa sổ.
- void setVisible (boolean b): hiện hoặc ẩn cửa sổ ra màn hình

Ví dụ:

```
package chapter04;
import javax.swing.*;
public class Main{
  public static void main(String[] args){
    JFrame  frm= new JFrame("My Application");
    frm.setSize(300,200);
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.setVisible(true);
  }
}
```



Kết quả của MyFrame.java

Thông thường khi tạo cửa sổ, ta nên kế thừa thay cho việc sử dụng trực tiếp.

Ví dụ:

```
package chapter04;
import javax.swing.*;
public class MyFrame extends JFrame{
  public MyFrame(String title){
    super(title);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
  public static void main(String[] args){
    MyFrame  frm = new MyFrame("My Appplication");
    frm.setSize(300,200);
    frm.setVisible(true);
  }
```


http://www.ebook.edu.vn

```
}
```

III. LẬP TRÌNH THEO BIẾN CỐ (EVENT):

Chương trình đồ họa thường được điều khiển bởi biến cố. Trong lập trình biến cố, mã chương trình thi hành khi biến cố được kích hoạt.

1) Biến cố và nguồn phát sinh biến cố:

- Khi vận hành chương trình đồ họa Java, chương trình tương tác với người dùng, còn các biến cố chi phối việc thi hành chương trình. Biến cố là loại tín hiệu báo cho chương trình biết có sự kiện nào đó đã xảy ra. Biến cố phát sinh từ hành động bên ngoài của người dùng, như thao tác di chuyển mouse, nhấn mouse, gõ phím,...Chương trình có thể chọn phản hồi hay bỏ qua biến cố.
- Thành phần GUI, nơi phát sinh biến cố, gọi là đối tượng nguồn (source object). Ví dụ, nhấp nút bất kỳ sẽ kích hoạt một biến cố. Nút được nhấp chính là đối tượng nguồn. Để truy cập đối tượng nguồn ta dựa vào phương thức getSource () trên biến cố. Các loại biến cố rất đa dạng xử lý những hành động của người dùng.
- Lớp biến cố chứa bất kỳ giá trị dữ liệu nào thích hợp với loại biến cố cụ thể. Chẳng hạn, lớp KeyEvent định nghĩa mọi hằng chính, như VK_DOWN (phím mũi tên xuống), và phương thức getKeyChar () (trả về ký tự phối hợp với biến cố).

Các đối tượng nguồn và một số loại biến cố phát sinh thường dùng:

Hành động của người dùng	Đối tượng nguồn	Loại biến cố phát sinh
Nhấp nút	JButton	ActionEvent
Thay đổi văn bản	JTextComponent	TextEvent
Nhấn Enter trên trường văn bản	JTextField	ActionEvent
Chọn khoản mục mới	JComboBox	ItemEvent, ActionEvent
Nhấp ô chọn	JCheckBox, JRadioButton	ItemEvent, ActionEvent
Chọn khoản mục menu	JMenuItem	ActionEvent
Di chuyển thanh cuốn	JScrollBar	AdjustmentEvent
Dời tiêu điểm đến hoặc ra khỏi thành phần	Component	FocusEvent
Di chuyển mouse	Component	MouseEvent
Thả hoặc nhấn phím	Component	KeyEvent

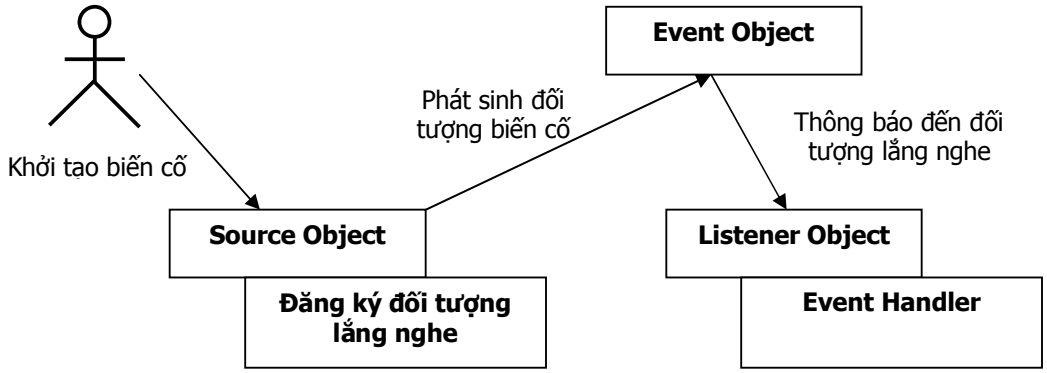
2) Đăng ký, lắng nghe và xử lý biến cố:

Java sử dụng mô hình dựa trên sự ủy nhiệm để xử lý biến cố. Hành động của người dùng tác động lên đối tượng nguồn sẽ phát sinh biến cố. Đối tượng nào quan tâm đến biến cố sẽ tiếp nhận biến cố. Đối tượng như thế gọi là đối tượng lắng nghe (listener). Muốn trở thành đối tượng lắng nghe, đối tượng đó phải được đối tượng nguồn đăng ký làm listener. Đối tượng nguồn sẽ duy trì danh sách đối tượng lắng nghe và thông báo đến tất

http://www.ebook.edu.vn

cả đối tượng lắng nghe đã đăng ký bằng cách gọi phương thức xử lý biến cố (gọi là handler) trên đối tượng lắng nghe, nhằm hồi đáp biến cố.

Mô hình biến cố:



- Phương thức đăng ký phụ thuộc vào loại biến cố. Thường theo cú pháp tổng quát là:
  addXListener (XListener)

Ví dụ:

- Đối với biến cố ActionEvent, phương thức mà đối tượng nguồn đăng ký là addActionListener.

Để hệ thống kích hoạt phương thức xử lý (handler) trên đối tượng lắng nghe, đối tượng lắng nghe phải thi hành phương thức xử lý chuẩn. Phương thức xử lý được định nghĩa trong giao diện lắng nghe biến cố tương ứng. Java cung cấp giao diện lắng nghe cho từng loại biến cố đồ họa. Mỗi đối tượng lắng nghe phải thực thi (implements) giao diện tương ứng.

Các loại biến cố, giao diện lắng nghe tương ứng thường dùng và các phương thức định nghĩa trong giao diện lắng nghe.

Lớp biến cố	Giao diện lắng nghe	Loại biến cố phát sinh
ActionEvent	ActionListener	actionPerformed (ActionEvent e)
ItemEvent	ItemListener	itemStateChanged (ItemEvent e)
TextEvent	TextListener	textValueChanged (TextEvent e)
FocusEvent	FocusListener	focusGained (FocusEvent e) focusLost (FocusEvent e)
MouseEvent	MouseListener	mousePressed (MouseEvent e) mouseReleased (MousEvent e) mouseClicked (MouseEvent e) mouseEntered (MouseEvent e) mouseExited (MouseEvent e)
	MouseMotionListener	mouseMoved (MouseEvent e) moouseDragged (MouseEvent e)
KeyEvent	KeyListener	keyPressed (KeyEvent e) keyReleased (KeyEvent e) keyTyped (KeyEvent e)
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged (AdjustmentEvent e)

http://www.ebook.edu.vn

3) Xử lý biến cố:

Đối tượng lắng nghe phải thực thi giao diện lắng nghe tương ứng. Lấy ví dụ, đối tượng lắng nghe dành cho đối tượng nguồn JButton buộc phải thực thi giao diện ActionListener. Giao diện ActionListener chứa phương thức actionPerformed (ActionEvent e). Phương thức này phải được định nghĩa lại trong lớp đối tượng lắng nghe. Ngay khi được thông báo, phương thức actionPerformed (ActionEvent e) sẽ được thi hành hầu xử lý biến cố.

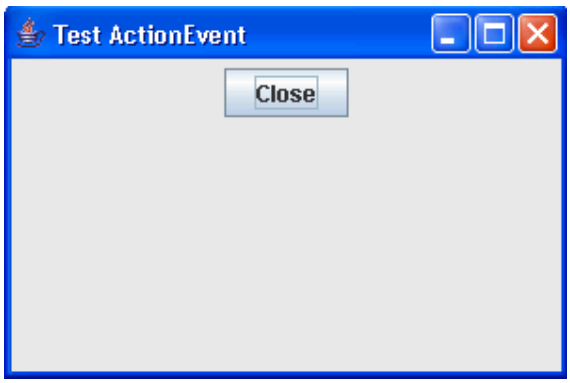
Ví dụ 2: Xử lý biến cố hành động (ActionEvent) đơn giản

- Chương trình sau hiển thị nút Close trong cửa sổ. Khi người dùng click nút Close chương trình sẽ kết thúc.

```
package chapter04;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TestActionEvent extends JFrame{
  private JButton btnClose;
  public TestActionEvent(){
    setTitle("Test ActionEvent");
    btnClose= new JButton("Close");
    getContentPane().setLayout(new FlowLayout());
    getContentPane().add(btnClose);
    // đăng ký đối tượng lắng nghe
    btnClose.addActionListener(new MyActionListener());
  }
  // định nghĩa lớp đối tượng lắng nghe
  class MyActionListener implements ActionListener{
    // định nghĩa lại phương thức hành động
    public void actionPerformed(ActionEvent e){
      if(e.getSource()==btnClose)
        System.exit(0);
    }
  }
  public static void main(String args[]){
    TestActionEvent frm = new TestActionEvent();
    frm.setSize(300,200);
```


http://www.ebook.edu.vn

```
    frm.setVisible(true);
  }
}
```



Kết quả của TestActionEvent.java

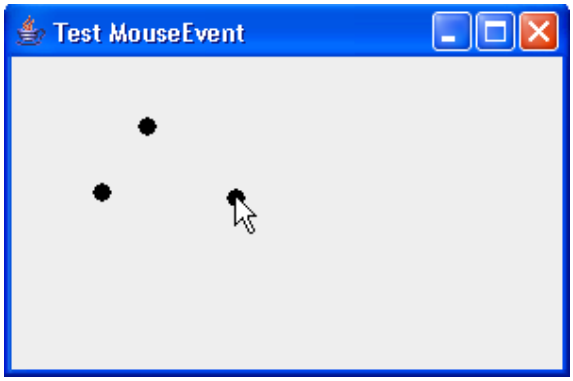
Ví dụ: Xử lý biến cố mouse (MouseEvent) đơn giản.

- Chương trình sau tạo khung cửa sổ và hiển thị hình tròn tô đầy tại con trỏ mouse khi nhấp mouse.

```
package chapter04;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TestMouseEvent extends JFrame{
  private int x,y=0;  // tọa độ  x,y
  public TestMouseEvent(){
  setTitle("Test MouseEvent");
  // đăng ký đối tượng lắng nghe
  addMouseListener(new MyMouseListener());
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
  public void paint(Graphics g){
    g.fillOval(x-5,y-5,10,10);  // vẽ hình oval
  }
  // định nghĩa lớp đối tượng lắng nghe
  class MyMouseListener implements MouseListener{
    // định nghĩa lại phương thức nhấp mouse
    public void mousePressed(MouseEvent e){
      x=e.getX();
```


http://www.ebook.edu.vn

```
        y=e.getY();
        repaint();  // gọi lại phương thức paint()
      }
      public void mouseReleased(MouseEvent e){ }
      public void mouseClicked(MouseEvent e){ }
      public void mouseEntered(MouseEvent e){ }
      public void mouseExited(MouseEvent e){ }
    }
    public static void main(String args[]){
      TestMouseEvent frm = new TestMouseEvent();
      frm.setSize(300,200);
      frm.setVisible(true);
    }
}
```



Kết quả của TestMouseEvent.java

Lưu ý:

- Quên không đăng ký đối tượng lắng nghe là lỗi thường gặp trong xử lý biến cố. Nếu hệ thống không khai báo đối tượng lắng nghe, đối tượng lắng nghe không thể hoạt động trên biến cố.

4) Lớp điều hợp và lớp không tên:

- Mô hình biến cố Java rất linh hoạt, nó cho phép hiệu chỉnh và chấp nhận biến thể. Một trong những biến thể hữu ích của mô hình là lớp điều hợp (adapter) xử lý biến cố. Lớp điều hợp được đăng ký dưới dạng đối tượng lắng nghe cho đối tượng nguồn. Java cung cấp lớp điều hợp tiện dụng cho từng giao diện lắng nghe biến cố với nhiều phương thức xử lý. Lớp điều hợp chỉ là một thực thi đơn giản của giao diện lắng nghe biến cố, chứa các phương thức rỗng cho từng phương thức định nghĩa trong giao diện.
- Chỉ có những giao diện lắng nghe chứa từ 2 phương thức trở lên thì mới có lớp điều hợp. Thông thường một giao diện lắng nghe XListener thì sẽ có một XAdapter.

http://www.ebook.edu.vn

Ví dụ: MouseListener thì sẽ có MouseAdapter. Lớp đối tượng lắng nghe có thể mở rộng (extends) lớp điều hợp để xử lý biến cố thay vì thực thi (implements) giao diện lắng nghe.

Ví dụ:

- Ta có thể sử dụng lớp điều hợp thay vì giao diện lắng nghe trong chương trình xử lý biến cố mouse đơn giản.

```
package chapter04;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TestMouseEventUsingAdapter extends JFrame{
  private int x,y=0;  // tọa độ  x,y
  public TestMouseEventUsingAdapter(){
    setTitle("Test MouseEvent");
    // đăng ký đối tượng lắng nghe
    addMouseListener(new MyMouseListener());
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
  public void paint(Graphics g){
    g.fillOval(x-5,y-5,10,10);  // vẽ hình oval
  }
  // định nghĩa lớp đối tượng lắng nghe
  class MyMouseListener extends MouseAdapter{
    // định nghĩa lại phương thức nhấp mouse
    public void mousePressed(MouseEvent e){
      x=e.getX();
      y=e.getY();
      repaint();  // gọi lại phương thức paint()
    }
  }
  public static void main(String args[]){
    TestMouseEventUsingAdapter frm = new
                          TestMouseEventUsingAdapter();
    frm.setSize(300,200);
    frm.setVisible(true);
  }
}
```


http://www.ebook.edu.vn

- Chương trình có thể rút gọn hơn nữa khi dùng lớp trong không tên. Lớp trong không tên (anonymouse inner class) là lớp lồng trong không có tên gọi.

Ví dụ:

```
package chapter04;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TestMouseEventUsingAnonymousInnerClass extends
JFrame{
  private int x,y=0;  // tọa độ  x,y
  public TestMouseEventUsingAnonymousInnerClass(){
    setTitle("Test MouseEvent");
    // đăng ký lớp điều hợp làm listener
    addMouseListener(new MouseAdapter(){
      // định nghĩa lại phương thức nhấp mouse
      public void mousePressed(MouseEvent e){
        processMousePressed(e);
      }
    });
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
  public void paint(Graphics g){
    g.fillOval(x-5,y-5,10,10);  // vẽ hình oval
  }
  public void processMousePressed(MouseEvent e){
    x=e.getX();
    y=e.getY();
    repaint();  // gọi lại phương thức paint()
  }
  public static void main(String args[]){
    TestMouseEventUsingAnonymousInnerClass frm =
              new TestMouseEventUsingAnonymousInnerClass();
    frm.setSize(300,200);
    frm.setVisible(true);
  }
}
```


http://www.ebook.edu.vn

IV. LỚP QUẢN LÝ BỐ CUC:

1) *Khái niệm về lớp quản lý bố cục:*

Ở nhiều hệ thống cửa sổ khác, các thành phần giao diện thường được sắp xếp dựa vào những số đo pixel mã hóa cứng.

Ví dụ:

- Đặt nút tại vị trí (10,10) trong cửa sổ. Dùng số đo pixel mã hóa cứng, giao diện người dùng trong rất hoàn hảo trên hệ thống này nhưng lại không sử dụng được trên hệ thống khác. Lớp quản lý bố cục (layout manager) của Java sẽ đưa ra mức độ phân chia giúp tự động sắp xếp giao diện người dùng trên tất cả hệ thống cửa sổ.

Những thành phần GUI được đặt vào vùng chứa. Mỗi vùng chứa có một lớp quản lý bố cục sắp xếp thành phần GUI trong phạm vi vùng chứa.

Một số lớp quản lý bố cục thường dùng là:

- FlowLayout
- GridLayout
- BorderLayout
- CardLayout
- GridbagLayout

Lớp quản lý bố cục được ấn định trong vùng chứa, như JFrame, JPanel hoặc JApplet.

Cú pháp:

```
Tên_vùng_chứa.setLayout(new <tên lớp quản lý bố cục>());
```

Để thêm thành phần vào vùng chứa, sử dụng phương thức:

```
void add(Component p)
void add(Component p,int index)
```

Ví dụ:

- Thêm nút btnClose vào vùng chứa container:

```
container.add(btnClose);
```

2) *Bố cục FlowLayout:*

- Dùng sắp xếp các thành phần từ trái qua phải theo một trật tự như chúng đã thêm vào.
- Có thể canh lề dựa vào 3 hằng:
  - *FlowLayout.RIGHT*
  - *FlowLayout.CENTER*
  - *FlowLayout.LEFT.*

Phương thức khởi tạo:

```
public  FLowLayout(int align, int hgap, int vgap)
```

- Tạo FlowLayout với kiểu canh lề, khoảng trống ngang và dọc cụ thể.

```
public  FLowLayout(int align)
```


http://www.ebook.edu.vn

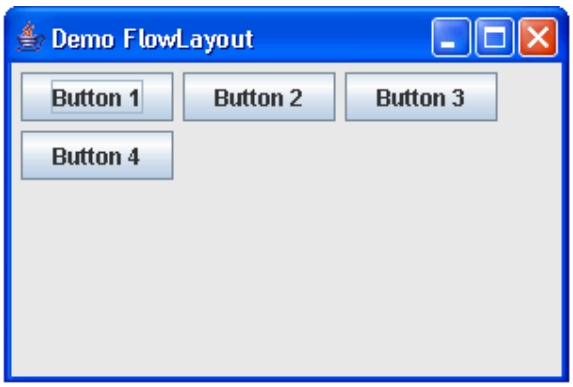
- Tạo FlowLayout với kiểu canh lề cụ thể, khoảng trống ngang và dọc mặc định là 5 pixel.

```
public  FLowLayout()
```

- Tạo lớp FlowLayout với kiểu canh lề giữa mặc định, khoảng trống ngang và dọc mặc định là 5 pixel.

Ví dụ:

```
package chapter04;
import java.awt.*;
import javax.swing.*;
public class ShowFlowLayout extends JFrame{
  JButton b1,b2,b3,b4;
  public ShowFlowLayout(){
    setTitle("Demo FlowLayout");
    Container container = getContentPane();
    container.setLayout(new FlowLayout(FlowLayout.LEFT));
    container.add(b1= new JButton("Button 1"));
    container.add(b2= new JButton("Button 2"));
    container.add(b3= new JButton("Button 3"));
    container.add(b4= new JButton("Button 4"));
  }
  public static void main(String[] args){
    ShowFlowLayout frm = new ShowFlowLayout();
    frm.setSize(300,200);
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.show();
  }
}
```



Kết quả của ShowFlowLayout.java

http://www.ebook.edu.vn

3) Bố cục GridLayout:

- Sắp xếp các thành phần theo khung lưới kẻ ô (ma trận) với số dòng và cột do phương thức tạo dựng quyết định.
- Thành phần được đặt vào lưới từ trái sang phải, bắt đầu từ hàng đầu tiên, tiếp đến là hàng thứ hai, ...

Phương thức khởi tạo:

```
public GridLayout(int rows,int columns,int hgap,int vgap)
```

- Tạo GridLayout với số dòng, cột, cùng khoảng trống ngang và dọc định rõ.

```
public GridLayout(int rows,int columns)
```

- Tạo GridLayout với số dòng, cột định rõ, khoảng trống ngang và dọc mặc định là 0 pixel.

Ví dụ:

```
package chapter04;
import java.awt.*;
import javax.swing.*;

public class ShowGridLayout extends JFrame{
  JButton btn[];
  public ShowGridLayout(){
    setTitle("Demo GridLayout");
    Container container = getContentPane();
    container.setLayout(new GridLayout(4,3,5,5));
    btn = new JButton[10];
    for(int i=0;i<btn.length;i++){
      btn[i]= new JButton("Button " +(i+1));
        container.add(btn[i]);
    }
  }
  public static void main(String[] args){
    ShowGridLayout frm = new ShowGridLayout();
    frm.pack();
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.show();
  }
}
```


http://www.ebook.edu.vn

Kết quả của ShowGridLayout.java

4) *Bố cục BorderLayout:*

- Lớp quản lý BorderLayout phân chia vùng chứa thành 5 khu vực (North, South,West, East, Center).
- Mỗi khu vực được định nghĩa bởi 1 hằng số sau:
  - *BorderLayout.NORTH*
  - *BorderLayout.SOUTH*
  - *BorderLayout.WEST*
  - *BorderLayout.EAST*
  - *BorderLayout.CENTER*

Phương thức khởi tạo:

```
public BorderLayout(int hgap, int vgap)
```

- Tạo BorderLayout với khoảng trống ngang và dọc định rõ xen giữa các thành phần.

```
public BorderLayout()
```

- Tạo BorderLayout không có khoảng trống ngang và dọc.

Ví dụ:

```
package chapter04;
import java.awt.*;
import javax.swing.*;
public class ShowBorderLayout extends JFrame{
  public ShowBorderLayout(){
    setTitle("Demo GridLayout");
    Container container = getContentPane();
    container.setLayout(new BorderLayout(5,5));
    container.add(new JButton("North"),BorderLayout.NORTH);
    container.add(new JButton("South"),BorderLayout.SOUTH);
    container.add(new JButton("West"),BorderLayout.WEST);
    container.add(new JButton("East"),BorderLayout.EAST);
    container.add(new JButton("Center"),BorderLayout.CENTER);
  }
```


http://www.ebook.edu.vn

```
  public static void main(String[] args){
    ShowBorderLayout frm = new ShowBorderLayout();
    frm.pack();
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.show();
  }
}
```

Kết quả của ShowBorderLayout.java

Lưu ý:

- Luôn luôn ấn định kiểu trình bày thật rõ ràng cho vùng chứa, mặc dù lớp quản lý BorderLayout được áp dụng theo mặc định cho khung nội dung của JFrame

V. JPanel (vùng chứa):

- Giả sử ta muốn đặt 10 nút và một trường văn bản trên khung. Nút được đặt theo sơ đồ khung lưới, nhưng trường văn bản phải đặt vào hàng riêng biệt. Rất khó thu được kết quả mong muốn nếu đặt tất cả các thành phần trên vào một vùng chứa đơn lẻ.
- Vớp lập trình đồ họa Java, ta có thể chia nhỏ cửa sổ thành nhiều panel. Những panel này hoạt động dưới dạng vùng chứa nhỏ hơn để ghép nhóm các thành phần giao diện người dùng.

1) Sử dụng Jpanel:

Phương thức khởi tạo panel:

```
public JPanel()
```

- Tạo vùng chứa ấn định bố cục FlowLayout mặc định.

```
public JPanel(LayoutManager layout)
```

- Tạo vùng chứa với 1 bố cục định rõ.

Chẳng hạn, muốn thêm nút vào panel p, ta sử dụng lệnh:

```
JPanel p= new JPanel();
p.add(new JButton("ButtonName"));
```

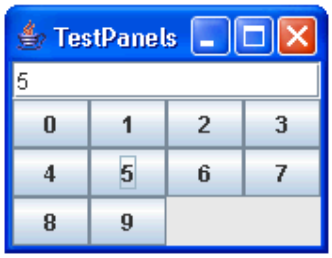
Ví dụ:

```
package chapter04;
import java.awt.*;
import java.awt.event.*;
```


http://www.ebook.edu.vn

```
import javax.swing.*;
public class TestPanel extends JFrame{

  private JTextField txtDisplay;
  private JButton btnNum[];
  public TestPanel(){
    setTitle("TestPanels");
    Container c= getContentPane();
    c.setLayout(new BorderLayout());
    txtDisplay =new JTextField();
    // tạo panel để ghép các nút số bằng bố cục GridLayout
    JPanel p = new JPanel();
    p.setLayout(new GridLayout(3,3));
    btnNum = new JButton[10];
    for(int i=0;i<btnNum.length;i++){
      p.add(btnNum[i]= new JButton(String.valueOf(i)));
      btnNum[i].addActionListener(new MyActionListener());
    }
    // đặt panel vào giữa khung
    c.add(p,BorderLayout.CENTER);
    // đặt trường văn bản bên trên panel
    c.add(txtDisplay,BorderLayout.NORTH);
  }
  class MyActionListener implements ActionListener{
    public void actionPerformed(ActionEvent e){
      String actionCommand = e.getActionCommand();
      if(e.getSource() instanceof JButton)
        txtDisplay.setText(actionCommand);
    }
  }
  public static void main(String args[]){
    TestPanel frm  = new TestPanel();
    frm.pack();
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.setVisible(true);
  }
}
```


http://www.ebook.edu.vn



Kết xuất của TestPanel.java

2) Sử dụng Panel vẽ đối tượng đồ họa:

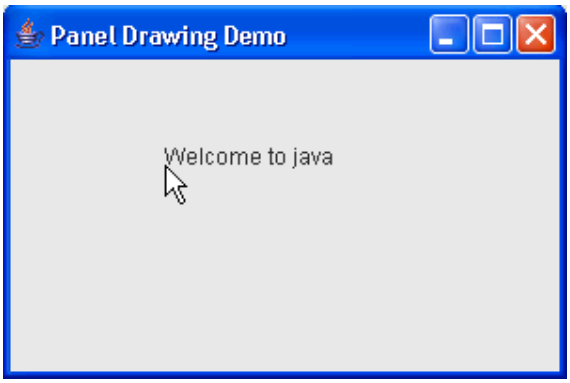
- Panel thường được dùng làm vật chứa nhỏ để ghép nhóm các thành phần hầu thu được bố cục mong muốn. Công dụng quan trọng khác của JPanel là vẽ đối tượng đồ họa.
- Muốn vẽ trên panel, ta cần tạo lớp mới mở rộng lớp JPanel và định nghĩa lại phương thức paintComponent () hầu chỉ thị cho panel cách vẽ đối tượng đồ họa. Mặc dù được phép vẽ thông điệp, hình dạng và hiển thị hình ảnh trực tiếp lên khung (JFrame) nhưng ta vẫn nên dùng JPanel vì theo cách này, bức vẽ sẽ không ảnh hưởng đến các thành phần khác.

Ví dụ: Vẽ trên panel

- Chương trình tạo một lớp con của JPanel sẽ hiển thị thông điệp. Ta dùng mouse di chuyển thông điệp và thông điệp luôn xuất hiện tại thời điểm đặt mouse.

```
package chapter04;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class PanelDrawingDemo extends JFrame{
  public PanelDrawingDemo(){
    setTitle("Panel Drawing Demo");
    PaintPanel p =new PaintPanel("Welcome to java");
    getContentPane().setLayout(new BorderLayout());
    getContentPane().add(p,BorderLayout.CENTER);
  }
  public static void main(String args[]){
    PanelDrawingDemo frm = new PanelDrawingDemo();
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.setSize(300,200);
    frm.setVisible(true);
  }
}
```

```
// class PaintPanel draw a message
class PaintPanel extends JPanel{
  private String message;
  private int x,y;
  public PaintPanel(String msg){
    message=msg;
    this.addMouseMotionListener(new MouseMotionAdapter(){
      public void mouseMoved(MouseEvent e){
        x=e.getX();
        y=e.getY();
        repaint();
      }
    });
  }
  public void paintComponent(Graphics g){
    super.paintComponent(g); //clear the viewing area
    g.drawString(message,x,y);
  }
}
```



Kết xuất của PanelDrawingDemo.java

VI. Thành phần giao diện người dùng:

1) JButon (nút nhấn):

Là thành phần khởi tạo biến cố khi được nhấn.

Phương thức tạo dựng:

```
public JButton(String text)
```

- Tạo nút với nhãn chỉ định

```
public JButton(Icon icon)
```

- Tạo nút với biểu tượng chỉ định.

```
public JButton(String text, Icon icon)
```

- Tạo nút với nhãn và biểu tượng chỉ định

Một số phương thức thường dùng trên Jbuton:

```
void setText(String text)
```

- Đặt lại nhãn cho nút

```
void setIcon(Icon icon)
```

- Đặt lại biểu tượng cho nút

```
void setEnabled( boolean b)
```

- Đặt tính hiệu lực cho nút

```
void setToolTipText(String text)
```

- Đặt chú thích (tooltiptext) cho nút

```
void setMnemonic(char c)
```

- Định phím tắt

```
void setVisible(boolean b)
```

- Cho nút ẩn hoặc hiện

Ví dụ:

- Chương trình sau hiển thị một chuỗi trong trường văn bản, click 2 nút left, right để canh lề trái, lề phải và click nút close để kết thúc chương trình.

```
package chapter04;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class ButtonDemo extends JFrame{
  private JButton btnLeft,btnRight,btnClose;
  private JTextField txtdisplay;
  public ButtonDemo(){
    setTitle("Button Demo");
    Container c = getContentPane();
    JPanel p = new JPanel();
    p.setLayout(new FlowLayout());
    p.add(btnLeft= new JButton("Left"));
    p.add(btnRight= new JButton("right"));
    Icon icon = new ImageIcon(this.getClass().getResource(
                              "images/close.jpg"));
    p.add(btnClose= new JButton("Close",icon));
    c.add(p,BorderLayout.CENTER);
```

```
    c.add(txtdisplay=new JTextField("Welcome to java",20)
                                        ,BorderLayout.NORTH);
    MyActionListener myAction = new MyActionListener();
    btnLeft.addActionListener(myAction);
    btnRight.addActionListener(myAction);
    btnClose.addActionListener(myAction);
  }
  class MyActionListener implements ActionListener{
    public void actionPerformed(ActionEvent e){
      if(e.getSource()==btnLeft){
        txtdisplay.setHorizontalAlignment(JTextField.LEFT);
      }else if(e.getSource()==btnRight){
        txtdisplay.setHorizontalAlignment(JTextField.RIGHT);
      }else if(e.getSource()==btnClose){
        System.exit(0);
      }
    }
  }
  public static void main(String[] args){
    ButtonDemo frm = new ButtonDemo();
    frm.pack();
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.setVisible(true);
  }
}
```

Kết quả của ButtonDemo.java

2) JLabel (nhãn):

Là vùng hiển thị một chuỗi ký tự ngắn, hình ảnh hoặc cả hai.

Phương thức tạo dựng:

```
public JLabel(String text)
```

- Tạo nhãn với chuỗi ký tự cho trước.

```
public JLabel(String text, int align)
```

- Tạo nhãn với chuỗi ký tự cho trước và được canh lề.

```
public JLabel(Icon icon)
```

- Tạo nhãn với biểu tượng.

Phương thức phương thức thường dùng trên JLabel

```
void setText(String text)
```

- Đặt lại nhãn

```
void setIcon(Icon icon)
```

- Đặt lại icon cho nhãn

Các hằng số canh lề:

- JLable.RIGHT
- JLable.LEFT
- JLable.CENTER.

3) JTextField (trường văn bản):

Thành phần cho phép nhập một dòng văn bản

Phương thức tạo dựng:

```
public JTextField(int columns)
```

- Tạo trường văn bản trống với số cột chỉ định

```
public JTextField(String text)
```

- Tạo trường văn bản với chuỗi định rõ

```
public JTextField(String text, int columns)
```

- Tạo trường văn bản với nội dung cho trước và số cột chỉ định

Một số phương thức thường dùng trên JTextField

```
void setText(String text)
```

- Đặt lại nội dung cho trường văn bản

```
String getText()
```

- Lấy về nội dung trong vùng văn bản

```
void setEditable(boolean state
```

- Cho phép người dùng có thể hiệu chỉnh hay không

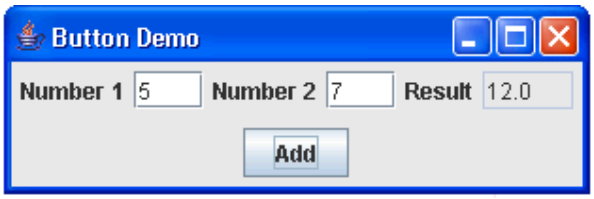
Ví dụ:

- Chương trình mẫu nhập hai trị số vào hai trường văn bản và hiển thị tổng của chúng ở trường thứ 3 khi nhấp nút Add.

```
package chapter04;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class TextFieldDemo extends JFrame{
```

```
private JButton btnAdd;
private JTextField txtNum1,txtNum2,txtResult;
public TextFieldDemo(){
  setTitle("Button Demo");
  Container c = getContentPane();
  JPanel p1 = new JPanel();
  p1.setLayout(new FlowLayout());
  p1.add(new JLabel("Number 1"));
  p1.add(txtNum1 = new JTextField(3));
  p1.add(new JLabel("Number 2"));
  p1.add(txtNum2 = new JTextField(3));
  p1.add(new JLabel("Result"));
  p1.add(txtResult = new JTextField(4));
  txtResult.setEditable(false);
  JPanel p2= new JPanel();
  p2.setLayout(new FlowLayout());
  p2.add(btnAdd= new JButton("Add"));
  c.add(p1,BorderLayout.CENTER);
  c.add(p2,BorderLayout.SOUTH);
  MyActionListener myAction = new MyActionListener();
  btnAdd.addActionListener(myAction);
}
class MyActionListener implements ActionListener{
  public void actionPerformed(ActionEvent e){
    if(e.getSource()==btnAdd){
      double num1 = Double.parseDouble(txtNum1.getText());
      double num2 = Double.parseDouble(txtNum2.getText());
      double result = num1 + num2;
      txtResult.setText(String.valueOf(result));
    }
  }
}
public static void main(String[] args){
  TextFieldDemo frm= new TextFieldDemo();
  frm.pack();
  frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frm.setVisible(true);
}
```

```
}
```



Kết quả của TextFieldDemo.java

4) JTextArea (Vùng văn bản):

Thành phần cho phép người dùng gõ nhiều dòng văn bản.

Phương thức tạo dựng:

```
public JTextArea(int rows, int cols)
```

- Tạo vùng văn bản với số dòng, cột xác định.

```
public JTextArea(String text, int rows, int cols)
```

- Tạo vùng văn bản với số dòng, cột chỉ định và nội dung cho trước

Một số phương thức thường dùng trên JTextArea

```
void setText(String text)
```

- Đặt lại nội dung

```
String getText()
```

- Lấy về nội dung

```
void setEditable(boolean b)
```

- Cho phép người dùng có thể chỉnh sửa văn bản hay không.

```
void setLineWrap(boolean b)
```

- Cho biết dòng chữ có tự động xuống dòng không.

```
String getSelectedText()
```

- Lấy về nội dung văn bản được đánh dấu chọn

```
void append(String s)
```

- Thêm chuỗi s vào cuối văn bản

```
void insert(String s,int pos)
```

- Chèn chuỗi s vào vị trí đã định trong vùng văn bản.

```
void replaceRange(String s, int start, int end)
```

- Thay thế phần văn bản từ vị trí start đến vị trí end bằng chuỗi s.

Lưu ý:

- Lớp JTextArea không xử lý thao tác cuốn, để xử lý thao tác cuốn cho JTextArea ta có thể tạo đối tượng JScrollPane chứa phiên bản của lớp JTextArea như sau:

```
JTextArea area = new JTextArea(10,30);
JScrollPane scrollPane = new JScrollPane(area);
```

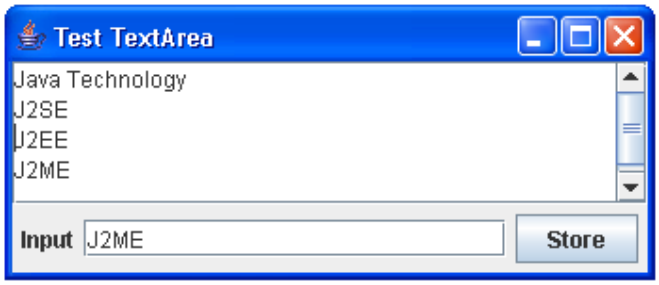
```
getContentPane().add(scrollPane, BorderLayout, CENTER);
```

Ví dụ:

- Chương trình mẫu cho phép người dùng nhập văn bản từ trường văn bản, sau đó ghi nối vào cuối nội dung ở vùng văn bản.

```
package chapter04;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TextAreaDemo extends JFrame{
  private JTextArea area;
  private JButton btnStore;
  private JTextField txtInput;
  public TextAreaDemo(){
    setTitle("Test TextArea");
    JPanel p =new JPanel();
    p.setLayout(new FlowLayout());
    p.add(new JLabel("Input"));
    p.add(txtInput= new JTextField(20));
    p.add(btnStore= new JButton("Store"));
    // create a scroll pane to hold text area
    JScrollPane scrollPane = new
                    JScrollPane(area = new JTextArea(5,20));
    area.setLineWrap(true);
    getContentPane().add(scrollPane,BorderLayout.CENTER);
    getContentPane().add(p,BorderLayout.SOUTH);
    // register listener
    MyActionListener myAction = new MyActionListener();
    txtInput.addActionListener(myAction);
    btnStore.addActionListener(myAction);
  }
  class MyActionListener implements ActionListener{
    public void actionPerformed(ActionEvent e){
      area.append(txtInput.getText().trim());
    }
  }
  public static void main(String args[]){
    TextAreaDemo frm = new TextAreaDemo();
```

```
        frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frm.pack();
        frm.setVisible(true);
    }
}
```



Kết quả của TextAreaDemo.java

5) *JComboBox (Hộp combo)*

Là danh sách liệt kê những khoản mục mà người dùng có thể chọn lựa.

Phương thức tạo dựng:

```
public  JComboBox()
```

- Tạo danh sách rỗng

Một số phương thức thường dùng trên JcomboBox:

```
void addItem(Object item)
```

- Thêm một đối tượng vào hộp combo

```
Object getItemAt(int index)
```

- Lấy về khoản mục từ hộp combo tại chỉ số đã định.

```
Object getSelectedItem()
```

- Lấy về khoản mục được chọn trên combo bởi người dùng.

```
void removeItem( Object item)
```

- Xoá một khoản mục khỏi danh sách.

```
void removeAllItems()
```

- Xoá toàn bộ các khoản mục khỏi danh sách.

```
int getItemCount()
```

- Trả về số khoản mục của danh sách.

6) *JList (danh sách):*

Là thành phần, về cơ bản, có cùng chức năng với hộp combo, nhưng nó cho phép người dùng chọn giá trị đơn lẻ hoặc nhiều giá trị.

Phương thức tạo dựng:

```
public JList(Object [])
```

- Tạo danh sách với mảng khoản mục chỉ định.

Một số phương thức thường dùng trên Jlist:

```
int getSelectedIndex()
```

- Trả về chỉ số của khoản mục được chọn trong danh sách.

```
int[] getSelectedIndices()
```

- Trả về mảng giá trị biểu thị chỉ số các khoản mục được chọn.

```
Object getSelectedValue()
```

- Trả về giá trị được chõn đầu tiên trong danh sách.

```
Object[] getSelectedValues()
```

- Trả về mảng đối tượng biểu thị các giá trị được chọn.

```
void setSelectionMode(int mode)
```

- Đặt chế độ chọn cho danh sách.

Lưu ý:

- Danh sách không tự động cuốn. Để danh sách cuốn được, ta dùng khung cuốn (scroll pane) và đặt danh sách vào đấy, tương tự như JTextArea.
- Danh sách JList phát sinh biến cố javax.swing.event.ListSelectionEvent để thông báo đến các đối tượng lắng nghe về khoản mục được chọn. Đối tượng lắng nghe phải thực thi phương thức valueChanged () hầu xử lý biến cố.

Ví dụ:

- Chương trình sau cho phép người dùng chọn tên sách trong danh sách và hiển thị ảnh bìa của sách đó trong nhãn.

```
package chapter04;
import javax.swing.*;
import java.awt.*;
import javax.swing.event.*;
public class ListDemo extends JFrame{
  private JList  lstBook;
  private JLabel lbBookImage;
  private ImageIcon bookImage[]= new ImageIcon[5];
  public ListDemo(){
    setTitle("List Demo");
    String bookName[]={"Java 2 Programming",
                       "Java Networking",
                       "Java with OOP",
                       "Jsp/Servlet Web Programing",
                       "Java 2 Research"};
    JScrollPane scrollPane = new JScrollPane(
                             lstBook = new JList(bookName));
```

```
    getContentPane().add(scrollPane,BorderLayout.WEST);
    for(int i=0;i<bookImage.length;i++){
    bookImage[i] = new ImageIcon(this.getClass().getResource(
                        "images/" + i + ".jpg"));
    }
    lbBookImage = new JLabel(bookImage[0]);
    getContentPane().add(lbBookImage,BorderLayout.CENTER);
    // register listeners
    lstBook.addListSelectionListener(new
                                    MySelectionListener());
  }
  class MySelectionListener implements ListSelectionListener{
    public void valueChanged(ListSelectionEvent e){
      lbBookImage.setIcon(bookImage[
                          lstBook.getSelectedIndex()]);
    }
  }
  public static void main(String args[]){
    ListDemo frm = new ListDemo();
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.pack();
    frm.setVisible(true);
  }
}
```

Kết quả của ListDemo.java

7) JCheckBox và JradioButton:

a) JCheckBox (ô chọn):

- Thành phần cho phép người dùng chọn hoặc xóa chọn một tùy chọn.
- Hình dáng: ☑

Phương thức tạo dựng:

```
public JCheckBox(String text)
```

- Tạo ô chọn với chuỗi văn bản chỉ định.

```
public JCheckBox(String text, boolean selected)
```

- Tạo ô chọn với chuỗi văn bản chỉ định và định rõ trạng thái ban đầu của ô chọn.

Một số phương thức thường dùng trên JcheckBox:

```
void setText(String text)
```

- Đặt lại nhãn cho ô chọn

```
void setIcon( Icon icon)
```

- Đặt lại biểu tượng cho ô chọn

```
void setVisible(boolean b)
```

- Cho ô chọn ẩn hoặc hiện

```
public boolean isSelected()
```

- Trả về trạng thái ô chọn

b) JRadioButton (nút radio)

- Là thành phần tương tự như JcheckBox.
- Hình dáng: 

Phương thức tạo dựng:

```
public JRadioButton(String)
public JRadioButton(String, boolean)
```

- Muốn nhóm các nút radio lại với nhau, ta cần tạo đối tượng javax.swing.ButtonGroup và đặt nút radio vào đối tượng bằng phương thức add (), chẳng hạn như đoạn mã minh họa sau:

```
JRadioButton jrd1 = new JRadioButton("Male",true);
JRadioButton jrd2 = new JRadioButton("Female");
ButtonGroup  btg= new ButtonGroup();
btg.add(jrb1);
btg.add(jrb1);
```

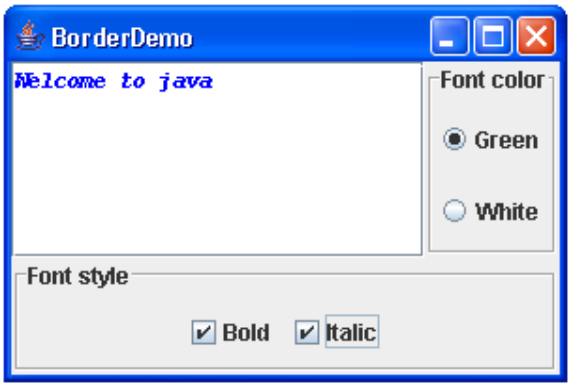
Ví dụ:

- Chương trình mẫu cho phép định dạng chuỗi ký tự trong vùng văn bản, người dùng có thể chọn in đậm và in nghiêng cùng lúc. Cũng có thể chọn màu chữ cho vùng văn bản.

```
package chapter04;
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
import java.awt.event.*;
public class BorderDemo extends JFrame{
```

```
private JCheckBox chkBold,chkItalic;
private JRadioButton jrdBlue, jrdBlack;
private JTextArea area;
public BorderDemo(){
  setTitle("BorderDemo");
  Container c = getContentPane();
  JPanel p1 = new JPanel();
  p1.setLayout(new FlowLayout());
  // set TitledBorder for panel p1
  p1.setBorder(new TitledBorder(new EtchedBorder(),
                                "Font style"));
  p1.add(chkBold= new JCheckBox("Bold"));
  p1.add(chkItalic= new JCheckBox("Italic"));
  JPanel p2 = new JPanel();
  p2.setLayout(new GridLayout(2,1));
  // set TitledBorder to panel p2
  p2.setBorder(new TitledBorder(new EtchedBorder(),
                                "Font color"));
  p2.add(jrdBlue= new JRadioButton("Green"));
  p2.add(jrdBlack= new JRadioButton("White"));
  ButtonGroup btg =new ButtonGroup();
  btg.add(jrdBlue);
  btg.add(jrdBlack);
  c.add(p1,BorderLayout.SOUTH);
  c.add(p2,BorderLayout.EAST);
  JScrollPane scrollPane = new JScrollPane(area = new
                           JTextArea("Welcome to java"));
  c.add(scrollPane,BorderLayout.CENTER);
  // register listener
  MyItemListener myAction = new MyItemListener();
  chkBold.addItemListener(myAction);
  chkItalic.addItemListener(myAction);
  jrdBlue.addItemListener(myAction);
  jrdBlack.addItemListener(myAction);
}
class MyItemListener implements ItemListener{
  public void itemStateChanged(ItemEvent e){
    int selectedStyle=0;
```

```
      if(chkBold.isSelected())
        selectedStyle = selectedStyle + Font.BOLD;
      if(chkItalic.isSelected())
        selectedStyle = selectedStyle + Font.ITALIC;
      area.setFont(new Font("Monospaced",selectedStyle,12));
      if(jrdBlue.isSelected())
        area.setForeground(Color.blue);
      else
        area.setForeground(Color.BLACK);
    }
  }
  public static void main(String[] args){
    BorderDemo frm= new BorderDemo();
    frm.setSize(300,200);
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.setVisible(true);
  }
}
```



Kết quả của JCheckBoxRadioButtonDemo.java

Border (khung viền):

- Là một trong những đặc tính mới thú vụ của thành phần swing (được chứa trong gói javax.swing.border).
- Ta thường thiết lập khung viền có tiêu đề trên lớp JPanel, ghép nhóm tập hợp thành phần giao diện người dùng có liên quan nhau.
- Để đặt khung viền có tiêu đề cho panel, ta thường dùng lệnh sau:

```
JPanel  p = new JPanel();
p.setBorder(new TitledBorder(new EtchedBorder(),"Tiêu đề"));
```

8) JOptionPane (Hộp thoại thông báo):

- Thường được dùng làm cửa sổ tạm để nhận thêm thông tin từ người dùng, hoặc thông báo về những biến cố đã xảy ra

Các thành phần của cửa sổ dạng hộp thoại thông báo là:

Cú pháp:

```
public showMessageDialog(Component ParentComponent,
                         Object Message,
                         String Title,
                         int MessageType)
```

- **ParentComponet:** là thành phần cha của hộp thoại, nơi hộp thoại khởi động.
- **Message:** là đối tượng cần hiển thị
- **Title:** là tiêu đề hộp thoại
- **Messagetype:** là kiểu thông điệp hiển thị.

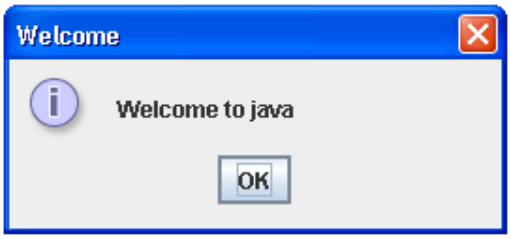
Có 5 kiểu thông điệp:

```
ERROR_MESSAGE
INFORMATION_MESSAGE
PLAIN_MESSAGE
WARNING_MESSAGE
QUESTION_MESSAGE
```

Ví dụ:

- Câu lệnh sau hiển thị một thông điệp "Welcome to java".

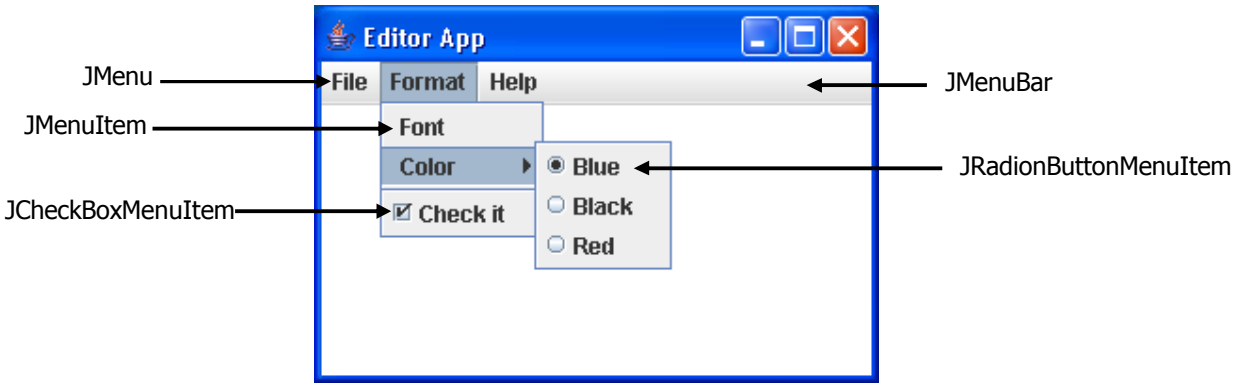
```
JOptionPane.showMessageDialog(null,
                         "Welcome to java",
                         "Welcome",
                         JOptionPane.INFORMATION_MESSAGE);
```



9) Menu (thực đơn):

Menu giúp chọn lựa dễ dàng và được sử dụng rộng rãi trong các chương trình ứng dụng dạng cửa sổ.

Java cung cấp 5 lớp: JMenuBar, JMenu, JMenuItem, JCheckBoxMenuItem và JRadioButtonMenuItem để sử dụng tạo menu.



Trình tự ứng dụng menu trong java:

- Tạo thanh menu và phối hợp nó với khung bất kỳ:

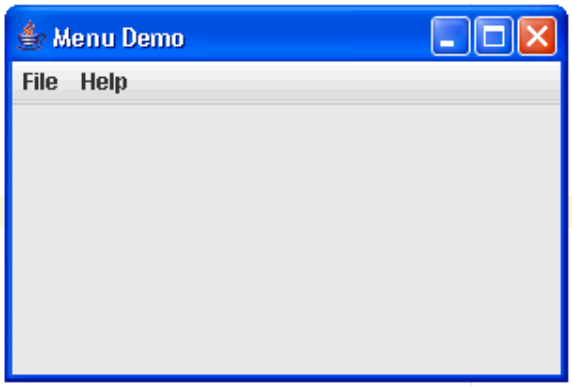
```
JFrame  frm = new JFrame("Menu Demo");
JMenuBar jmb = new JMenuBar();
frm.setJMenuBar(jmb);
```

- Tạo menu: Sử dụng phương thức tạo dựng sau đây:

```
public  JMenu(String label)
```

Ví dụ:

```
JMenu  fileMenu = new JMenu("File");
JMenu  helpMenu = new JMenu("Help");
jmb.add(fileMenu);
jmb.add(helpMenu);
```

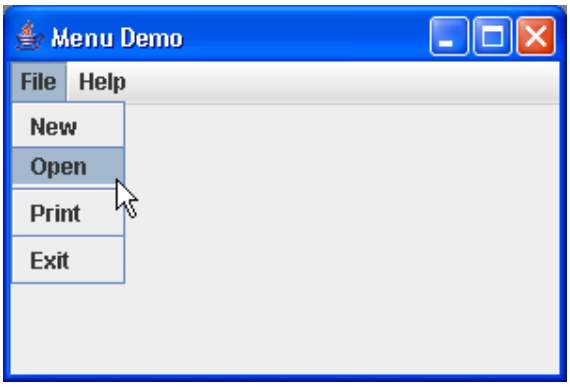


Tạo các khoản mục và chèn vào menu:

```
fileMenu.add(new JMenuItem("New"));
fileMenu.add(new JMenuItem("Open"));
```

```
fileMenu.addSeparator(); // Tạo khoảng cách trong menu
fileMenu.add(new JMenuItem("Print"));
fileMenu.addSeparator(); // Tạo khoảng cách trong menu

fileMenu.add(new JMenuItem("Exit"));
```

Đoạn mã này lần lượt bổ sung các khoản mục New, Open, vạch phân cách, Print, vạch phân cách và Exit vào menu File.

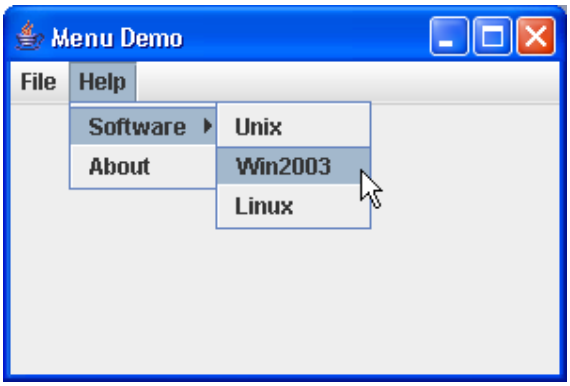


Tạo các khoản mục trên menu con.

- Ta cũng có thể nhúng menu bên trong menu, lúc đó menu được nhúng trở thành menu con.

```
JMenu softwareHelpSubMenu  = new JMenu("Software");
helpMenu.add(softwareHelpSubMenu);
helpMenu.add(new JMenuItem("About"));
softwareHelpSubMenu.add(new JMenuItem("Unix"));
softwareHelpSubMenu.add(new JMenuItem("Win2003"));
softwareHelpSubMenu.add(new JMenuItem("Linux"));
```

Đoạn mã này thêm menu softwareHelpSubMenu và khoản mục About vào trong helpMenu. Các khoản mục Unix, Win2003 và Linux vào menu softwareHelpSubMenu.

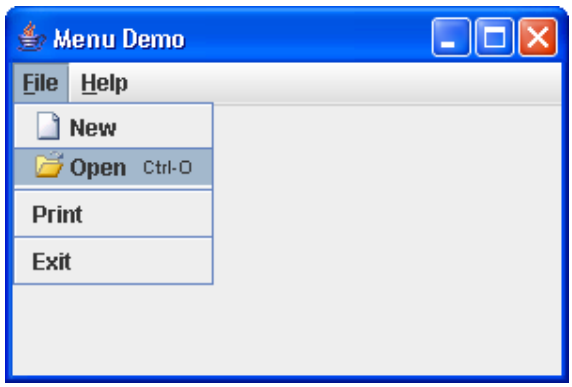


10) Biểu tượng ảnh, phím tắt quy ước và phím tăng tốc.

Các thành phần JMenu, JMenuItem, JCheckBoxMenuItem, JRadioButtonMenuItem đều có phương thức setIcon (Icon icon) và setMnemonic (char c) để đặt biểu tượng và phím tắt quy ước cần thiết.

Ví dụ:

```
JMenuItem jmiNew,jmiOpen;
fileMenu.add(jmiNew=new JMenuItem("New"));
fileMenu.add(jmiOpen=new JMenuItem("Open"));
// đặt biểu tượng cho khoản mục New và Open
jmiNew.setIcon(new
ImageIcon(getClass().getResource("images/new.gif")));
jmiOpen.setIcon(new
ImageIcon(getClass().getResource("images/open.gif")));
// đặt phím tắt qui ước cho menu File và Help
fileMenu.setMnemonic('F');
helpMenu.setMnemonic('H');
```



- Muốn chọn menu, nhấn ALT và phím tắt quy ước. Chẳng hạn, tổ hợp ALT+ F sẽ chọn menu File.

Ngoài ra ta có thể dùng phím tăng tốc cho phép chọn trực tiếp khoản mục bằng cách nhấn phím Ctrl và phím tăng tốc.

Ví dụ:

- Tạo tổ hợp phím Ctrl – O để thực hiện khoản mục Open.

```
jmiOpen.setAccelerator(KeyStroke.getKeyStroke(
                    KeyEvent.VK_O,ActionEvent.CTRL_MASK));
```

- Phương thức setAccelerator () đưa ra đối tượng KeyStroke.
- Phương thức lớp getKeyStroke () của lớp KeyStroke tạo phiên bản của cú gõ phím. VK_O là hằng biểu thị phím O, CTRL_MASK là hằng cho biết phím Ctrl được phối hợp với phím gõ.

Ví dụ:

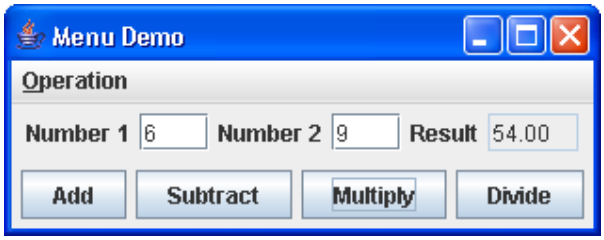
- Chương trình mẫu tạo giao diện người dùng thực hiện các phép toán số học. Giao diện chứa nhãn và trường văn bản cho Number 1, Number 2 và Result.
- Trường Result hiển thị kết quả của phép toán số học giữa Number 1 và Number 2.
- Chương trình có 4 nút Add, Subtract, Multiply, Divide
- Ngoài ra còn tạo menu thực hiện cùng phép toán. Người dùng có thể chọn phép toán từ nút hoặc từ menu.

```
package chapter04;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.text.DecimalFormat;
public class MenuDemo extends JFrame{
  private JButton btnAdd,btnSub,btnMul, btnDiv;
  private JTextField txtNum1,txtNum2,txtResult;
  private JMenuItem jmiAdd, jmiSub, jmiMul,jmiDiv, jmiClose;
  private DecimalFormat fmt = new DecimalFormat("0.00");
  public MenuDemo(){
    setTitle("Menu Demo");
    // create menu bar
    JMenuBar  jmb= new JMenuBar();
    // set menu bar to the frame
    setJMenuBar(jmb);
    //add menu "Operation" to menu bar
    JMenu operMenu = new JMenu("Operation");
    operMenu.setMnemonic('O');
    jmb.add(operMenu);
    //add menu items with mnemonics to menu "Operation"
    operMenu.add(jmiAdd= new JMenuItem("Add",'A'));
    operMenu.add(jmiSub= new JMenuItem("Subtract",'S'));
    operMenu.add(jmiMul= new JMenuItem("Multiply",'M'));
    operMenu.add(jmiDiv= new JMenuItem("Divide",'D'));
    operMenu.addSeparator();
    operMenu.add(jmiClose= new JMenuItem("Close"));
    //set keyboard accelerators
    jmiAdd.setAccelerator(KeyStroke.getKeyStroke(
                      KeyEvent.VK_A,ActionEvent.CTRL_MASK));
    jmiSub.setAccelerator(KeyStroke.getKeyStroke(
```

```
                    KeyEvent.VK_S,ActionEvent.CTRL_MASK));
  jmiMul.setAccelerator(KeyStroke.getKeyStroke(
                    KeyEvent.VK_M,ActionEvent.CTRL_MASK));
  jmiDiv.setAccelerator(KeyStroke.getKeyStroke(
                    KeyEvent.VK_D,ActionEvent.CTRL_MASK));
  Container c = getContentPane();
  JPanel p1 = new JPanel();
  p1.setLayout(new FlowLayout());
  p1.add(new JLabel("Number 1"));
  p1.add(txtNum1 = new JTextField(3));
  p1.add(new JLabel("Number 2"));
  p1.add(txtNum2 = new JTextField(3));
  p1.add(new JLabel("Result"));
  p1.add(txtResult = new JTextField(4));
  txtResult.setEditable(false);
  JPanel p2= new JPanel();
  p2.setLayout(new FlowLayout());
  p2.add(btnAdd= new JButton("Add"));
  p2.add(btnSub= new JButton("Subtract"));
  p2.add(btnMul= new JButton("Multiply"));
  p2.add(btnDiv= new JButton("Divide"));
  c.add(p1,BorderLayout.CENTER);
  c.add(p2,BorderLayout.SOUTH);
  //register listener
  MyActionListener myAction = new MyActionListener();
  btnAdd.addActionListener(myAction);
  btnSub.addActionListener(myAction);
  btnMul.addActionListener(myAction);
  btnDiv.addActionListener(myAction);
  jmiAdd.addActionListener(myAction);
  jmiSub.addActionListener(myAction);
  jmiMul.addActionListener(myAction);
  jmiDiv.addActionListener(myAction);
  jmiClose.addActionListener(myAction);
}
class MyActionListener implements ActionListener{
  public void actionPerformed(ActionEvent e){
    String actionCommand = e.getActionCommand();
```

```
    if(e.getSource() instanceof JButton){
      if(actionCommand.equals("Add"))
        calculator('+');
      else if(actionCommand.equals("Subtract"))
        calculator('-');
      else if(actionCommand.equals("Multiply"))
        calculator('*');
      else if(actionCommand.equals("Divide"))
        calculator('/');
    }
    else if(e.getSource() instanceof JMenuItem){
      if(actionCommand.equals("Add"))
        calculator('+');
      else if(actionCommand.equals("Subtract"))
        calculator('-');
      else if(actionCommand.equals("Multiply"))
        calculator('*');
      else if(actionCommand.equals("Divide"))
        calculator('/');
      else if(actionCommand.equals("Close"))
        System.exit(0);
    }
  }
}
private void calculator(char operator){
  double num1 = Double.parseDouble(txtNum1.getText());
  double num2 = Double.parseDouble(txtNum2.getText());
  double result = 0;
  switch(operator){
    case '+': result=num1+num2;
              break;
    case '-': result=num1-num2;
              break;
    case '*': result=num1*num2;
              break;
    case '/': result=num1/num2;
              break;
  }
```

```
    txtResult.setText(fmt.format(result));
  }
  public static void main(String[] args){
    MenuDemo frm = new MenuDemo();
    frm.pack();
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.setVisible(true);
  }
}
```



Kết xuất của MenuDemo.java

11)JTabbedPane (Khung nhiều tab):

- Là thành phần Swing hữu ích, cung cấp tập hợp tab loại trừ nhau để truy cập nhiều thành phần.
- Ta thường đặt panel bên trong một JTabbedPane và phối hợp tab với mỗi panel. JTabbedPanel rất dễ sử dụng, vì panel tự động được chọn bằng cách nhấp tab tương ứng.

Phương thức khởi tạo:

```
public  JTabbedPane()
```

- Tạo khung trống có nhiều tab.

Muốn bổ sung thành phần vào JTabbedPane, sử dụng phương thức:

```
public add(Component component, Object constraints)
```

- Component là thành phần hiển thị khi tab được nhấp
- Contraints có thể là tiêu đề của tab.

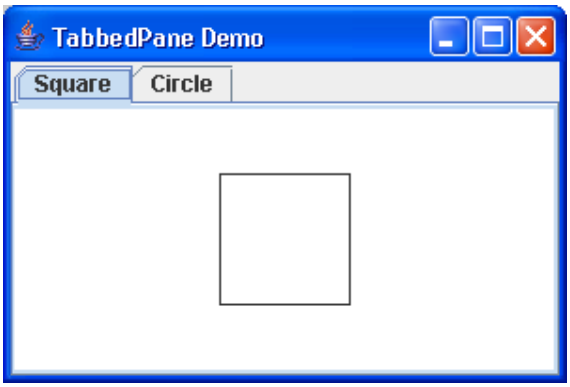
Ví dụ:

- Chương trình dùng nhiều khung tab với 2 tab hiển thị 2 dạng hình Square (hình vuông), Circle (hình tròn).
- Chọn hình cần hiển thị bằng cách nhấp tab tương ứng.

```
package chapter04;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class TabbedPaneDemo extends JFrame{
```

```
private JTabbedPane jtpFigure = new JTabbedPane();
public TabbedPaneDemo(){
  setTitle("TabbedPane Demo");
  // add FigurePanel to the tab
  jtpFigure.add(new FigurePanel(
               FigurePanel.SQUARE),"Square");
  jtpFigure.add(new FigurePanel(
               FigurePanel.CIRCLE),"Circle");
  getContentPane().add(jtpFigure);
}
class FigurePanel extends JPanel{
  final static int SQUARE =1;
  final static int CIRCLE =2;
  private int figure=1;
  public FigurePanel(int fig){
    figure =fig;
    this.setBackground(Color.white);
  }
  //drawing a figure on the panel
  public void paintComponent(Graphics g){
    super.paintComponent(g);
    int width= getSize().width;
    int height= getSize().height;
    int side =(int)(0.5*Math.min(width,height));
    switch(figure){
      case 1: g.drawRect((width-side)/2,
                         (height-side)/2,side,side);
              break;
      case 2: g.drawOval((width-side)/2,
                         (height-side)/2,side,side);
              break;
    }
  }
} //end inner class
public static void main(String args[]){
  TabbedPaneDemo frm = new TabbedPaneDemo();
  frm.setSize(300,200);
  frm.setVisible(true);
```

```
    }
}
```



Kết xuất của TabbedPaneDemo.java

Chương 5 XỬ LÝ BIỆT LỆ

Sau bài học này, học viên có thể:

- Hiểu được biệt lệ và xác định được các dạng biệt lệ trong lập trình java.
- Có khả năng phân tích và xử lý các ngoại lệ.
- Hiểu được các phương pháp xử lý ngoại lệ.

I. KHÁI NIỆM:

- Biệt lệ (Exception) là lỗi xảy ra trong thời gian thực thi chương trình (runtime error), làm gián đoạn mạch điều khiển bình thường. Thông thường các điều kiện thực thi chương trình gây ra biệt lệ. Nếu các điều kiện này không được quan tâm, thì việc thực thi có thể kết thúc đột ngột.
- Java cung cấp khả năng xử lý lỗi thi hành cho các lập trình viên. Với khả năng này, gọi là khả năng xử lý biệt lệ, ta có thể thiết kế chương trình ổn định và tin cậy hơn khi thi hành. Giảm thiểu việc kết thúc bất thường của hệ thống và của chương trình.

II. CÁC DẠNG BIỆT LỆ:

Các biệt lệ thường gặp:

- **ArithmeticException:** lỗi do tính toán, thường là chia cho 0.
- **ArrayIndexOutOfBoundsException:** lỗi do truy xuất chỉ số các phần tử của mảng.
- **NullPointerException:** lỗi do tham chiếu đối tượng có giá trị null.
- **ClassNotFoundException:** lỗi do không tìm thấy lớp.
- **NumberFormatException:** Lỗi khi chuyển đổi kiểu.
- **IOException:** lỗi nhập xuất.
- **FileNotFoundException:** lỗi truy cập tin tin không tồn tại.
- **InterruptedException:** Lỗi liên quan đến trì hoãn tuyến trình.
- **SQLException:** lỗi khi thực thi câu lệnh SQL với cơ sở dữ liệu.

Tất cả các biệt lệ của Java đều là dẫn xuất từ lớp Exception. Trong lớp Exception có 3 phương thức rất thông dụng cho ta biết được thông tin chi tiết của từng biệt lệ:

```
public String getMessage()
```

- Nhận về một chuỗi là thông tin chi tiết của biệt lệ.

```
public String toString()
```

- Chuyển đối tượng thành chuỗi để có thể in ra màn hình.

```
public void printStackTrace()
```

- Trình bày hệ thống việc gọi các phương thức cho phép dò tìm biệt lệ.

III. XỬ LÝ BIỆT LỆ:

Để xử lý biệt lệ, ta có thể sử dụng 2 phương pháp:

- Bắt biệt lệ.
- Ném biệt lệ.

1) Bắt biệt lệ:

Cú pháp:

```
try{
   // Các lệnh có khả năng xảy ra biệt dịch
}
catch(ExceptionType1 param1){
  // Đoạn lệnh xử lý tương ứng với biệt lệ xảy ra
}
catch(ExceptionType2 param2){
  // Đoạn lệnh xử lý tương ứng với biệt lệ xảy ra
}
...
catch(ExceptionTypeN paramN){
  // Exception Block
}
finally{
  // Đoạn lệnh luôn thực hiện bất kể biệt lệ có xảy
     ra hay không.
}
```

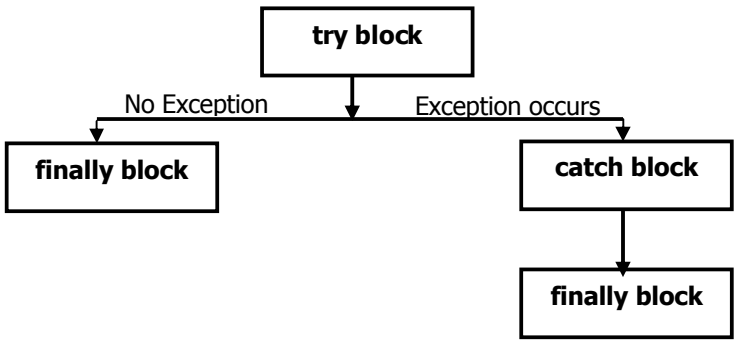
Cách thực hiện của khối try-catch:

- Nếu không có biệt lệ nào xảy ra trong suốt quá trình thực thi mệnh đề try, các mệnh đề catch sẽ được bỏ qua.
- Trường hợp một trong các câu lệnh bên trong khối try ném ra một biệt lệ, Java sẽ bỏ qua các câu lệnh còn lại và bắt đầu tìm kiếm phương thức xử lý biệt lệ đó.
  - *Nếu kiểu biệt lệ so khớp với biệt lệ liệt kê trong mệnh đề catch, mã trong mệnh đề catch sẽ được thực thi.*
  - *Nếu không tìm thấy phương thức xử lý, chương trình sẽ chấm dứt và in ra thông điệp báo lỗi trên console.*

Khối finally:

- Là tùy chọn không bắt buộc
- Được đặt sau khối catch.
- Khối finally bảo đảm lúc nào cũng được thực hiện bất chấp biệt lệ có xảy ra hay không.

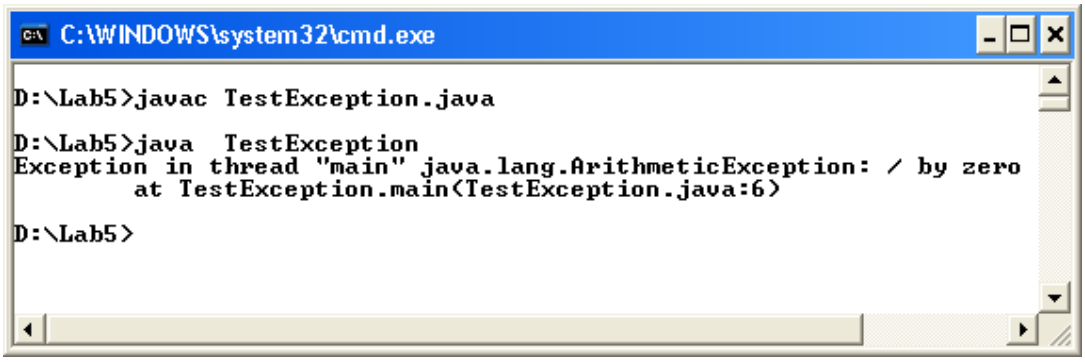
Sơ đồ hoạt động:



Ví dụ:

- Chương trình sau thực hiện phép toán chia x cho y. Biệt lệ sẽ xảy ra khi y=0. Chương trình không quan tâm tới biệt lệ.

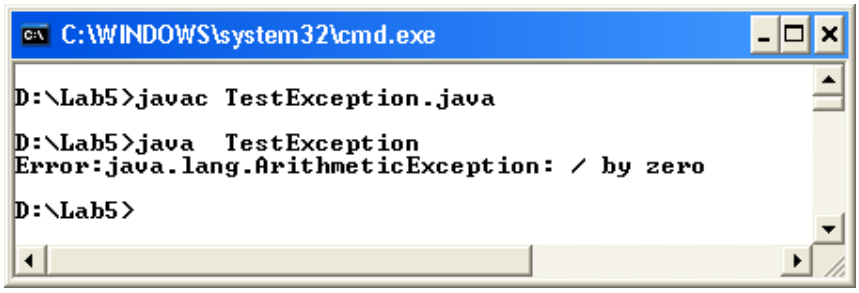
```
public class TestException{
  public static void main(String args[]){
    int x=5,y=0;
    int  z=x/y;
    System.out.println("z=" + z);
  }
}
```



Kết quả của TestException. Java

- Chương trình sau thực hiện phép toán chia x cho y. Biệt lệ sẽ xãy ra khi y=0. Chương trình có quan tâm tới biệt lệ và bắt ngoại lệ nếu nó xảy ra.

```
public class TestException{
  public static void main(String args[]){
    int x=5,y=0;
    try{
      int  z=x/y;
      System.out.println("z=" + z);
    }
    catch(Exception ex){
      System.out.println("Error:"+ ex.toString());
    }
  }
}
```



Kết quả của TestException. Java

2) Ném biệt lệ:

Cú pháp:

```
[modifier] <returnType> <methodName>([params]) throws
                                                ExceptionName{
  // method body
}
```

Ví dụ:

```
class ThrowsDemo{
  public static void main(String args[]){
    a();
  }
  public static void a(){
    try{
      b();
    }
    catch(ClassNotFoundException e){
      e.printStackTrace();
    }
  }
  public static void b() throws ClassNotFoundException{
    c();
  }
  public static void c() throws ClassNotFoundException{
    Class cls = Class.forName("java.lang.Integer");
    System.out.println(cls.getName());
    System.out.println(cls.isInterface());
  }
}
```

Kết quả của ThrowsDemo.java

Chương 6 XỬ LÝ LUỒNG VÀ TẬP TIN

Sau bài học này, học viên có thể:

- Sử dụng lớp File để quản lý tập tin và thư mục trên hệ thống.
- Hiểu và sử dụng luồng nhập xuất (I/O) dữ liệu dạng byte và dạng ký tự.

I. LỚP FILE VÀ CÁCH SỬ DỤNG:

Lớp java.io.File giúp biết được chi tiết về tập tin, ngày giờ tập tin được tạo ra, kích thước của tập tin, vị trí lưu trên đĩa...

Ta cũng có thể dùng lớp File để tạo thư mục, đổi tên, xóa tập tin...

Để tạo đối tượng File tham chiếu đến tập tin hay thư mục trên hệ thống, ta dùng phương thức khởi tạo:

```
public  File(String pathName)
public  File(String parent, String child)
```

Ví dụ:

```
File file1 = new File("data.txt");
File file2 = new File("c:/java");
```

Để lấy được đường dẫn thư mục làm việc hiện hành, ta dùng lệnh:

```
System.getProperty("user.dir");
```

Một số phương thức thường dùng trên File

```
public boolean isFile()
```

- Trả về true nếu đối tượng là tập tin.

```
public boolean isDirectory()
```

- Trả về true nếu đối tượng là thư mục.

```
public boolean exists()
```

- Trả về true nếu đối tượng file tồn tại.

```
public long length()
```

- Trả về kích thước của tập tin.

```
public String[] list()
```

- Nếu đối tượng là File là thư mục, phương thức trả về mảng String của tất cả các tập tin và thư mục chứa trong thư mục, ngược lại, trả về null.

```
public boolean mkdir()
```

- Tạo thư mục con

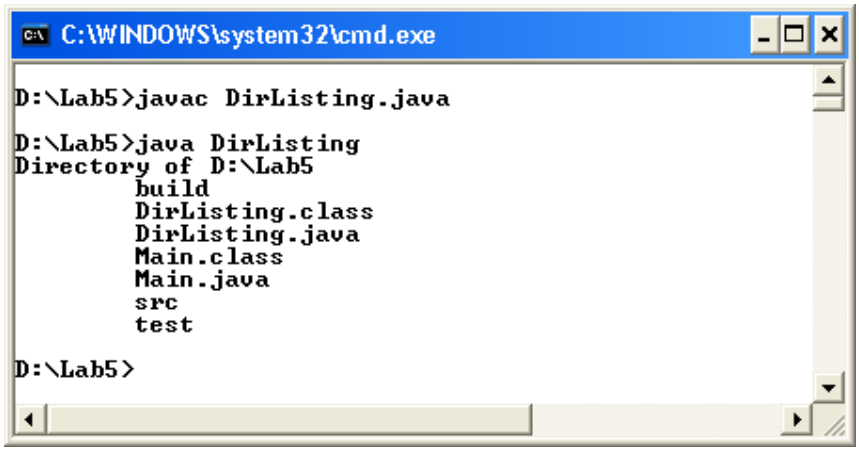
```
public boolean delete()
```

- Xóa thư mục, trả về true nếu thành công.

Ví dụ:

- Chương trình mẫu sau liệt kê tập tin và thư mục chứa trong thư mục làm việc hiện hành.

```
import java.io.*;
public class DirListing{
  public static void main(String args[]){
    File dir = new File(System.getProperty("user.dir"));
    if(dir.isDirectory()){
      System.out.println("Directory of " + dir);
      String listing[] = dir.list();
      for(int i=0;i<listing.length;i++)
        System.out.println("\t" +listing[i]);
    }
  }
}
```



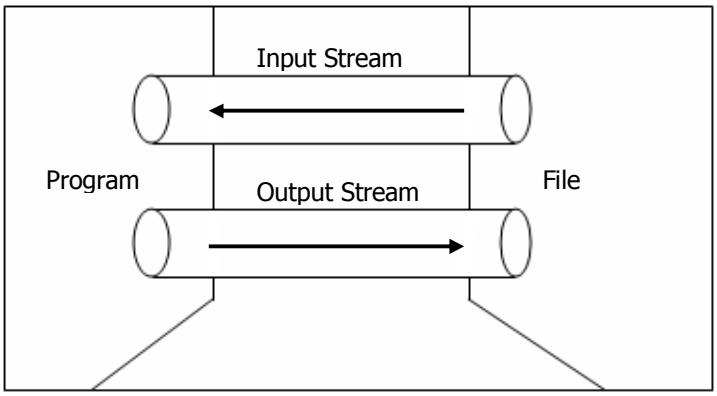
Kết xuất của DirListing. java

II. LUỒNG (STREAM):

1) Khái niệm:

- Trong Java, mọi I/O đều được xử lý theo luồng. Luồng (stream) là khái niệm về một luồng lưu chuyển dữ liệu một chiều liên tục.
- Hãy hình dung một bể bơi có các ống nước thông với bể khác. Xem nước trong bể bơi đầu là dữ liệu, còn nước ở bể bơi kia là chương trình. Lưu lượng nước chảy qua ống được gọi là luồng. Nếu muốn nhập, chỉ cần mở van để nước chảy ra khỏi bể dữ liệu vào bể chương trình. Muốn xuất, mở van để nước chảy khỏi bể chương trình vào bể dữ liệu.
- Khối thư viện java.io cung cấp các luồng nhập xuất khác nhau phục vụ cho khả năng đọc và ghi dữ liệu. Luồng nhập xuất không những được kết nối với tập tin mà còn

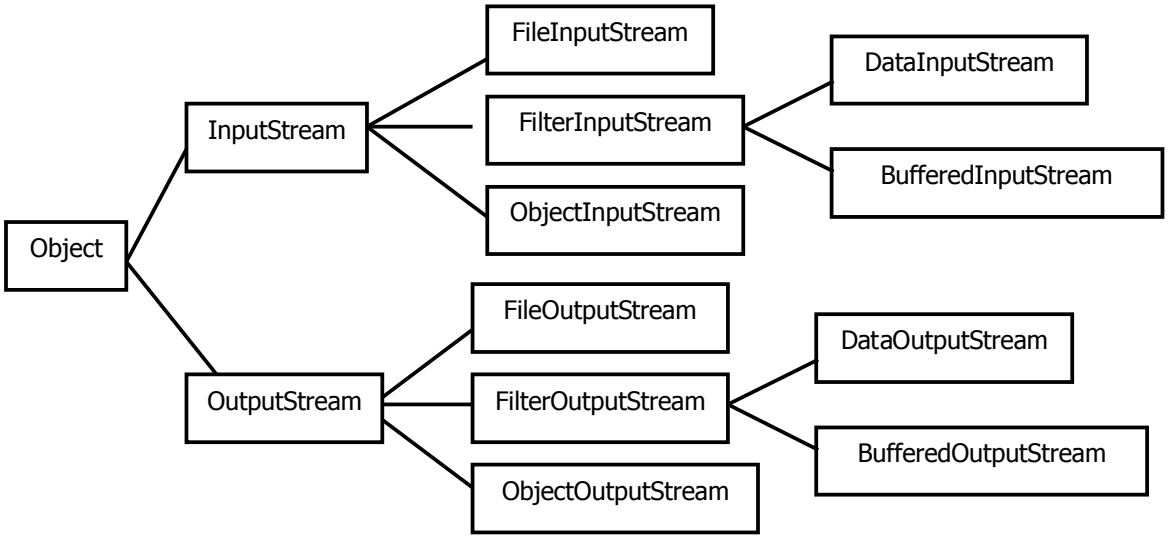
được dùng trong việc kết nối mạng, hay các vùng đệm của bộ nhớ máy tính giúp cho việc truy xuất được nhanh.



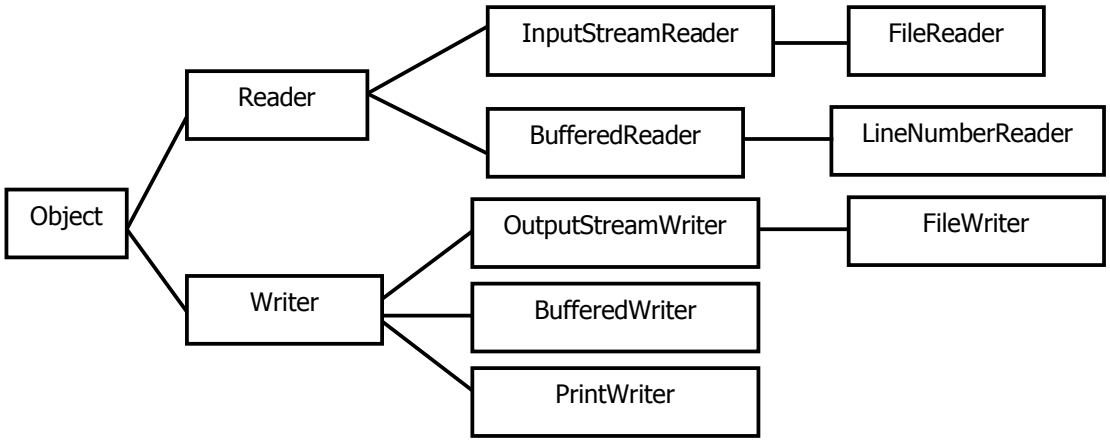
Chương trình nhận dữ liệu qua luồng nhập và gói dữ liệu qua luồng xuất

Có thể xếp lớp luồng vào hai loại: luồng byte (byte stream) và luồng ký tự (character stream).

- Mối quan hệ phân cấp của luồng byte thường dùng:



- Mối quan hệ phân cấp của luồng ký tự thường dùng:



2) Luồng nhập xuất cơ bản:

a) Lớp InputStream và lớp Reader:

- Lớp trừu tượng InputStream và Reader, mở rộng Object, lần lượt là lớp gốc (cha) của mọi luồng nhập byte và ký tự.
- Hai lớp này và lớp con của chúng rất giống nhau, ngoại trừ lớp InputStream dùng byte làm đơn vị thông tin cơ bản, còn Reader sử dụng ký tự.

Một số phương thức thường dùng trong InputStream:

```
public int read() throws IOException
```

- Đọc byte tiếp theo và trả về giá trị của nó. Giá trị của byte được trả về ở dạng int, cuối luồng phương thức trả về -1.

```
public int read(byte b[]) throws IOException
```

- Đọc các byte b.length thành mảng b, giá trị trả về là số byte đọc được thật sự. Cuối luồng trả về -1.

```
public void close() throws IOException
```

- Đóng luồng nhập.

```
public int available() throws IOException
```

- Trả về số byte còn lại trong luồng.

```
public long skip(long n) throws IOException
```

Bỏ qua và loại bỏ n byte dữ liệu khỏi luồng nhập này. Số byte thật sự bị bỏ qua sẽ được trả về.

Lớp Reader chứa tất cả phương thức vừa liệt kê, ngoại trừ avalible ().

b) Lớp OutputStream và lớp Writer:

- Cả hai OutputStream và Writer lần lượt là lớp gốc (cha) của tất cả luồng xuất byte và ký tự.

Một số phương thức thường dùng trong OuputStream lẫn Writer:

```
public void write(int b) throws IOException
```

- Ghi một byte (đối với OutputStream) hay một ký tự (với Writer)

```
public void write(byte b[]) throws IOException
```

- Ghi mọi byte trong mảng b sang luồng xuất (OutputStream) hoặc mọi ký tự trong mảng ký tự (Writer)

```
public void close() throws IOException
```

- Đóng luồng nhập.

```
public void flush() throws IOException
```

- Dồn luồng xuất (có nghĩa gởi dữ liệu lưu tạm trong luồng xuất đến đích của nó).

3) Xử lý tập tin ngoại trú:

Phải dùng luồng tập tin để đọc hoặc ghi vào tập tin đĩa. Ta có thể sử dụng:

- FileInputStream hay FileOutputStream cho luồng byte
- FileReader hay FileWriter cho luồng ký tự.

Phương thức khởi tạo sau:

```
public FileInputStream (String fileNameString)
public FileOutputStream (String fileNameString)
public FileReader (String fileNameString)
public FileWrite (String fileNameString)
```

Ví dụ:

```
FileInputStream infile = new FileInputStream
                                ("c:\data\in.dat");
FileOutStream outfile = new FileOutStream
                               ("c:\data\out.dat");
```

Ta cũng có thể sử dụng đối tượng tập tin tạo luồng tập tin như câu lệnh sau:

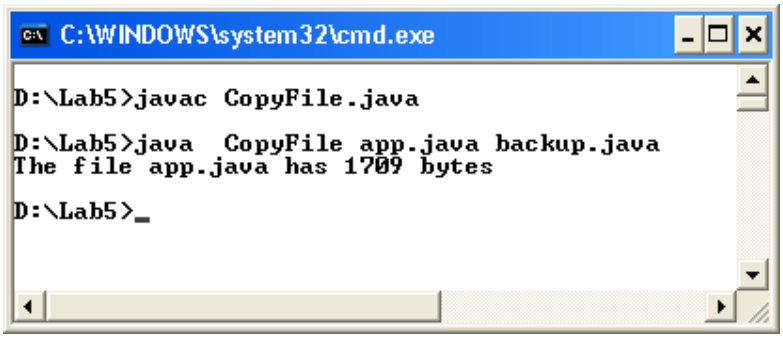
```
FileInputStream infile = new FileInputStream(new File
                                ("c:\data\in.dat"));
```

Ví dụ:

- Chương trình minh họa sử dụng FileInputStream và FileOutputStream sao chép tập tin.
- Người dùng cần cung cấp một tập tin nguồn và tập tin đích làm đối số dòng lệnh.

```
import java.io.*;
public class CopyFile {
  public static void  main (String args[]){
    FileInputStream fis=null;
    FileOutputStream fos=null;
    if(args.length!=2){
      System.out.println("Syntax is:
                        java CopyFile sourcefile destfile");
      System.exit (0);
    }
    try{
      fis = new FileInputStream (args[0]);
      File outFile = new File (args[1]);
      if(outFile.exists()){
        System.out.println("File " + args[1]
                          + " already exists");
```

```
        return ;
      }
      else
        fos = new FileOutputStream(outFile);
        // Display the file size
      System.out.println("The file " + args[0] + " has " +
fis.available() + " bytes");
      int r;
      while((r=fis.read())!=-1)
        fos.write((byte)r);
    }
    catch(FileNotFoundException ex){
      System.out.println("File not found " + args[0]);
    }
    catch(IOException ex){
      System.out.println(ex.getMessage());
    }
    finally{
      try{
        if(fis!=null) fis.close();
        if(fos!=null) fos.close();
      }
      catch(IOException ex){
        System.out.println(ex.getMessage());
      }
    }
  }
}
```



Kết quả của CopyFile.java

4) Luồng lọc:

- Luồng lọc (filter stream) được định nghĩa là luồng lọc các byte hay ký tự nhằm mục đích nào đó. Luồng nhập cơ bản cung cấp phương thức đọc vốn chỉ có thể dùng để đọc byte hay ký tự.
- Nếu muốn đọc số nguyên, số kép, chuỗi, ta có thể sử dụng lớp lọc bao bọc một luồng nhập. Dùng lớp lọc cho phéo đọc số nguyên, số thực và chuỗi thay vì đọc byte và ký tự.

Một số luồng lọc thường dùng:

Tên lớp	Cách dùng lớp
DataInputStream	Xử lý dạng thức nhị phân cho mọi loại dữ liệu sơ cấp
BufferedInputStream	Lấy dữ liệu từ vùng đệm và đọc dữ liệu từ luồng, nếu cần.
DataOutputStream	Xuất dạng thức nhị phân của mọi loại dữ liệu sơ cấp, vốn rất hữu ích nếu chương trình khác sử dụng đầu xuất này
BufferedOutputStream	Xuất sang vùng đệm trước,sau đó sang luồng, nếu cần. Ta có thể gọi phương thức **flush()** ghi dữ liệu vùng đệm vào luồng

5) Luồng dữ liệu:

Luồng dữ liệu (DataInputStream và DataOutputStream) đọc và ghi dạng dữ liệu sơ cấp. Phương thức khởi tạo sau:

```
public DataInputStream(InputStream instream)
public DataOutputStream(OutputStream outstream)
```

Ví dụ:

```
DataInputStream infile = new DataInputStream(new
                            FileInputStream("in.dat"));
DataOutputStream outfile = new DataOutputStream(new
                            FileOutputStream("out.dat"));
```

Một số phương thức thường dùng:

```
pubic int readByte() throws IOException
pubic int readInt() throws IOException
pubic long readLong() throws IOException
pubic double readDouble() throws IOException
pubic String readLine() throws IOException

pubic void writeByte(byte b) throws IOException
pubic void writeInt(int i) throws IOException
pubic void writeLong(long l) throws IOException
pubic void writeDouble(byte d) throws IOException
```

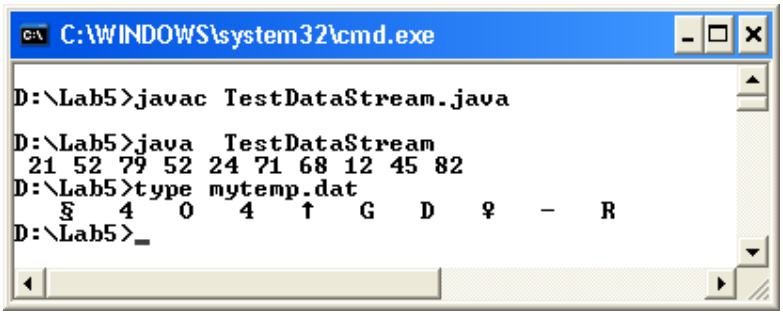
```
pubic void writeChars(String s) throws IOException
```

Ví dụ:

- Chương trình minh họa tạo 10 số nguyên ngẫu nhiên, lưu chúng vào tập tin dữ liệu, truy xuất dữ liệu từ tập tin và hiển thị số nguyên trên console.

```
import java.io.*;
public class TestDataStream{
  public static void main (String args[]){
    DataInputStream dis=null;
    DataOutputStream dos=null;
    // write data to file
    try{
      dos = new DataOutputStream(new FileOutputStream(
                                  "mytemp.dat"));
      for(int i=0;i<10;i++){
        dos.writeInt((int)(Math.random()*100));
      }
    }
    catch(IOException ex){
      System.out.println(ex.getMessage());
    }
    finally{
      try{
        if(dos!=null)
          dos.close();
      }
      catch(Exception ex){
      }
    }
    // read data from file
    try{
      dis = new DataInputStream(new FileInputStream(
                                 "mytemp.dat"));
      for(int i=0;i<10;i++){
        System.out.print(" " + dis.readInt());
      }
```

```
    }
    catch(FileNotFoundException ex){
      System.out.println("File not found");
    }
    catch(IOException ex){
      System.out.println(ex.getMessage());
    }
    finally{
      try{
        if(dis!=null)
          dis.close();
      }
      catch(Exception ex){
      }
    }
  }
}
```



Kết xuất của TestDataStream.java

6) *Luồng in ấn (PrintWriter):*

- Vì luồng xuất dữ liệu hiển thị dữ liệu ở dạng nhị phân, nên ta không thể xem nội dung của nó ở dạng văn bản (Xem kết quả của ví dụ ở mục trước).
- Trong Java, có thể dùng luồng in ấn (print stream) xuất dữ liệu vào tập tin được xem ở dạng văn bản.

Phương thức khởi tạo sau:

```
public PrintWriter(Writer out)
public PrintWriter(Writer out, boolean autoFlush)
```

Một số phương thức thường dùng:

```
pubic void print(Object o) throws IOException
pubic void print(String s) throws IOException
```

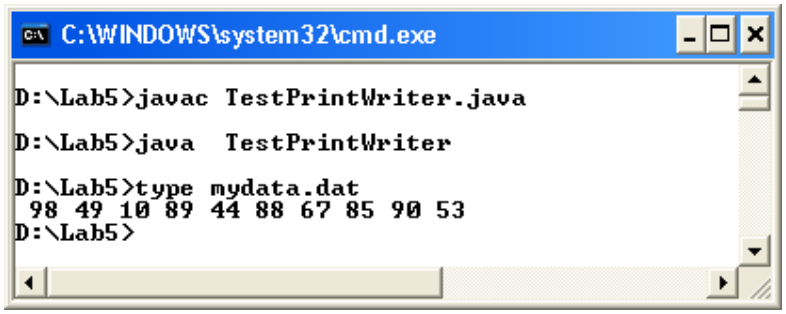
```
pubic void print(int i) throws IOException
pubic void print(long l) throws IOException
pubic void print(double d) throws IOException
pubic void print(boolean b) throws IOException
```

- Ta có thể thay thế print bằng println. Phương thức println() có chức năng in dữ liệu, theo sau là một dòng mới.

Ví dụ:

- Chương trình minh họa tạo 10 số ngẫu nhiên và lưu chúng vào tập tin dữ liệu văn bản "mydata.dat'. Có thể xem tập tin này trên console bằng lệnh type trên Windows hay cat trên Linux.

```
import java.io.*;
public class TestPrintWriter{
  public static void main(String args[]){
    PrintWriter pw=null;
    // write data to file
    try{
      pw = new PrintWriter(new FileOutputStream(
                          "mydata.dat"),true);
      for(int i=0;i<10;i++){
        pw.print( " " + (int)(Math.random ()*100));
      }
    }
    catch(IOException ex){
      System.out.println(ex.getMessage ());
    }
    finally{
      try{
        if(pw!=null)
        pw.close ();
      }
      catch(Exception ex){
      }
    }
  }
}
```



Kết xuất của TestPrintWriter.java

7) Luồng đệm:

- Java tăng tốc nhập / xuất dựa vào các luồng đệm (buffered stream), giảm bớt số lần đọc và ghi.
- Dùng luồng đệm giúp ta có thể đọc / ghi mỗi lần một khối byte hay ký tự thay vì đọc từng byte hoặc từng ký tự.

Các lớp luồng đệm trong java:

- BufferedInputStream
- BufferedOutputStream
- BufferedReader
- BufferedWriter

Các phương thức khởi tạo luồng đệm:

```
public BufferedInputStream(InputStream in)
public BufferedOutputStream(OutputSream out)
public BufferedReader(Reader out)
public BufferedWriter(Writer out)
```

- Lớp luồng đệm thừa kế phương thức các lớp cha. Ngoài việc sử dụng phương thức từ lớp cha, BufferedReader còn có phương thức readLine () chịu trách nhiệm đọc nội dung.

Ví dụ:

- Chương trình mẫu minh họa xem tập tin trong vùng văn bản. Khi người dùng gõ tên tập tin vào trường văn bản và nhấp nút View, tập tin sẽ hiển thị trong vùng văn bản.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
public class ViewFile extends JFrame implements
ActionListener{
  private JTextArea  contentView;
  private JTextField txtInput;
  private JButton  btnView;
```

```
public ViewFile(){
  setTitle ("View File");
  JPanel p = new JPanel();
  p.setLayout (new FlowLayout (FlowLayout.LEFT));
  p.add (new JLabel ("File name"));
  p.add (txtInput= new JTextField (20));
  p.add(btnView = new JButton("View"));
  //register listener
  btnView.addActionListener(this);
  contentView  = new JTextArea(20,30);
  JScrollPane pane = new JScrollPane(contentView);
  getContentPane().add(pane,BorderLayout.CENTER);
  getContentPane().add(p,BorderLayout.NORTH);
}
public void actionPerformed(ActionEvent e){
  if(e.getSource()== btnView)
    showFile();
}
public void showFile(){
  BufferedReader inFile=null;
  String filename = txtInput.getText().trim();
  String inLine;
  try{
    inFile = new BufferedReader(new FileReader(
                                filename));
    boolean firstLine=true;
    while((inLine=inFile.readLine())!=null){
      if(firstLine){
        firstLine=false;
        contentView.append(inLine);
      }
      else
        contentView.append("\n" + inLine);
    }
  }
```

```
    catch(Exception ex){
      System.out.println(ex.toString());
    }
    finally{
      try{
        if(inFile!=null)
          inFile.close();
       }
       catch(Exception ex){
       }
    }
  }
  public static void main(String args[]){
    ViewFile  frm = new ViewFile();
    frm.setDefaultCloseOperation(
                                    JFrame.EXIT_ON_CLOSE);
    frm.pack ();
    frm.setVisible (true);
  }
}
```

Kết xuất của ViewFile. java

8) Lớp StreamTonkenizer:

- Để phân tích cú pháp và từ dựng một cách mạnh mẽ và hiệu quả hơn ta cần sử dụng lớp StreamTokenizer.

- Lớp StreamTokenizer có khả năng nhận dạng ra các token trên luồng. Nếu hình dung tập hợp một chuỗi các ký tự làm nên một câu thì token chính là từng từ, từng dấu chấm câu hay con số.

Phương thức khởi tạo sau:

```
public StreamTokenizer(InputStream in)
```

Một số phương thức thường dùng:

```
pubic int nextToken () throws IOException
```

- Lấy ra từng token, phương thức trả về một trong những giá trị sau:
  - *StreamTokenizer.TT_WORD: token đọc được là một từ*
  - *StreamTokenizer.TT_NUMBER: token đọc được là một số*
  - *StreamTokenizer.TT_EOL: đọc đến cuối dòng*
  - *StreamTokenizer.TT_EOF: đọc đến cuối luồng.*

Lớp StreamTokenizer có 2 thuộc tính:

```
public String sval
```

- Lấy về giá trị của token nếu token đọc được là TT_WORD

```
public double nval
```

- Lấy về giá trị của token nếu token đọc được là TT_NUMBER

Chương 7 ĐA TUYẾN TRÌNH

Sau bài học này, học viên có thể:

- Hiểu được khái niệm về tuyến và cơ chế lập trình đa tuyến
- Tạo được tuyến trong chương trình.
- Xử lý cơ chế đồng bộ (synchronized) giữa các tuyến trình.

I. Khái niệm:

1) Tuyến (Thread):

Là đơn vị nhỏ nhất của đoạn mã có thể thi hành được để thực hiện một công việc riêng biệt nào đó. Những chương trình trước đây đều chạy ở một tuyến; nghĩa là vào thời điểm xác định chỉ có một câu lệnh được thực thi.

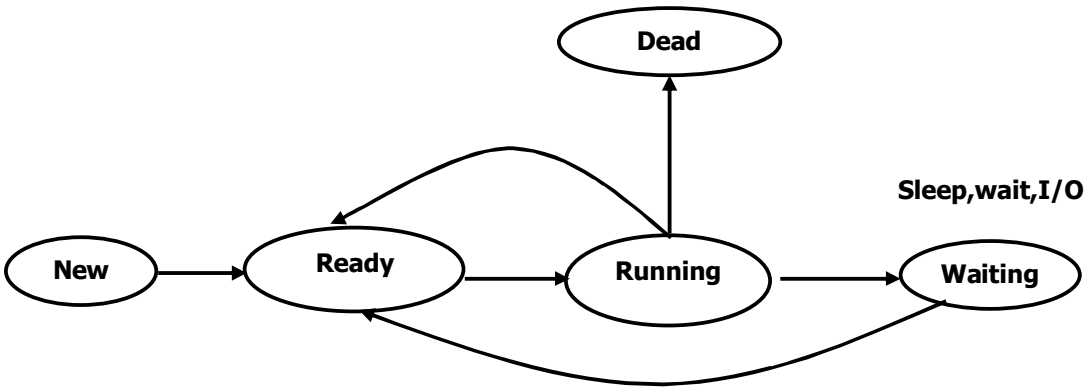
2) Đa tuyến (MultiThread):

- Là khả năng làm việc với nhiều tuyến
- Đa tuyến chuyên sử dụng cho việc thực thi nhiều công việc đồng thời
- Đa tuyến giảm thời gian rỗi của hệ thống đến mức thấp nhất và tăng hiệu suất thi hành của chương trình.

Tuyến có thể được tạo ra bằng 2 cách:

- Dẫn xuất từ lớp Thread
- Ứng dụng giao diện Runnable.

Chu kỳ sống của tuyến:



II. PHƯƠNG PHÁP XÂY DỰNG TUYẾN:

1) Tạo tuyến bằng cách dẫn xuất lớp Thread:

- Lớp Thread chứa phương thức tạo dựng Thread() cũng như nhiều phương thức hữu ích có chức năng chạy, khởi động, tạm ngừng, tiếp tục, gián đoạn và ngừng tuyến.
- Để tạo 1 tuyến ta có thể định nghĩa 1 lớp kế thừa từ lớp Thread và phải định nghĩa lại phương thức run() để thông báo cho hệ thống biết cách thực thi tuyến khi nó vận hành.

Cấu trúc tạo 1 tuyến:

```
class ThreadX extends Thread{
  public void run(){
    // cài đặt công việc của tuyến
  }
  ...
}
```

Thi hành tuyến:

```
ThreadX tx = new ThreadX();
tx.start();
```

Một số phương thức thường dùng trong lớp tuyến:

```
public void run()
```

- Phương thức được hệ thống gọi để thực thi công việc của tuyến, ta phải giành quyền phương thức này và cung cấp mã cho tuyến thi hành.
- Phương thức run() không được gọi trực tiếp từ đối tượng Thread.

```
public void start()
```

- Khởi động tuyến, qua đó làm cho phương thức run() cũng được kích hoạt.

```
public static void sleep(long millis) throws
                                    InterruptedException
```

- Đặt tuyến vào trạng thái "ngủ" trong khoảng thời gian xác định bằng mili giây.

```
public void stop()
```

- Có chức năng ngừng tuyến.

```
public void join() throws InterrupteException
```

- Chờ đợi tuyến này chết đi.

```
public void setPriority(int p)
```

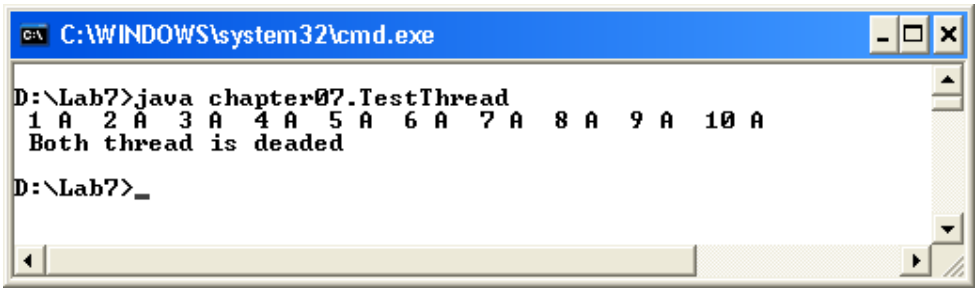
- Ấn định quyền ưu tiên p(xếp từ 1 đến 10) cho tuyến này.

Ví dụ:

- Chương trình này tạo và chạy 2 tuyến sau:
  - *Tuyến thứ nhất in các số nguyên từ 1 đến 10.*
  - *Tuyến thứ hai in chữ A 10 lần.*

```
package chapter07;
class PrintNum extends Thread{
  public void run(){
    for(int i=1;i<=10;i++){
      System.out.print(" " + i);
      try{
```

```
        Thread.sleep(300);
      }catch(Exception ex){}
    }
  }
}
class PrintChar extends Thread{
  public void run(){
    for(int i=1;i<=10;i++){
      System.out.print(" A ");
      try{
        Thread.sleep(300);
      }
      catch(Exception ex){
      }
    }
  }
}
public class TestThread{
  public static void main(String args[]){
    PrintNum t1= new PrintNum();
    PrintChar t2= new PrintChar();
    t1.start();
    t2.start();
    try{
      t1.join();
      t2.join();
    }
    catch(Exception ex){
    }
    System.out.println("\n Both thread is deaded");
  }
}
```



Kết quả của TestThread.java

2) Tạo tuyến bằng cách ứng dụng giao diện Runable:

- Ở phần trước, ta đã tạo và chạy tuyến thi hành bằng cách khai báo một lớp tuyến mở rộng lớp Thread. Phương thức này sẽ có hiệu lực nếu lớp tuyến chỉ kế thừa từ lớp Thread, nhưng sẽ vô dụng khi lớp tuyến thừa kế nhiều lớp. Để thừa kế nhiều lớp, ta phải ứng dụng giao diện.Java cung cấp giao diện Runnable để thay thế lớp Thread.

Cấu trúc định nghĩa 1 tuyến:

```
class RunnableY implements Runnable {
  public void run(){
  // process Threads
  }
...
}
```

Thi hành tuyến:

```
RunnableY ry = new RunnableY();
Thread ty = new Thread(ry);
tx.start();
```

Ví dụ:

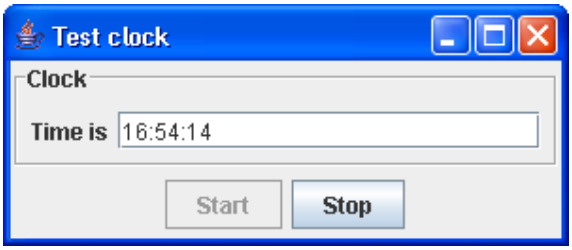
- Chương trình mẫu định nghĩa một tuyến hiển thị thời gian (hh:mm:ss) trong một panel.
- Tập tin ClockPanel.java

```
package chapter07;
import javax.swing.*;
import java.awt.*;
import java.util.*;
import javax.swing.border.*;
public class ClockPanel extends JPanel implements
Runnable{

    private JTextField txtDisplay;
    private Thread t;
```

```
    public ClockPanel(){
        add(new JLabel("Time is"));
        add(txtDisplay= new JTextField(20));
        setBorder(new TitledBorder(new
EtchedBorder(),"Clock"));
    }
    public void run(){
        String msg="";
        while(true){
           Date today = new Date();
           msg = today.getHours() + ":" +
today.getMinutes()+":" + today.getSeconds();
           txtDisplay.setText(msg);
           try{
             Thread.sleep(1000);
           }catch(Exception ex){
           }
        }
    }
}
// TestClock.java
package chapter07;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.border.*;
public class TestClock extends JFrame implements
                                        ActionListener{
  JButton btnStart,btnStop;
  ClockPanel p1 = new ClockPanel();
  Thread t;
  public TestClock(){
    setTitle("Test clock");
    getContentPane().add(p1,BorderLayout.NORTH);
    JPanel p2 = new JPanel();
```

```
    p2.add(btnStart = new JButton("Start"));
    p2.add(btnStop = new JButton("Stop"));

    getContentPane().add(p2,BorderLayout.CENTER);
    btnStart.addActionListener(this);
    btnStop.addActionListener(this);
    btnStop.setEnabled(false);
  }
  public void actionPerformed(ActionEvent e){
    if(e.getSource()==btnStart){
      start();
      btnStart.setEnabled(false);
      btnStop.setEnabled(true);
    }
    else if(e.getSource()==btnStop){
      stop();
      btnStart.setEnabled(true);
      btnStop.setEnabled(false);
    }
  }
  public void start(){
    t = new Thread(p1);
    t.start();
  }
  public void stop(){
    if(t!=null){
      t.stop();
    }
  }
  public static void main(String args[]){
    TestClock frm =new TestClock();
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.pack();
    frm.setVisible(true);
  }
```

```
}
```



Kết quả của TestClock.java