



Ngôn Ngữ Java

REVIEW



A decorative graphic consisting of a thick green line that curves from the bottom left towards the top left, then extends horizontally across the bottom. At the top left end of the curve is a cluster of overlapping circles in shades of green, yellow, orange, and pink. At the bottom left end of the horizontal line is a solid green circle. At the bottom right end of the horizontal line is another cluster of overlapping circles in shades of green, yellow, orange, and pink.

# Tổng quan về công nghệ Java

# Nội dung

- Lịch sử phát triển
- Công nghệ Java
- Các dạng chương trình Java
- Đặc điểm của Java
- Máy ảo Java (Java Virtual Machine)
- Viết, dịch, thực thi chương trình HelloWorld
- Môi trường, công cụ: giới thiệu một số IDE phổ biến

# Lịch sử phát triển

- 1991: Sun Microsystems phát triển OAK nhằm mục đích viết phần mềm điều khiển (phần mềm nhúng) cho các sản phẩm gia dụng.



- 1995: internet bùng nổ, phát triển mạnh. Sun phát triển OAK và giới thiệu ngôn ngữ lập trình mới tên Java



- Java là ngôn ngữ hướng đối tượng tựa C, C++


# Lịch sử phát triển Java Development Kit (JDK)

- Môi trường phát triển và thực thi do Sun Microsystems cung cấp (<http://java.sun.com>)
- Bao gồm phần mềm và công cụ giúp compile, debug and execute ứng dụng.
  - ✓ JDK 1.0 - 1996
  - ✓ JDK 1.1 - 1997
  - ✓ JDK 1.2 (Java 2) - 1998
  - ✓ JDK 1.3 - 2000
  - ✓ Java 1.4 - 2002
  - ✓ Java 5 (1.5) - 2004
  - ✓ Java 6 - 2006
  - ✓ Java SE 7 - 2011
  - ✓ Java SE 8 ~2013 ~2014



# Java Development Kit (JDK)

- Bao gồm

- ✓ javac                      Chương trình dịch chuyển mã nguồn sang bytecode
  - ✓ java                      Bộ thông dịch: Thực thi java application
  - ✓ appletviewer              Bộ thông dịch: Thực thi java applet mà không cần sử dụng trình duyệt như Netscape, hay IE, v.v.
  - ✓ javadoc                    Bộ tạo tài liệu dạng HTML từ mã nguồn và chú thích Lập trình trên thiế
- 



# Java Development Kit (JDK)

- Bao gồm

- ✓ jdb

Bộ gỡ lỗi (java debugger)

- ✓ javap

Trình dịch ngược bytecode



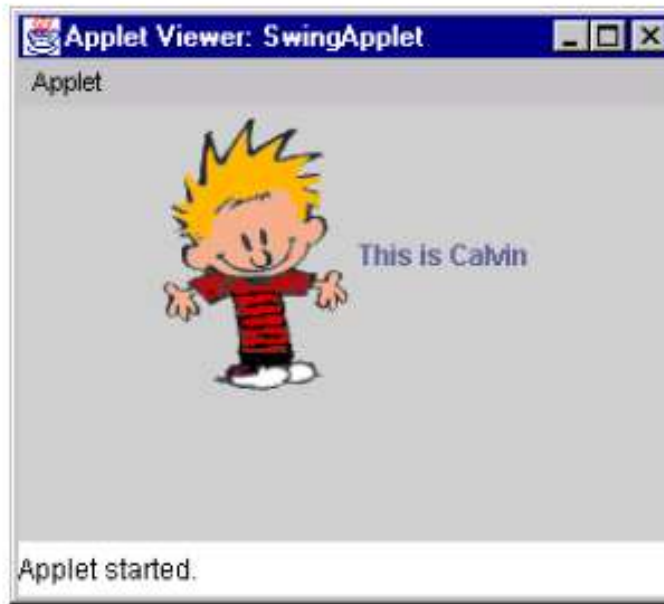
# Công nghệ Java

- Công nghệ:
  - ✓ Ngôn ngữ lập trình
  - ✓ Môi trường thực thi và triển khai
  - ✓ Môi trường phát triển
- Công nghệ J2SE (Java 2 Standard Edition)
- Công nghệ J2EE (Java 2 Enterprise Edition)
- Công nghệ J2ME (Java 2 Micro Edition)



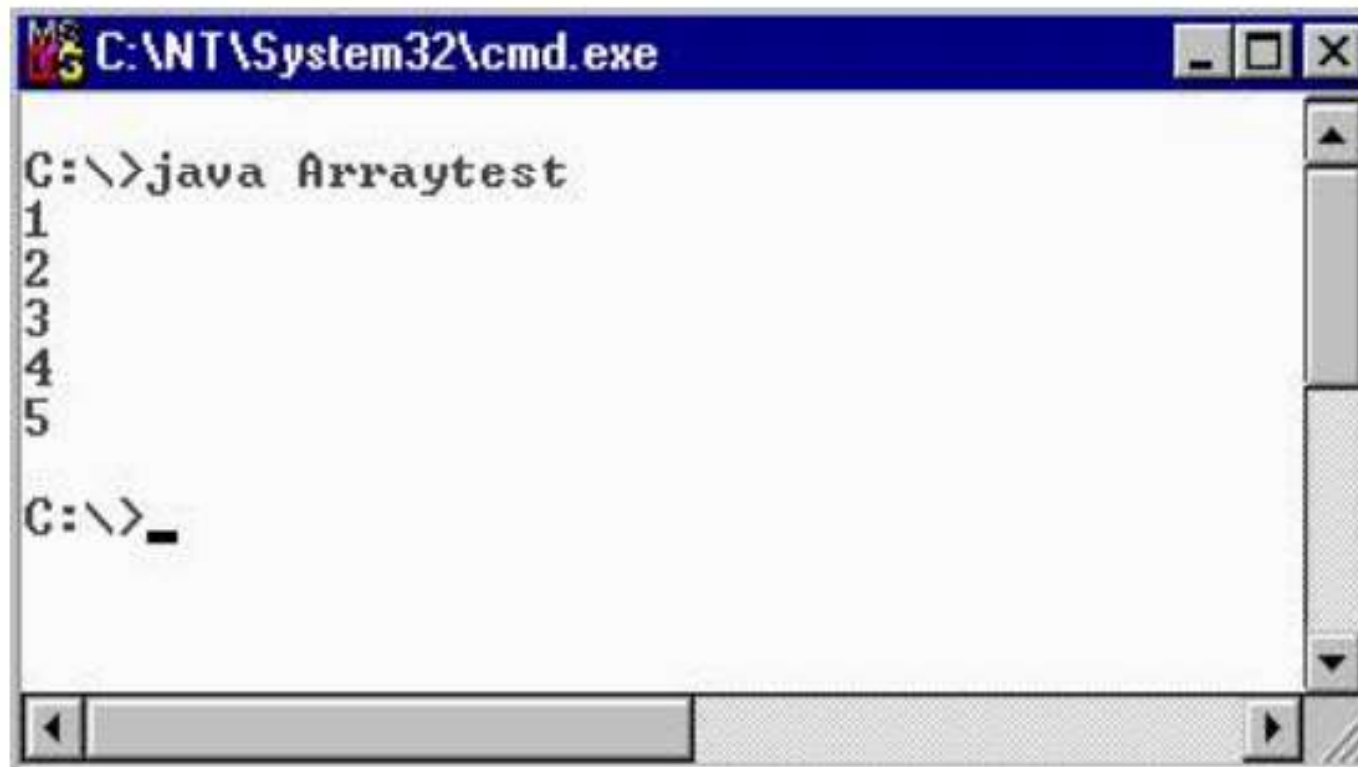
# Các dạng chương trình Java

- Applets



# Các dạng chương trình Java (tt)

- Console Applications

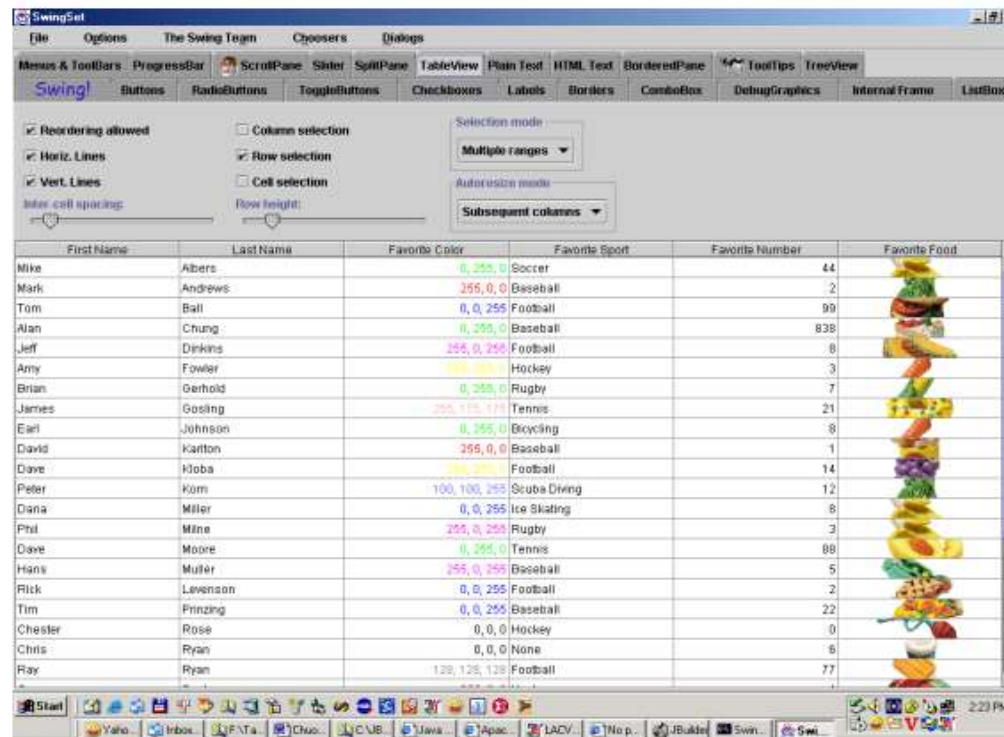


```
C:\NT\System32\cmd.exe

C:\>java Arraytest
1
2
3
4
5
C:\>_
```

# Các dạng chương trình Java (tt)

- Ứng dụng Desktop



# Các dạng chương trình Java (tt)

- Ứng dụng Web

The screenshot displays the homepage of the website **nhadat.com.vn**, which translates to "real estate for everyone". The interface is organized into several sections:

- Header:** Includes the site logo, contact information (phone 9302982, email info@nhadat.com), and links for "Đăng ký" (Register) and "Giúp đỡ" (Help).
- Navigation Bar:** Features links for "trang đầu" (Home), "tìm mua/thuê" (Find buy/rent), "bán/cho thuê" (Sell/for rent), and "săn lùng mặt tiền" (Hunt for prime locations).
- Left Sidebar:**
  - Location filter: "TP Hồ Chí Minh" and "tỉnh thành khác" (Other provinces/cities).
  - Service type: "THỰC ĐƠN NHANH" (Quick menu) with options like "Trang nhà đất chăm com của tôi" and "Trở thành đối tác của nhadat.com".
  - Registration form: A field for email address with a "Gửi" (Send) button.
  - Logos for "VINHTUONG" and "HOREA".
  - Text: "Khu phố yên vui Tịch nghi hiện đại" (Peaceful and modern residential area).
- Main Content Area:**
  - Featured Listing:** A large banner for "Đón lộc KIM CƯƠNG Thịnh Vượng KIMHONG" featuring an image of a modern apartment building.
  - Tin tức (News):** A list of three news items with brief descriptions and links to "chi tiết" (Details).
  - Thị trường (Market):** A section titled "Nhà đất Giá Tốt Nhất" (Best Price Real Estate) listing various properties and their locations.
  - Biệt Thự Cho Thuê Q.Thủ Đức:** A listing for a villa for rent in Thủ Đức district, highlighting its large area and modern design.
- Right Sidebar:**
  - chợ nhadatchiamcom:** A section encouraging users to choose from various conditions and prices.
  - Filters:** Buttons for "Đất" (Land), "Quận" (District), "Giá" (Price), and "20 mục mới nhất" (20 latest items).
  - nhà/đất cho thuê:** A section for rental properties.
  - Tìm nhà/đất theo mã số:** A search bar with a "Tìm" (Find) button.
  - Công ty Môi Giới:** A list of real estate agencies, including "Tân Kỳ Nguyễn", "Phúc Đức", "Hồng Nhật", "Du Lịch Địa Ốc Ánh Dương", "Mạnh Hùng", "Anh Trung", and "nhà đất ĐỒ THI MỚI".

# Các dạng chương trình Java (tt)


- Một dạng phần mềm nhúng



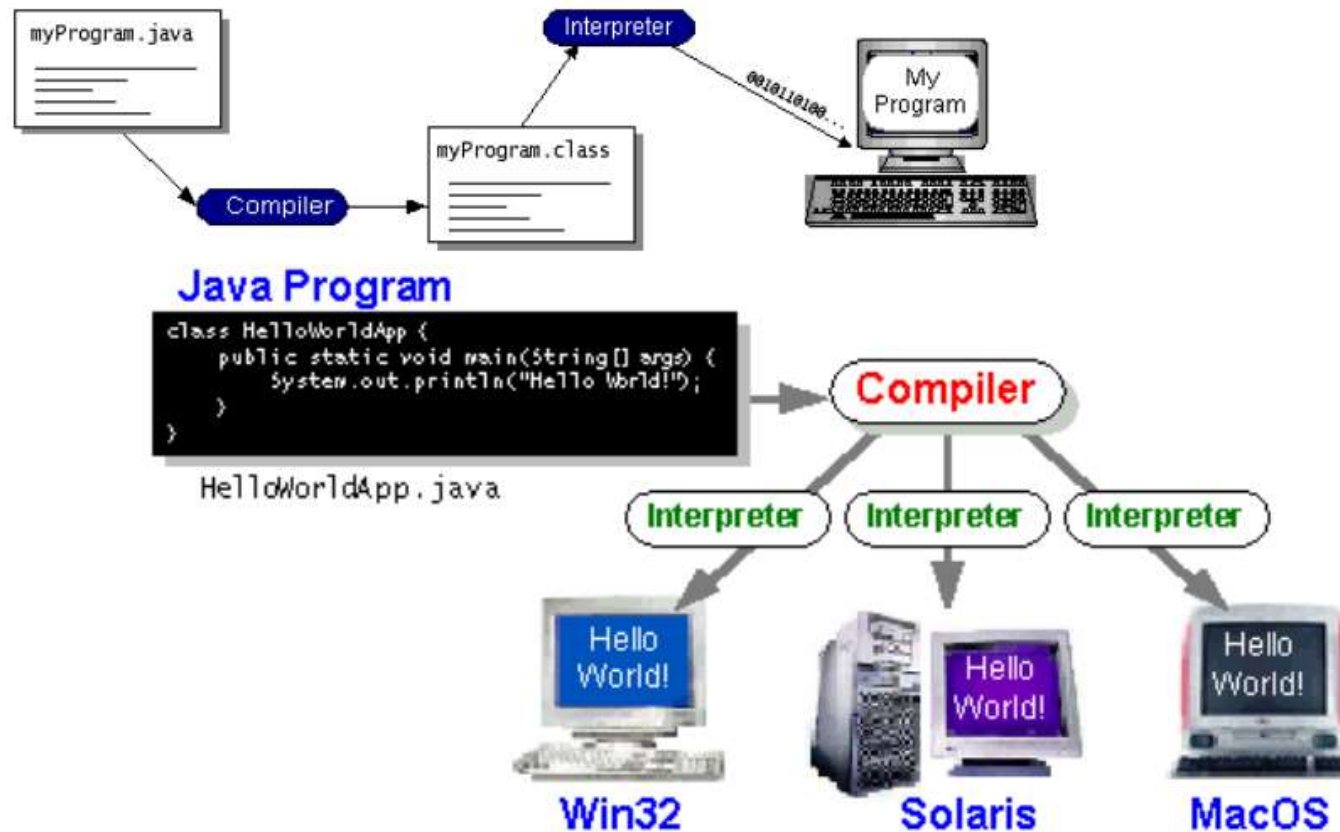




# Đặc điểm Java


- Tựa C++, hướng đối tượng hoàn toàn.
  - Khả chuyển, độc lập nền.
  - Thông dịch (vừa biên dịch vừa thông dịch).
  - Cơ chế giải phóng bộ nhớ tự động.
  - An toàn, bảo mật.
- 

# Dịch và thực thi chương trình Java






# Java Virtual Machine

- Là phần mềm dựa trên cơ sở máy tính ảo.
  - Có thể xem như 1 hệ điều hành thu nhỏ.
  - Cung cấp môi trường thực thi cho chương trình java (độc lập nền)
  - Hình thành 1 lớp trừu tượng:
    - Phần cứng máy tính bên dưới
    - Hệ điều hành
    - Mã đã biên dịch
  - Chương trình java chỉ chạy khi có JVM
  - JVM đọc và thực thi từng câu lệnh java
- 







## Giải phóng bộ nhớ (Garbage Collection)

- Java cung cấp một tiến trình mức hệ thống để theo dõi việc cấp phát bộ nhớ
  - Garbage Collection
    - Đánh dấu và giải phóng các vùng nhớ không còn được sử dụng
    - Được tiến hành tự động
    - Cơ chế hoạt động phụ thuộc vào các phiên bản máy ảo
- 

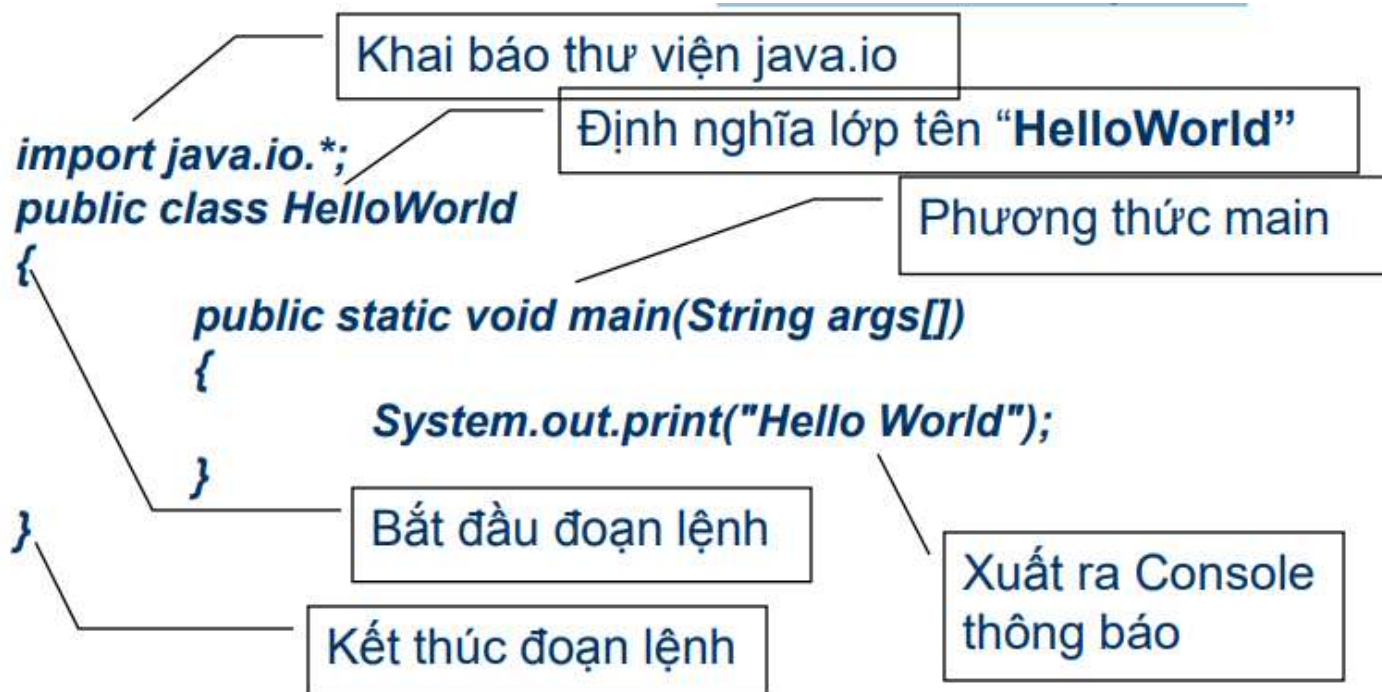


## Môi trường, công cụ

- Môi trường phát triển và thực thi của Sun – JDK 7
  - IDE (Integrated Development Enviroment)
    - Jcreator Pro 5.0
    - NetBeans 7.0
    - Eclipse 3.7
    - JBuilder 2008
- 
- 

# Chương trình Hello World

- Dùng Notepad soạn thảo đoạn lệnh bên dưới và lưu lại với tên HelloWorld.java



# Chương trình Hello World

- Biên dịch: dùng chương trình javac

*C:\> javac HelloWorld.java*

- Biên dịch thành công tạo ra tập tin có đuôi .class (HelloWorld.class)

- Thông dịch (thực thi): dùng chương trình java

*C:\> java HelloWorld*

# Chương trình Hello World

- Lưu ý: Phải khai báo đường dẫn chỉ đến thư mục cài đặt java, và thư mục chứa các class cần thực thi
- Ví dụ:

*C:\> set path=C:\jdk1.5\bin\*

*C:\> set classpath = D:\ThucHanhJava\BT1\*

# Chương trình Hello World

## TestGreeting.java:

```
import java.io.*;
public class TestGreeting
{
    public static void main(String[]
args)
    {
        Greeting gr = new Greeting();
        gr.greet();
    }
}
```

## Greeting.java:

```
public class Greeting
{
    public void greet()
    {
        System.out.print("Hello World");
    }
}
```



# Chương trình Hello World

- Biên dịch TestGreeting.java

*javac TestGreeting.java*

- Greeting.java được biên dịch tự động
- Thực hiện

*java TestGreeting*


- Kết quả

*Hello World*





# Java Applets

- Được nhúng trong một ứng dụng khác (web browser)
  - Có giao diện hạn chế (đồ họa)
  - Không truy cập được tài nguyên của client
- 



# Applet đơn giản

## Welcome.java:

```
// Java packages
import java.awt.Graphics;
import java.applet.Applet;
public class Welcome extends Applet
{
    public void paint(Graphics g)
    {
        // call superclass version of method paint
        super.paint(g);
        // draw a String
        g.drawString("Welcome to Java programming!", 25, 25);
    }
}
```

# Nhúng vào trang web

```
<html>
```

```
  <applet code = "Welcome.class"
```

```
    width = "300" height = "45">
```

```
  </applet>
```

```
</html>
```

# Thực hiện (trong web browser)



# Thực hiện

appletviewer Welcome.html






# Căn bản về ngôn ngữ Java





# Nội dung

- Biến & Hằng
  - Kiểu dữ liệu (kiểu cơ sở, kiểu tham chiếu)
  - Toán tử, biểu thức
  - Các cấu trúc điều khiển (chọn, rẽ nhánh, lặp)
  - Lớp bao kiểu cơ sở
  - Phương thức và cách sử dụng
  - Một số ví dụ minh họa
- 
-

# Biến

- Biến là một vùng nhớ lưu các giá trị của chương trình
- Mỗi biến gắn với 1 kiểu dữ liệu và 1 định danh duy nhất là tên biến
- Tên biến phân biệt chữ hoa và chữ thường. Tên biến bắt đầu bằng 1 dấu \_, \$, hay 1 ký tự, không được bắt đầu bằng 1 ký số.
- Khai báo ; = ;
- Gán giá trị = ;
- Lưu ý: trong java nếu lúc khai báo không khởi tạo giá trị cho biến thì nó sẽ nhận 1 giá trị mặc định. Mỗi kiểu dữ liệu có 1 kiểu dữ liệu mặc định khác nhau.

# Hằng

- Là một giá trị bất biến trong chương trình
- Tên đặt theo qui ước như tên biến
- Được khai báo dùng từ khóa final, và thường dùng tiếp vĩ ngữ đối với các hằng số (l, L, d, D, f, F)
- Ví dụ: `final int x = 10; // khai báo hằng số nguyên x = 10`  
`final long y = 20L; // khai báo hằng số long y = 20`
- Hằng ký tự: đặt giữa cặp nháy đơn „“
- Hằng chuỗi: là một dãy ký tự đặt giữa cặp nháy đôi “”


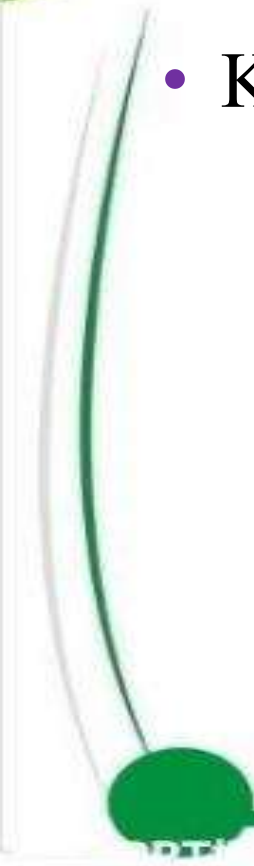



# Hàng ký tự đặc biệt

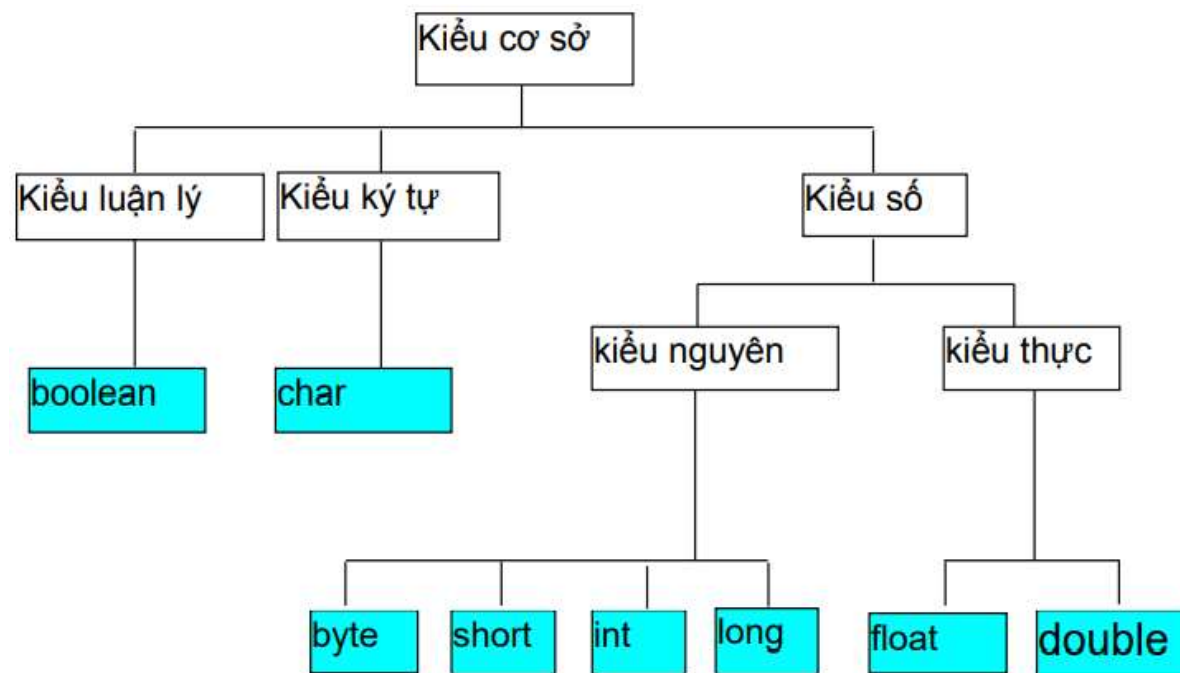
Ký tự	Ý nghĩa
\b	Xóa lùi (BackSpace)
\t	Tab
\n	Xuống hàng
\r	Dấu enter
\"	Nháy kép
\'	Nháy đơn
\\	\
\f	Đẩy trang
\uxxxx	Ký tự unicode



# Kiểu dữ liệu

- Kiểu dữ liệu cơ sở (primitive data type)
  - Kiểu dữ liệu tham chiếu (reference data type)
- 
- 
- 

# Kiểu dữ liệu cơ sở



# Kiểu dữ liệu cơ sở (tt)

Kiểu	Kích thước (bits)	Giá trị	Giá trị mặc định
<b>boolean</b>	[ <i>Note:</i> The representation of a boolean is specific to the Java Virtual Machine on each computer platform.]	true và false	<b>false</b>
<b>char</b>	16	"\u0000" to "\uFFFF" (0 to 65535)	<b>null</b>
<b>byte</b>	8	-128 to +127 ( $-2^7$ to $2^7 - 1$ )	<b>0</b>
<b>short</b>	16	-32,768 to +32,767 ( $-2^{15}$ to $2^{15} - 1$ )	<b>0</b>
<b>int</b>	32	-2,147,483,648 to +2,147,483,647 ( $-2^{31}$ to $2^{31} - 1$ )	<b>0</b>
<b>long</b>	64	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 ( $-2^{63}$ to $2^{63} - 1$ )	<b>0l</b>
<b>float</b>	32	1.40129846432481707e-45 to 3.4028234663852886E+38	<b>0.0f</b>
<b>double</b>	64	4.94065645841246544e-324 to 1.7976931348623157E+308	<b>0.0d</b>

## Kiểu dữ liệu cơ sở (tt)

- Chuyển đổi kiểu dữ liệu: khi có sự không tương thích về kiểu dữ liệu (gán, tính toán biểu thức, truyền đối số gọi phương thức)
  - ✓ Chuyển kiểu hẹp (lớn  $\rightarrow$  nhỏ): cần ép kiểu = (kiểu dữ liệu) ;
  - ✓ Chuyển kiểu rộng (nhỏ  $\rightarrow$  lớn): tự động chuyển



# Kiểu dữ liệu cơ sở (tt)

- Lưu ý

1. Không thể chuyển đổi giữa kiểu **boolean** với **int** và ngược lại.

2. Nếu 1 toán hạng kiểu **double** thì

“Toán hạng kia chuyển thành **double**”

Nếu 1 toán hạng kiểu **float** thì

“Toán hạng kia chuyển thành **float**”

Nếu 1 toán hạng kiểu **long** thì

“Toán hạng kia chuyển thành **long**”

Ngược lại “Tất cả chuyển thành **int** để tính toán”

# Kiểu dữ liệu cơ sở (tt)

- Ví dụ minh họa

1. `byte x = 5;`

2. `byte y = 10;`

3. `byte z = x + y;`

`// Dòng lệnh thứ 3 báo lỗi chuyển kiểu cần sửa lại`

`// byte z = (byte) (x + y);`

# Kiểu dữ liệu tham chiếu

- Kiểu mảng

- ✓ Mảng là tập hợp các phần tử có cùng tên và cùng kiểu dữ liệu.

- ✓ Mỗi phần tử được truy xuất thông qua chỉ số

- ❖ Khai báo mảng

*<kiểu dữ liệu>[] <tên mảng> ; // mảng 1 chiều*

*<kiểu dữ liệu> <tên mảng>[]; // mảng 1 chiều*

*<kiểu dữ liệu>[][] <tên mảng> ; // mảng 2 chiều*

*<kiểu dữ liệu> <tên mảng>[][]; // mảng 2 chiều*



## Kiểu dữ liệu tham chiếu (tt)

- **Khởi tạo**

*int arrInt[] = {1, 2, 3};*

*char arr Char[] = {„a“, „b“, „c“};*

*String arrString[] = {“ABC”, “EFG”, “GHI”};*

- **Cấp phát & truy cập mảng**

*int arrInt = **new** int[100];*

*int arrInt[100]; // Khai báo này trong Java sẽ bị báo lỗi.*

*Chỉ số mảng **n** phần tử: từ **0** đến **n-1** Lập trình trên thiết bị*

## Kiểu dữ liệu tham chiếu (tt)

- Kiểu đối tượng
- Khai báo đối tượng ;  
*<kiểu đối tượng> <biến ĐT>;*
- Khởi tạo đối tượng = new ;  
*<kiểu đối tượng> <biến ĐT> = new <kiểu đối tượng>;*
- Truy xuất thành phần đối tượng ..  
*<biến đối tượng>.<thuộc tính>*  
*<biến đối tượng>.<phương thức>*

# Toán tử, biểu thức

- Toán tử số học

Toán tử	Ý nghĩa
+	Cộng
-	Trừ
*	Nhân
/	Chia nguyên
%	Chia dư
++	Tăng 1
--	Giảm 1

# Toán tử, biểu thức (tt)

- Phép toán trên bit

Toán tử	Ý nghĩa
&	AND
	OR
^	XOR
<<	Dịch trái
>>	Dịch phải
~	Bù bit

# Toán tử, biểu thức (tt)

- Toán tử quan hệ và logic

Toán tử	Ý nghĩa
=	So sánh bằng
!=	So sánh khác
>	So sánh lớn hơn
<	So sánh nhỏ hơn
>=	So sánh lớn hơn hay bằng
<=	So sánh nhỏ hơn hay bằng
	OR (biểu thức logic)
&&	AND (biểu thức logic)
!	NOT (biểu thức logic)

# Toán tử, biểu thức (tt)

- Toán tử gán

Toán tử	Ví dụ	Ý nghĩa
=	$a = b$	Gán $a = b$
+=	$a += 5$	$a = a + 5$
-=	$a -= 10$	$b = b - 10$
*=	$c *= 3$	$c = c * 3$
/=	$d /= 2$	$d = d / 3$
%=	$e \% = 4$	$e = e \% 4$

# Toán tử, biểu thức (tt)

- **Toán tử điều kiện**

**Cú pháp:**  $\langle \text{điều kiện} \rangle ? \langle \text{biểu thức 1} \rangle : \langle \text{biểu thức 2} \rangle$

- **Ví dụ:**

- $\text{int } x = 10;$
- $\text{int } y = 20;$
- $\text{int } Z = (x < y) ? 30 : 40;$
- *//kết quả  $z = 30$  do biểu thức  $(x < y)$  là đúng*

# Cấu trúc điều khiển

- **Cấu trúc if ... else**

- **Dạng 1:** *if* (<điều\_kiện>) {  
    <khởi\_lệnh>;  
}

- **Dạng 2:** *if* (<điều\_kiện>) {  
    <khởi\_lệnh1>;  
}  
    *else* {  
        <khởi\_lệnh2>;  
    }



# Cấu trúc điều khiển

- Cấu trúc switch ... case

```
✓  switch (<biến>
    {
        case <giá trị_1> :
            <khởi_lệnh_1>;
            break;

        ....
        case <giá trị_n> :
            <khởi_lệnh_n>;
            break;

        default:
            <khởi_lệnh_default>;
    }
```

# Cấu trúc điều khiển

- **Cấu trúc lặp**

➤ **Dạng 1:**      *while* (<điều\_kiện\_lặp>) {  
                    <khởi\_lệnh>;  
                    }

➤ **Dạng 2:**      *do* {  
                    <khởi\_lệnh>;  
                    } *while* (<điều\_kiện>);

➤ **Dạng 3:** *for* (*khởi\_tạo\_biến\_đếm*; *đk\_lặp*; *tăng\_biến*) {  
                    <khởi\_lệnh>;  
                    }

# Cấu trúc điều khiển

- **Cấu trúc lệnh nhảy jump:** dùng kết hợp nhãn (label) với từ khóa **break** và **continue** để thay thế cho lệnh **goto** (trong C).

- Ví dụ:

label:

```
for (...) {  
    for (...) {  
        if (<biểu thức điều kiện>)  
            break label;  
        else  
            continue label;  
    }  
}
```

# Lớp bao kiểu dữ liệu cơ sở



Data type	Wrapper Class (java.lang.*)	Ghi chú
boolean	Boolean	<p>1.Gói (package): chứa nhóm nhiều class.</p> <p>2.Ngoài các Wrapper Class, gói java.lang còn cung cấp các lớp nền tảng cho việc thiết kế ngôn ngữ java như: String, Math, ...</p>
byte	Byte	
short	Short	
char	Character	
int	Integer	
long	Long	
float	Float	
double	Double	

A decorative graphic consisting of a thick green line that curves from the bottom left towards the center, then extends horizontally to the right. At the top left end of this curve is a cluster of overlapping circles in shades of green, yellow, orange, and pink. At the bottom left end of the horizontal line is a solid green circle. At the bottom right end of the horizontal line is another cluster of overlapping circles in shades of green, yellow, orange, and pink.

# Hướng đối tượng trong Java



## Nội dung

- Các khái niệm cơ bản về lớp, đối tượng.
  - Lớp và đối tượng trong java.
  - Tính đóng gói.
  - Tính kế thừa.
  - Tính đa hình.
  - Interface.
- 
- 

## Các khái niệm cơ bản

- Đối tượng (object): trong thế giới thực khái niệm đối tượng có thể xem như một thực thể: người, vật, bảng dữ liệu,...
  - ✓ Đối tượng giúp hiểu rõ thế giới thực
  - ✓ Cơ sở cho việc cài đặt trên máy tính
  - ✓ Mỗi đối tượng có định danh, thuộc tính, hành vi Ví dụ: đối tượng sinh viên MSSV: “TH0701001”; Tên sinh viên: “Nguyễn Văn A”
- Hệ thống các đối tượng: là 1 tập hợp các đối tượng
  - ✓ Mỗi đối tượng đảm trách 1 công việc
  - ✓ Các đối tượng có thể trao đổi thông tin với nhau
  - ✓ Các đối tượng có thể xử lý song song, hay phân tán

## Các khái niệm cơ bản

- Lớp (class): là khuôn mẫu (template) để sinh ra đối tượng. Lớp là sự trừu tượng hóa của tập các đối tượng có các thuộc tính, hành vi tương tự nhau, và được gom chung lại thành 1 lớp.
- Ví dụ: lớp các đối tượng Sinh viên
  - Sinh viên “Nguyễn Văn A”, mã số TH0701001
  - 1 đối tượng thuộc lớp Sinh viên
  - Sinh viên “Nguyễn Văn B”, mã số TH0701002
  - là 1 đối tượng thuộc lớp Sinh viên
- Đối tượng (object) của lớp: một đối tượng cụ thể thuộc 1 lớp là 1 thể hiện cụ thể của 1 lớp đó.



# Lớp và đối tượng trong Java

- **Khai báo lớp**

```
class < Class Name >
```

```
{
```

```
    < danh sách thuộc tính >
```

```
    < các khởi tạo >
```

```
    < danh sách các phương thức >
```

```
}
```

# Lớp và đối tượng trong Java (tt)

- Thuộc tính: các đặc điểm mang giá trị của đối tượng, là vùng dữ liệu được khai báo bên trong lớp

*class < Class Name >*

*{*

*< tiền tố > < kiểu dữ liệu > < tên thuộc tính >;*

*}*

Kiểm soát truy cập đối với thuộc tính

- \* **public**: có thể truy xuất từ bất kỳ 1 lớp khác.
- \* **protected**: có thể truy xuất được từ những lớp con.
- \* **private**: không thể truy xuất từ 1 lớp khác.

# Lớp và đối tượng trong Java (tt)

Phương thức: chức năng xử lý, hành vi của các đối tượng.

```
class < Class Name > {
```

```
...
```

```
< tiền tố > < kiểu trả về > < tên phương thức > (<  
các đối số >){
```

```
...
```

```
}
```

```
}
```

# Lớp và đối tượng trong Java (tt)

- **public**: có thể truy cập được từ bên ngoài lớp khai báo.
- **protected**: có thể truy cập được từ lớp khai báo và các lớp dẫn xuất (lớp con).
- **private**: chỉ được truy cập bên trong lớp khai báo.
- **static**: phương thức lớp dùng chung cho tất cả các thể hiện của lớp, có thể được thực hiện kể cả khi không có đối tượng của lớp.
- **final**: không được khai báo chồng ở các lớp dẫn xuất.
- **abstract**: không có phần source code, sẽ được cài đặt trong các lớp dẫn xuất.
- **synchronized**: dùng để ngăn những tác động của các đối tượng khác lên đối tượng đang xét trong khi đang đồng bộ hóa. Dùng trong lập trình multithreads.

# Lớp và đối tượng trong Java (tt)

Ví dụ 1:

```
class Sinhvien {
```

```
// Danh sách thuộc tính
```

```
String      maSv, tenSv, dcLienlac;
```

```
int          tuoi;
```

```
...
```

```
// Danh sách các khởi tạo
```

```
Sinhvien(){} 
```

```
Sinhvien (...) { ...} 
```

```
...
```

```
// Danh sách các phương thức
```

```
public void capnhatSV (...) { ...} 
```

```
public void xemThongTinSV() { ...} 
```

```
...
```

```
}
```

# Lớp và đối tượng trong Java (tt)

...

*// Tạo đối tượng mới thuộc lớp Sinhvien*

*Sinhvien sv = new Sinhvien();*

...

*// Gán giá trị cho thuộc tính của đối tượng*

*sv.maSv = "TH0601001";*

*sv.tenSv = "Nguyen Van A";*

*sv.tuoi = "20";*

*sv.dcLienlac = "KP6, Linh Trung, Thu Duc";*

...

*// Gọi thực hiện phương thức*

*sv.xemThongTinSV();*

# Lớp và đối tượng trong Java (tt)

**Ví dụ 2:**

```
class Sinhvien {  
// Danh sách thuộc tính  
    private String          maSv;  
    String          tenSv, dcLienlac;  
    int tuoi;  
    ...  
}  
...  
Sinhvien sv = new Sinhvien();  
sv.maSv = "TH0601001"; /* Lỗi truy cập thuộc tính private  
                           từ bên ngoài lớp khai báo */  
Sv.tenSv = "Nguyen Van A";
```

...

## Lớp và đối tượng trong Java (tt)

- **Khởi tạo (constructor):** là một loại phương thức đặc biệt của lớp, dùng để khởi tạo một đối tượng.  
Dùng để khởi tạo giá trị cho các thuộc tính của đối tượng.  
Cùng tên với lớp.  
Không có giá trị trả về.  
Có thể có tham số hoặc không.
- **Lưu ý:** Mỗi lớp sẽ có 1 constructor mặc định (nếu ta không khai báo constructor nào). Ngược lại nếu ta có khai báo 1 constructor khác thì constructor mặc định chỉ dùng được khi khai báo tường minh.



# Lớp và đối tượng trong Java (tt)

## Ví dụ 1

```
class Sinhvien {
```

```
...
```

```
// Không có định nghĩa constructor nào
```

```
}
```

```
...
```

```
// Dùng constructor mặc định Sinhvien
```

```
sv = new Sinhvien();
```

# Lớp và đối tượng trong Java (tt)

**Ví dụ 2:**

```
class Sinhvien
```

```
{
```

```
    // không có constructor mặc định  
    Sinhvien() {...}
```

```
}
```

```
Sinhvien sv = new Sinhvien();
```

```
// lỗi biên dịch
```

```
class Sinhvien
```

```
{
```

```
    // khai báo constructor mặc định  
    Sinhvien () {}  
    Sinhvien (< các đối số >) {...}
```

```
}
```

```
Sinhvien sv = new Sinhvien();
```

# Lớp và đối tượng trong Java (tt)

## **Phương thức khai báo chồng (overloading method):**

Việc khai báo trong một lớp nhiều phương thức có cùng tên nhưng khác tham số (khác kiểu dữ liệu, khác số lượng tham số) gọi là khai báo chồng phương thức.

**Ví dụ:**     *class Sinhvien*

{     ...

*public void xemThongTinSV() {...}*

*public void xemThongTinSV(String psMaSv)*

*{...}*

}

# Lớp và đối tượng trong Java (tt)

**Tham chiếu this:** là một biến ẩn tồn tại trong tất cả các lớp, **this** được sử dụng trong khi chạy và tham khảo đến bản thân lớp chứa nó.

**Ví dụ:** *class Sinhvien{*

*String maSv, tenSv, dcLienlac;*

*int tuoi;*

*...*

*public void xemThongTinSV() {*

*System.out.println (**this**.maSv);*

*System.out.println (**this**.tenSv);*

*...*

*}*

*}*

## Tính đóng gói

- **Đóng gói:** nhóm những gì có liên quan với nhau vào thành một và có thể sử dụng một cái tên để gọi.
- **Ví dụ:**
  - ✓ Các phương thức đóng gói các câu lệnh.
  - ✓ Đối tượng đóng gói dữ liệu và các hành vi/phương thức liên quan.(Đối tượng = Dữ liệu + Hành vi/Phương thức)

## Tính đóng gói

Đóng gói dùng để che dấu một phần hoặc tất cả thông tin, chi tiết cài đặt bên trong với bên ngoài.

**Ví dụ:** khai báo các lớp thuộc cùng gói trong java  
***package** ; // khai báo trước khi khai báo lớp*

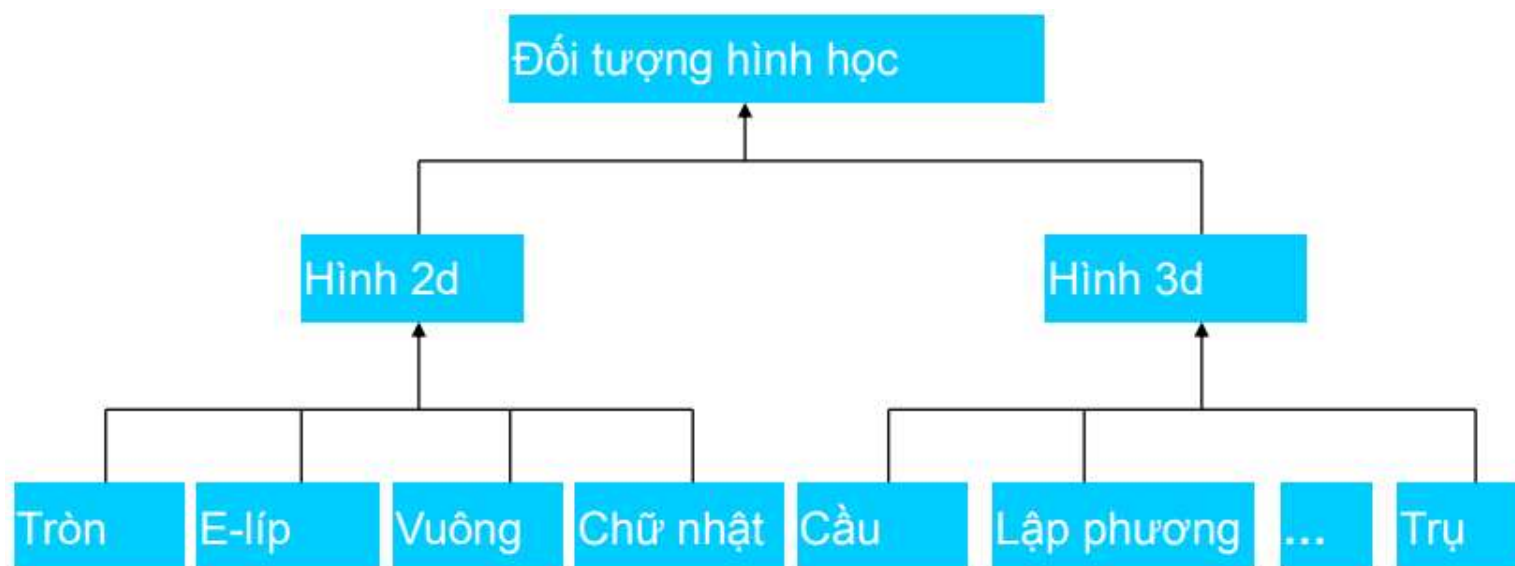
***class** < tên lớp >*

*{*

*...*

*}*

# Tính kế thừa



- Thừa hưởng các thuộc tính và phương thức đã có
- Bổ sung, chi tiết hóa cho phù hợp với mục đích sử dụng mới
  - ✓ Thuộc tính: thêm mới
  - ✓ Phương thức: thêm mới hay hiệu chỉnh

# Tính kế thừa (tt)

**Lớp dẫn xuất hay lớp con (SubClass)**

**Lớp cơ sở hay lớp cha (SuperClass)**

Lớp con có thể kế thừa tất cả hay một phần các thành phần dữ liệu (thuộc tính), phương thức của lớp cha (public, protected, default)

Dùng từ khóa **extends**.

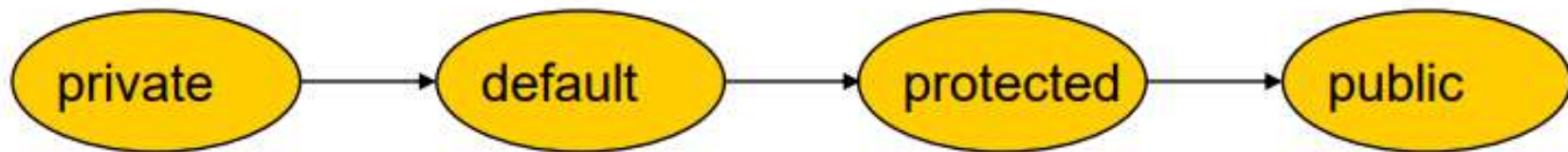
**Ví dụ:** *class nguoi { ...*  
    }  
    *class sinhvien extends nguoi { ...*  
    }

**Lưu ý:** default không phải là 1 từ khóa



## Tính kế thừa (tt)

- Phương thức định nghĩa lại (Overriding Method)
  - ✓ Được định nghĩa trong lớp con
  - ✓ Có tên, kiểu trả về & các đối số giống với phương thức của lớp cha
  - ✓ Có kiểu, phạm vi truy cập “lớn hơn” phương thức trong lớp cha



# Tính kế thừa (tt)

• Ví dụ:

```
abstract class Hinhhoc { ...  
    public float tinhdientich() {  
        return 0;  
    }  
    ...  
}  
  
class HinhVuong extends Hinhhoc {  
    private int canh;  
    public float tinhdientich() {  
        return canh*canh;  
    }  
    ...  
}
```

Chỉ có thể **public** do phương thức `tinhdientich()` của lớp cha là **public**

## Tính kế thừa (tt)

```
class HinhChuNhat extends HinhVuong
{
    private int cd;
    private int cr;
    public float tinhdientich() {
        return cd*cr;
    }
    ...
}
```

Chỉ có thể **public** do phương thức *tinhdientich()* của lớp cha là **public**

# Tính kế thừa (tt)

- **Lớp nội:** là lớp được khai báo bên trong 1 lớp khác. Lớp nội thể hiện tính đóng gói cao và có thể truy xuất trực tiếp biến của lớp cha.

**Ví dụ:**

```
public class A {  
    // ...  
    int static class B {  
        // ...  
        int public B(int par_1)  
        {  
            field_2 = par_1 + field_1;  
        }  
    }  
}
```

## Tính kế thừa (tt)

- **Lớp final**: là lớp không cho phép các lớp khác dẫn xuất từ nó hay lớp final không thể có lớp con.

Định nghĩa dùng từ khóa **final**

```
public final class A
```

```
{
```

```
...
```

```
}
```

## Tính kế thừa (tt)

- **Lớp trừu tượng:** là lớp dùng để thể hiện sự trừu tượng hóa ở mức cao.
- **Ví dụ:** lớp “Đối tượng hình học”, “Hình 2D”, “Hình 3D” (Ví dụ định nghĩa lớp các đối tượng hình học cơ bản)
- **Phương thức finalize() của lớp Object - protected void finalize():** được “Bộ thu gom rác” gọi tự động khi nhận ra không còn tham chiếu nào đến đối tượng đang xét.

# Tính đa hình

Ví dụ:

```
class A_Object {  
    // ...  
    void method_1() {  
        // ...  
    }  
}  
  
class B_Object extends A_Object {  
    // ...  
    void method_1() {  
        // ...  
    }  
}
```



# Tính đa hình

```
class C {  
    public static void main(String[] args) {  
        A_Object arr_Object = new A_Object[2];  
        B_Object var_1 = new B_Object();  
        arr_Object[0] = var_1;  
        A_Object var_2;  
        for (int i=0; i<2; i++) {  
            var_2 = arr_Object[i];  
            var_2.method_1();  
        }  
    }  
}
```

Phần tử đầu tiên của mảng `arr_Object[0]` tham chiếu đến 1 đối tượng kiểu `Object` dẫn xuất từ `A_Object`

- Với `i = 0` thì biến `var_2` có kiểu là `B_Object`, và lệnh `var_2.method_1()` sẽ gọi thực hiện phương thức `method_1` của lớp `B_Object`.
- Với `i = 1` thì biến `var_2` có kiểu là `A_Object`, và lệnh `var_2.method_1()` sẽ gọi thực hiện phương thức `method_1` của lớp `A_Object`.



# Giao tiếp - Interface

- Interface: giao tiếp của một lớp, là phần đặc tả (không có phần cài đặt cụ thể) của lớp, nó chứa các khai báo phương thức và thuộc tính để bên ngoài có thể truy xuất được. (java, C#, ...)
  - ✓ Lớp sẽ cài đặt các phương thức trong interface.
  - ✓ Trong lập trình hiện đại các đối tượng không đưa ra cách truy cập cho một lớp, thay vào đó cung cấp các interface. Người lập trình dựa vào interface để gọi các dịch vụ mà lớp cung cấp.
  - ✓ Thuộc tính của interface là các hằng và các phương thức của giao tiếp là trừu tượng (mặc dù không có từ khóa abstract).

# Giao tiếp – Interface (tt)

## Ví dụ:

```
// Định nghĩa một interface Shape trong tập tin shape.java public
interface Shape
{
    // Tính diện tích
    public abstract double area();
    // Tính thể tích
    public abstract double volume();
    // trả về tên của shape
    public abstract String getName();
}
```

# Giao tiếp – Interface (tt)

**// Lớp Point cài đặt/hiện thực interface tên shape.**

*// Định nghĩa lớp Point trong tập tin Point.java*

```
public class Point extends Object implements Shape {
```

```
    protected int x, y;
```

```
    // Tọa độ x, y của 1 điểm
```

```
    // constructor không tham số.
```

```
    public Point() {
```

```
        setPoint( 0, 0 );
```

```
    }
```

```
    // constructor có tham số.
```

```
    public Point(int xCoordinate, int yCoordinate) {
```

```
        setPoint( xCoordinate, yCoordinate );
```

```
    }
```

# Giao tiếp – Interface (tt)

*// gán tọa độ x, y cho 1 điểm*

```
public void setPoint( int xCoordinate, int yCoordinate ) {  
    x = xCoordinate;  
    y = yCoordinate;  
}
```

*// lấy tọa độ x của 1 điểm*

```
public int getX() {  
    return x;  
}
```

*// lấy tọa độ y của 1 điểm*

```
public int getY() {  
    return y;  
}
```

## Giao tiếp – Interface (tt)

*// Thể hiện tọa độ của 1 điểm dưới dạng chuỗi*

```
public String toString() {  
    return "[" + x + ", " + y + "];  
}
```

*// Tính diện tích public double area() {  
 return 0.0;*

```
}
```

*// Tính thể tích public double volume() {  
 return 0.0;*

```
}
```

## Giao tiếp – Interface (tt)

```
// trả về tên của đối tượng shape  
public String getName() {  
    return "Point";  
}  
} // end class Point
```

# Giao tiếp – Interface (tt)

- **Kế thừa giao diện**

```
public interface InterfaceName extends interface1,  
    interface2, interface3  
{  
    // ...  
}
```





# Quản lý Exceptions







# NỘI DUNG

- Giới thiệu về Exception
  - Kiểm soát Exception
  - Ví dụ minh họa
  - Thư viện phân cấp các lớp Exception
- 
- 

# Giới thiệu về Exception

## ***Ví dụ 1:***

```
...  
int x = 10;  
int y = 0;  
float z = x/y;  
System.out.print("Ket qua la:" + z);  
...
```

*Dòng lệnh thứ 3 có lỗi chia cho 0, vì vậy đoạn chương trình kết thúc và dòng lệnh thứ 4 xuất kết quả ra màn hình không thực hiện được.*

# Giới thiệu về Exception

**Ví dụ 2:**

```
...  
void docfile(String filename)  
{  
    ...  
    FileInputStream fin = new FileInputStream(filename);  
    ...  
}
```

Dòng lệnh trên có khả năng xảy ra lỗi đọc file (chẳng hạn khi file không có trên đĩa)

# Giới thiệu về Exception

- Exception
  - ✓ Dấu hiệu của lỗi trong khi thực hiện chương trình
  - ✓ ví dụ: lỗi chia cho 0, đọc file không có trên đĩa, ...
- Quản lý Exception (Exception handling)
  - ✓ Kiểm soát được lỗi từ những thành phần chương trình
  - ✓ Quản lý Exception theo 1 cách thống nhất trong những project lớn
  - ✓ Hạn chế, bỏ bớt những đoạn source code kiểm tra lỗi trong chương trình.

# Kiểm soát Exception

## Ví dụ 1:

```
...  
    try {  
        int x = 10;  
        int y = 0;  
        float z = x/y;  
        System.out.print("Ket qua la:" + z);  
    }  
    catch(ArithmeticException e) {  
        System.out.println("Loi tinh toan so hoc")  
    }  
...
```

# Kiểm soát Exception (tt)

## Ví dụ 2:

...

*void docfile(String filename) throws  
IOException {*

...

*FileInputStream fin = new  
FileInputStream(filename);*

...

*}*

# Kiểm soát Exception (tt)

**Hoặc**

```
...  
void docfile(String filename) { ...  
    try {  
        ...  
        FileInputStream fin = new  
            FileInputStream(filename);  
        ...  
    }  
    catch (IOException e) {  
        System.out.println("Loi doc file");  
    }  
}
```

## Kiểm soát Exception (tt)

- Khi có lỗi phương thức sẽ ném ra một exception
- Việc kiểm soát exception giúp chương trình kiểm soát được những trường hợp ngoại lệ và xử lý lỗi.
- Những lỗi không kiểm soát được sẽ có những ảnh hưởng bất lợi trong chương trình.
- Dùng từ khóa throws để chỉ định những loại exception mà phương thức có thể ném ra.
- <tiền tố> <tên phương thức>(<đối số>) throws <các exceptions>



## Kiểm soát Exception (tt)

- Đoạn code có thể sinh ra lỗi cần đặt trong khối lệnh bắt đầu bằng try.
- Đoạn code để kiểm tra, xử lý trong trường hợp có lỗi xảy ra đặt trong khối lệnh catch.

```
try {  
    // Đoạn mã có thể sinh ra lỗi ...  
}  
catch (<kiểu Exception>){  
    // Đoạn mã kiểm soát lỗi  
}
```

## Kiểm soát Exception (tt)

- Khởi lệnh đặt trong finally luôn được thực thi cho dù có Exception hay không.
- Thường dùng để giải phóng tài nguyên

***try {***

*// Đoạn mã có thể sinh ra lỗi ...*

***}***

***Catch (<kiểu Exception>) {*** *// Đoạn mã kiểm soát lỗi*

***}***

***finally {***

*// Đoạn mã luôn luôn được thực thi*

***}***

# Kiểm soát Exception (tt)

```
try {  
    // Khối lệnh trước dòng lệnh sinh ra lỗi  
    // Dòng lệnh sinh ra lỗi (Exception)  
    ...  
}  
catch (<Kiểu Exception>){  
    // Đoạn mã kiểm soát lỗi  
}  
finally { ...  
}
```

Khối lệnh sau dòng lệnh sinh ra lỗi sẽ bị bỏ qua và không thực hiện khi có exception

## Ví dụ kiểm soát Exception chia cho 0 (tt)

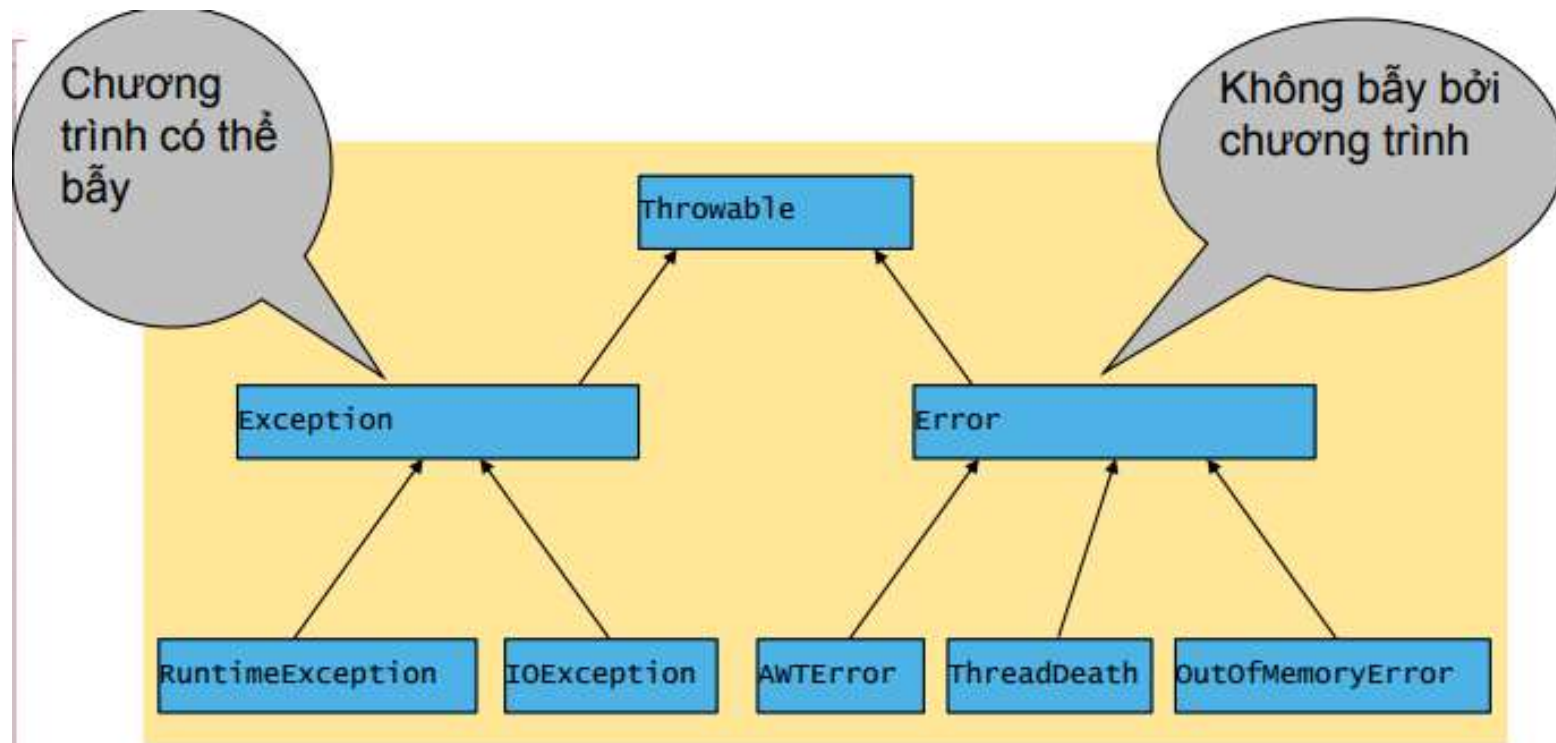
```
import java.io.*;

public class MainClass {
    public static void main(String[] args) {
        try {
            int num_1, num_2;
            BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("\n Nhap so thu 1:");
            num_1 = Integer.parseInt(in.readLine());
            System.out.print("\n Nhap so thu 2:");
            num_2 = Integer.parseInt(in.readLine());
            float rs = num_1/num_2;
            System.out.print("\n Ket qua:" + rs);
        }
    }
}
```

## Ví dụ kiểm soát Exception chia cho 0 (tt)

```
catch (ArithmeticException e) {  
    System.out.print("Loi chia cho 0");  
}  
catch (IOException e) {  
    System.out.print("Loi xuat nhap");  
}  
catch (Exception e) {  
    System.out.print("Loi khac");  
}  
    System.out.print("Kiem soat duoc loi hay Khong co loi");  
}
```

# Phân cấp thư viện của lớp Throwable



- Có thể định nghĩa các exception mới bằng cách dẫn xuất (**extends**) từ những lớp Exception đang có.