

## Chapter 2

# Ngôn ngữ Lập trình Java

CT176 – LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

# Mục tiêu

---

Chương này nhằm giới thiệu các **thành phần cơ bản** của ngôn ngữ lập trình Java, cách **biên dịch** và **thực thi** chương trình và cơ bản về cách xử lý **ngoại lệ** trong Java

# Nội dung

---

- Cấu trúc của một chương trình Java
- Dịch và thực thi một chương trình Java
- Cú pháp của ngôn ngữ Java
- Các kiểu dữ liệu cơ bản trong Java
- Cấu trúc điều khiển

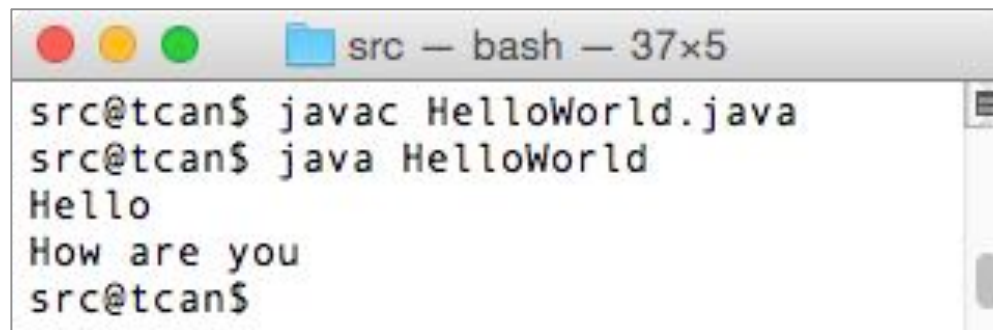
# Cấu trúc một chương trình Java

# Ví dụ 1 – Hello World

- Một chương trình Java hiển thị câu chào hỏi ra màn hình:

```
/* HelloWorld.java */  
  
public class HelloWorld {  
    public static void main(String args[]) {  
        System.out.println("Hello");  
        System.out.println("How are you");  
    }  
}
```

- Kết quả thực thi chương trình:



```
src — bash — 37x5  
src@tcan$ javac HelloWorld.java  
src@tcan$ java HelloWorld  
Hello  
How are you  
src@tcan$
```

# Cấu trúc một chương trình Java

chú thích

```
/* HelloWorld.java */
```

```
public class HelloWorld {  
    public static void main(String args[]) {  
        System.out.println("Hello!");  
        System.out.println("How are you?");  
    }  
}
```

tên chương trình  
(phải giống tên tập tin,  
không bao gồm phần mở rộng)

chương trình chính  
(điểm bắt đầu, entry  
point, của chương trình)

các lệnh trong chương trình  
lệnh System.out.println() dùng để hiển thị một chuỗi ra màn hình

## • Chú ý:

- tên chương trình và tên tập tin phải giống nhau
- hàm `main()` có chức năng giống như hàm `main()` trong C

## Ví dụ 2 – Chương trình có nhiều hàm

chương trình chính

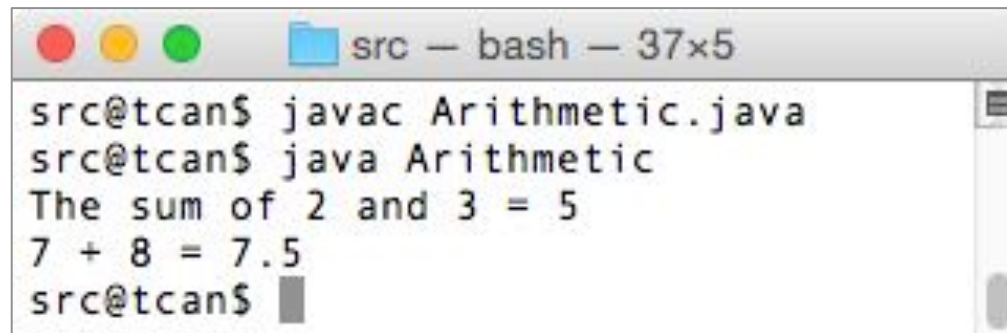
```
public class Arithmetic {  
    public static void main(String[] args) {  
        System.out.println("The sum of 2 and 3 = " + 5);  
        System.out.println("7 + 8 = " + avg(7, 8));  
    }  
}
```

gọi hàm

```
public static float avg(float a, float b) {  
    return (a + b)/2;  
}
```

hàm tính trung bình hai số avg()

- Kết quả thực thi chương trình:



```
src — bash — 37x5  
src@tcan$ javac Arithmetic.java  
src@tcan$ java Arithmetic  
The sum of 2 and 3 = 5  
7 + 8 = 7.5  
src@tcan$
```

## Ví dụ 3 – Giao diện đồ họa

```
import javax.swing.JFrame;
public class MyGUIApp {
    public static void main(String[] args) {
        JFrame myWindow;
        myWindow = new JFrame();
        myWindow.setSize(300, 200);
        myWindow.setTitle("My First Java Program");
        myWindow.setVisible(true);
    }
}
```

Khai báo sử dụng lớp JFrame



- Lệnh import: khai báo sử dụng các lớp (thư viện lớp) từ bên ngoài



# Dịch và thực thi một chương trình Java

# Đặc điểm của Java

- Java là một ngôn ngữ lập trình **vừa thông dịch, vừa biên dịch.**
  - Chương trình Java, sau khi phát triển xong sẽ được **biên dịch** (compile) sang dạng bytecode bằng **trình biên dịch Java**.
  - Khi cần thực thi một chương trình bytecode, **máy ảo Java** sẽ **thông dịch** từng lệnh bytecode sang mã máy.
- Chương trình Java có tính **đa nền** (multi-platform): có thể thực thi trên nhiều kiến trúc máy tính và hệ điều hành khác nhau nhờ vào cơ chế **thông dịch**.

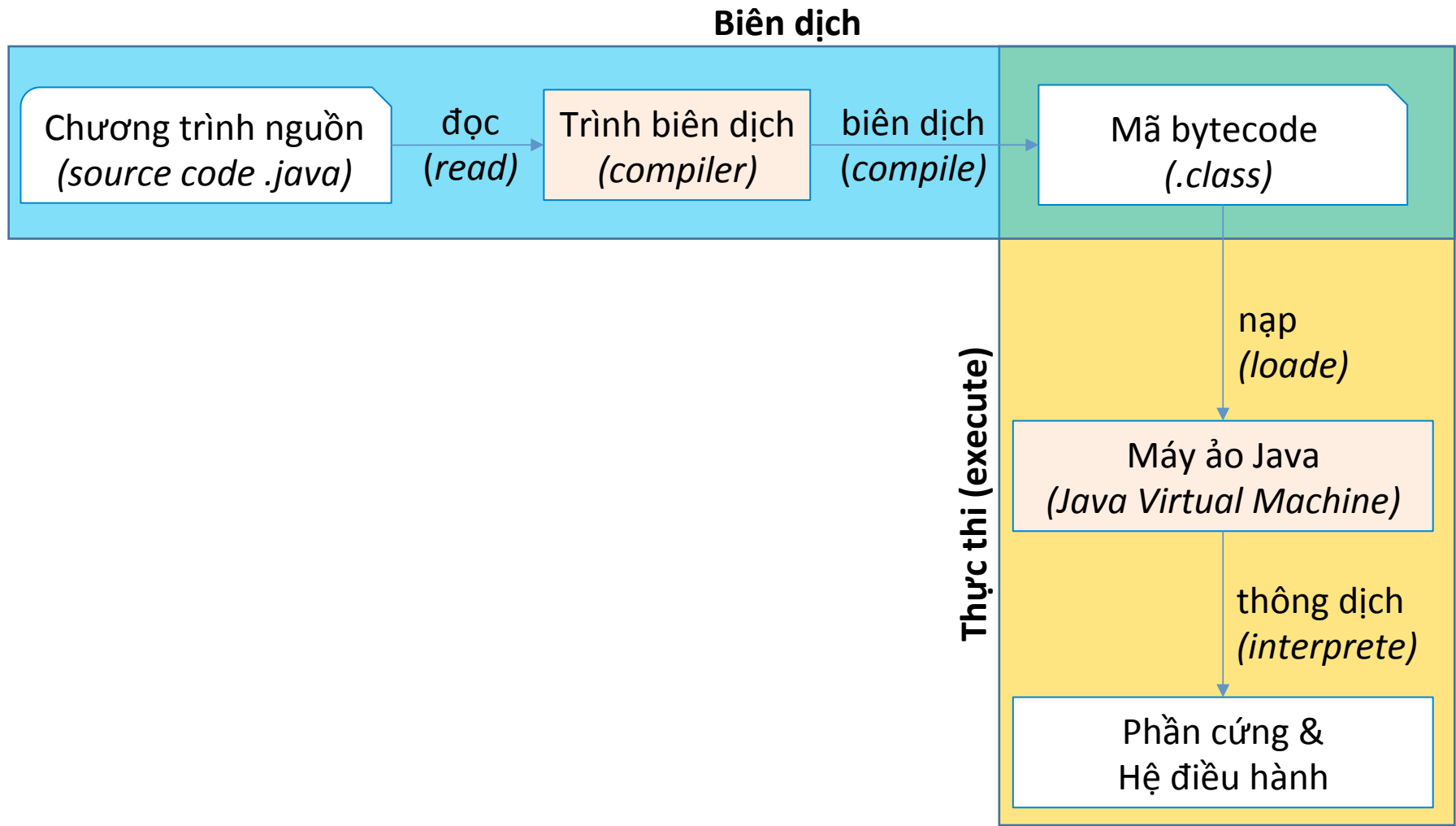
# Quá trình phát triển 1 chương trình Java

- Người lập trình **viết chương trình** Java:
  - Bao gồm 1 tập các **câu lệnh** (statements)
  - Dùng công cụ soạn thảo văn bản hay môi trường lập trình IDE
  - Lưu trong các tập tin có phần mở rộng **.java**
  - Được gọi là các **chương trình nguồn** (source code)
- Trình biên dịch Java **biên dịch** các chương trình nguồn:
  - Thành các chương trình dạng **bytecode**
  - Được lưu trong các tập tin với phần mở rộng **.class**
  - Các **lỗi cú pháp** nếu có, sẽ được sinh ra

# Quá trình phát triển 1 chương trình Java

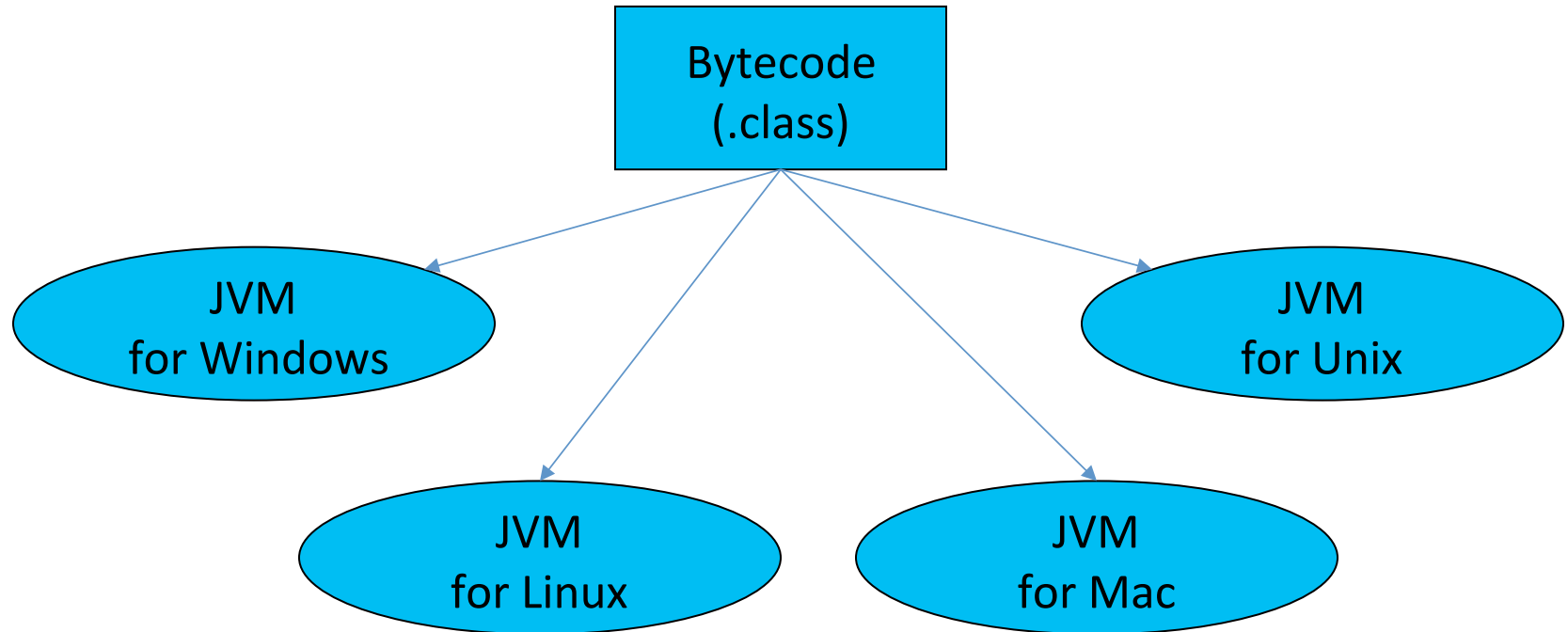
- Máy ảo Java sẽ **thực thi** các chương trình bytecode:
  - Bộ nạp (loader) sẽ **nạp** chương trình bytecode vào JVM
  - JVM sẽ **thông dịch** các lệnh trong chương trình bytecode ra mã máy ở nền tảng tương ứng để thực thi
- Máy ảo Java:
  - Hoạt động như là 1 **máy tính ảo**: thực thi các mã bytecode (vs. CPU là máy tính “thật”, thực thi các mã máy do JVM thông dịch ra)
  - Mã bytecode là **giống nhau** đối với JVM trên tất cả các nền tảng (hệ điều hành)  $\Rightarrow$  Các JVM trên từng nền tảng sẽ dịch mã bytecode sang mã máy ở nền tảng tương ứng

# Quá trình phát triển 1 chương trình Java



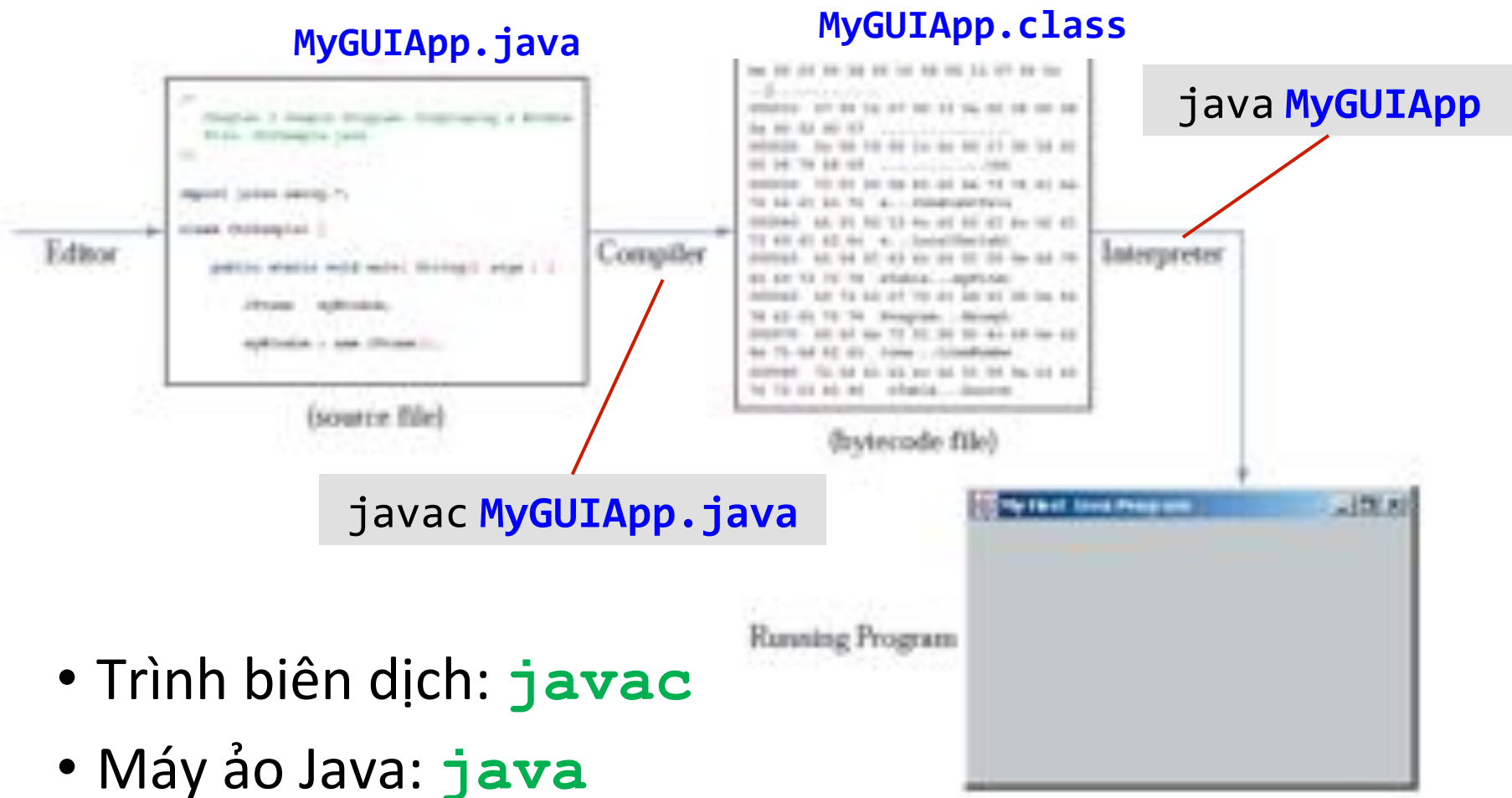
# Quá trình phát triển 1 chương trình Java

- Máy ảo Java – tính khả chuyển:



**Why Java bytecode?**

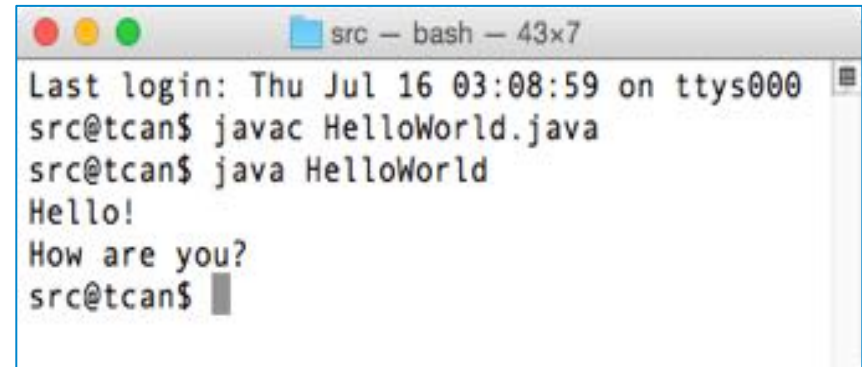
# Dịch và thực thi 1 chương trình Java



- Trình biên dịch: **javac**
- Máy ảo Java: **java**

# Môi trường dịch và thực thi chương trình

- Giao diện dòng lệnh:
  - Unix + Mac OS: Terminal
  - Windows: Command Prompt



```
src — bash — 43x7
Last login: Thu Jul 16 03:08:59 on ttys000
src@tcan$ javac HelloWorld.java
src@tcan$ java HelloWorld
Hello!
How are you?
src@tcan$
```



```
Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Tran Cong AN>javac HelloWorld.java

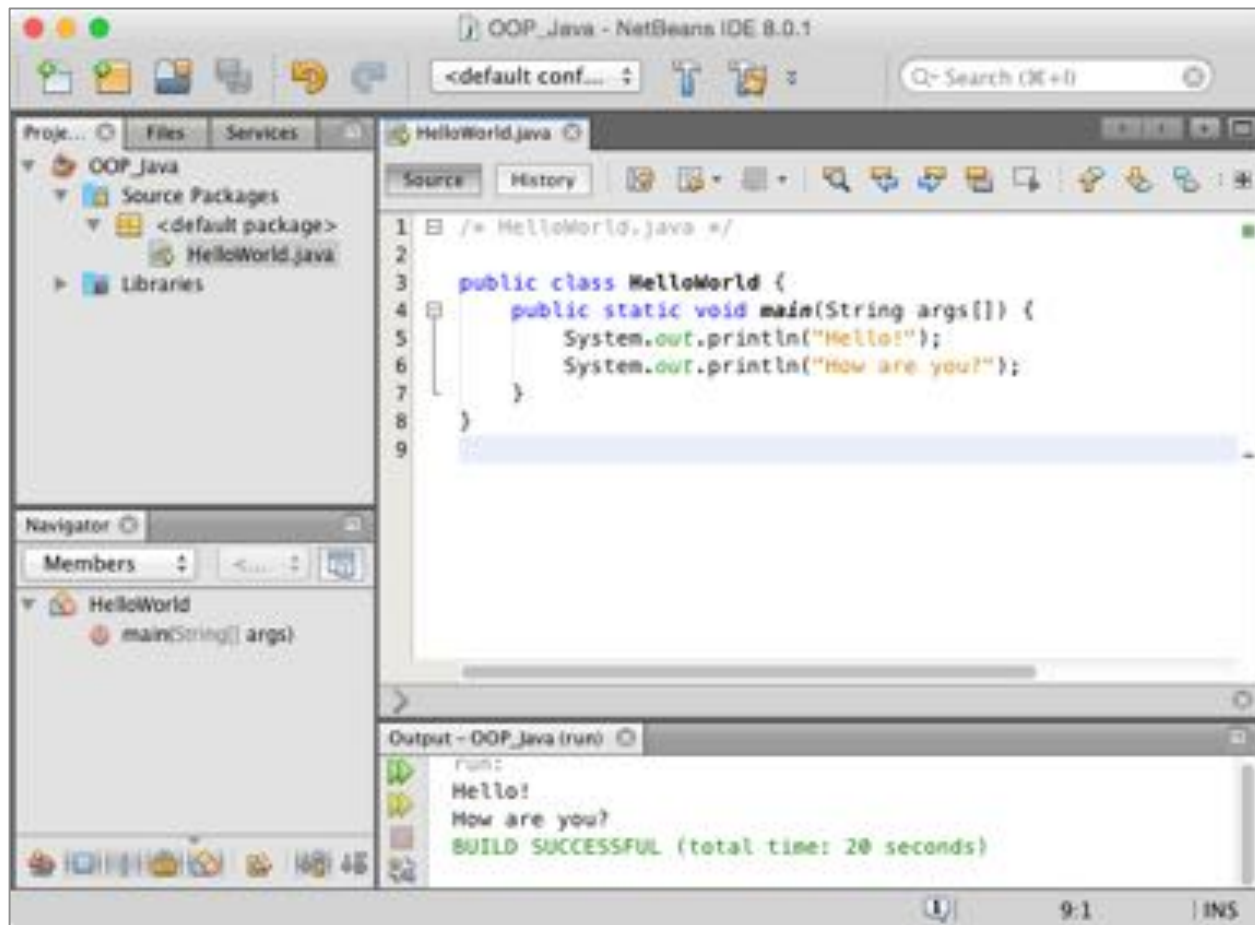
C:\Users\Tran Cong AN>java HelloWorld
Hello!
How are you?

C:\Users\Tran Cong AN>
```



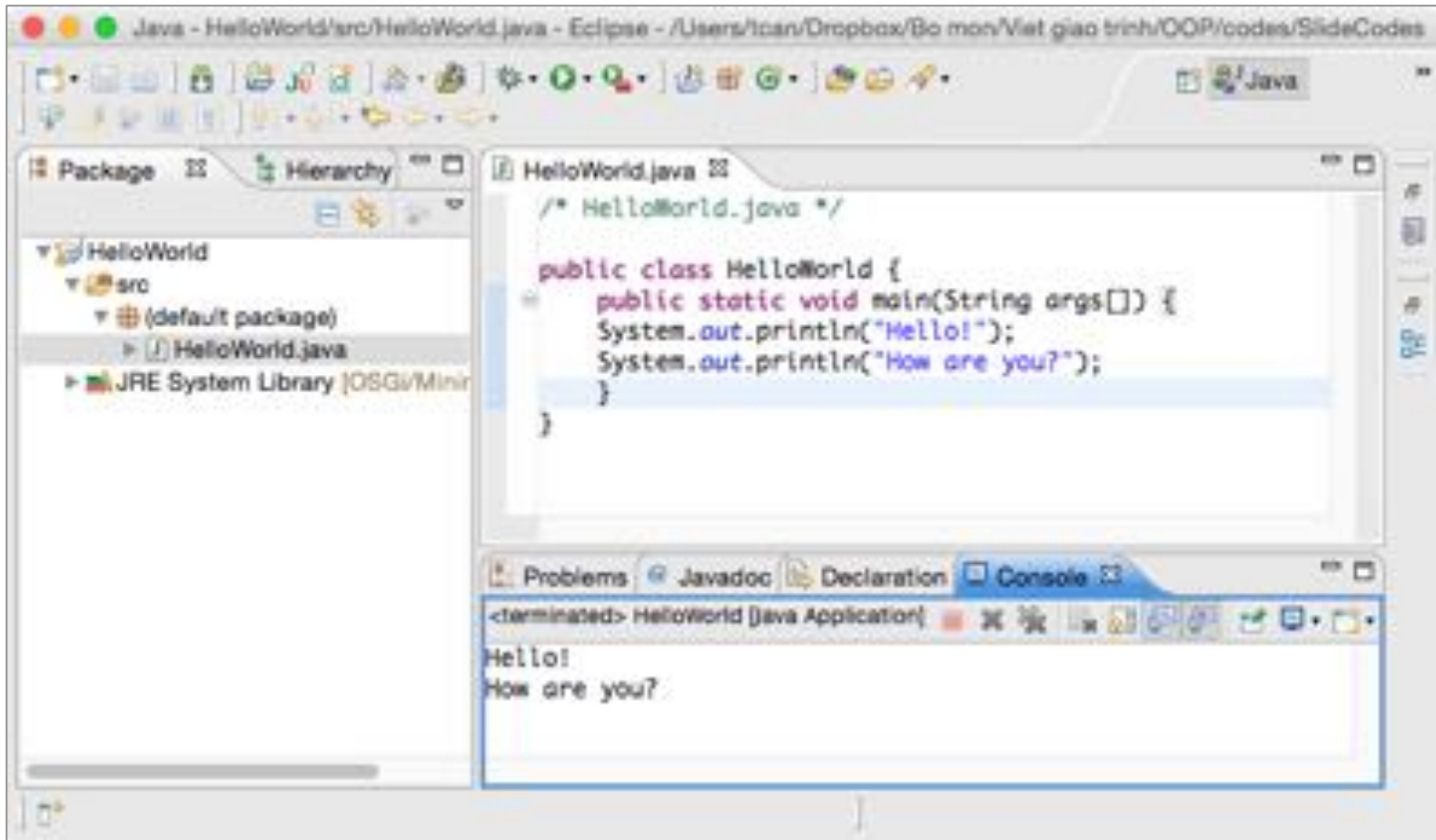
# Môi trường dịch và thực thi chương trình

- Môi trường phát triển tích hợp: **Netbean**, Eclipse,...



# Môi trường dịch và thực thi chương trình

- Môi trường phát triển tích hợp: Netbean, **Eclipse**,...



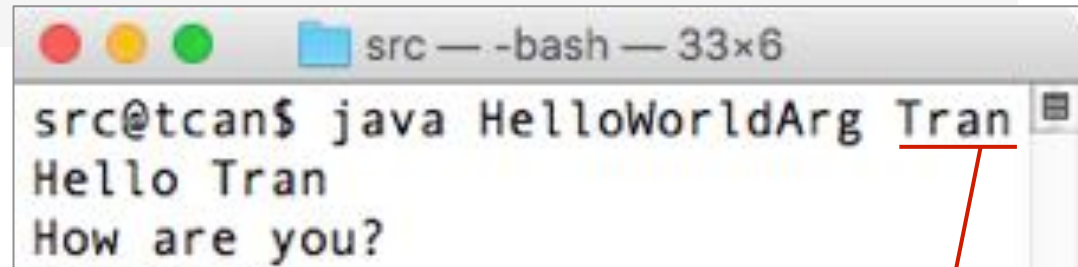
# Đối số dòng lệnh (command line argument)

- Khi gọi thực thi một chương trình Java, ta có thể truyền vào các **đối số** (dữ liệu) **từ dòng lệnh** cho chương trình
  - Cú pháp: `java <tên chương trình> [danh sách đối số]`
  - Các đối số cách nhau bằng khoảng trắng
  - Nếu giá trị của đối số có khoảng trắng thì bao giá trị của đối số bằng cặp dấu nháy "
- Giá trị của các đối số dòng lệnh sẽ được truyền vào cho đối số `args` của hàm `main(String args[])`
- Chỉ số của các đối số bắt đầu từ 0: `args[0]`, `args[1]`, ...

# Đối số dòng lệnh (command line argument)

```
/* HelloWorld.java */
```

```
public class HelloWorldArg {  
    public static void main(String args[]) {  
        System.out.println("Hello " + args[0]);  
        System.out.println("How are you?");  
    }  
}
```



A terminal window titled 'src — -bash — 33x6' showing the command `src@tcan$ java HelloWorldArg Tran` and its output: `Hello Tran` and `How are you?`. A red line underlines the argument 'Tran' in the command.



A terminal window titled 'src — -bash — 39x6' showing the command `src@tcan$ java HelloWorldArg "Cong Huy"` and its output: `Hello Cong Huy` and `How are you?`. A red line underlines the argument 'Cong Huy' in the command.

tham số

# Thiết lập môi trường phát triển

- Cài đặt JDK (Java Development Kit):
  1. JDK download link : <http://goo.gl/IOEB4I>
  2. Chọn bộ cài đặt thích hợp (Windows, Linux, Mac OS,...)
  3. Chạy bộ cài đặt theo hướng dẫn
  4. Kiểm tra việc cài đặt: Thực thi các lệnh sau từ dòng lệnh
    - Trình biên dịch Java: `javac -version`
    - Máy ảo Java: `java -version`

```
C:\Program Files\Java\jdk1.7.0_67\bin>javac -version
javac 1.7.0_67

C:\Program Files\Java\jdk1.7.0_67\bin>java -version
java version "1.7.0_67"
Java(TM) SE Runtime Environment (build 1.7.0_67-b01)
Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04, mixed mode)
```

- Bộ cài đặt JDK **bao gồm cả JVM**

# Thiết lập môi trường phát triển

- Thiết lập môi trường phát triển bằng tay (manually):
  1. Copy thư mục chứa JDK vào máy tính (từ máy đã cài đặt)
  2. Chọn Computer / Properties/ Advanced System Settings / Advanced / Environment Variables...
  3. Click New:
    - Variable name: `JAVA_HOME`
    - Variable value: nhập vào đường dẫn chứa JDK
  4. Chọn biến `path` trong User variable for USER
  5. Click Edit và thêm vào ô Variable Values:  
`;%JAVA_HOME%/BIN;.;`
  6. Nhấn OK để đóng tất cả các hộp thoại
  7. Kiểm thử việc cài đặt (tương tự slide trước)

<https://duythanhcse.wordpress.com/2011/12/20/cach-thiet-lap-bien-moi-truong/>

# Các thành phần cơ bản của Java

# Câu lệnh & Chú thích (Statement & Comment)

- Một **câu lệnh** (statement) Java kết thúc bằng dấu **;**
- Các lệnh và các định danh (identifier) phân biệt chữ hoa, chữ thường (case sensitive)
- Ví dụ:  

```
System.out.println("Hello!");
```

```
System.out.Println("How are you?"); -> lỗi
```
- Có hai loại **chú thích** trong Java (giống ngôn ngữ C):
  - Trên một dòng: `// chú thích`
  - Nhiều dòng: `/* chú thích */`
- Một **khối lệnh** (block) được bao bởi cặp ngoặc nhọn **{ }**



# Biến và Định danh (Variable & Identifier)

- **Biến** là một vùng nhớ được đặt tên, dùng để trữ dữ liệu xử lý trong chương trình
  - Khai báo biến:  
`<kiểu dữ liệu> <tên biến>;`  
`<kiểu dữ liệu> <tên biến 1> [<, tên biến 2>...];`
- **Tên biến** được đặt theo qui tắc đặt tên định danh:
  - Có thể chứa các **ký tự** (A-Z, a-z), **số** (0-9), dấu `_` và `$`
  - Ký tự đầu tiên không được là 1 số
  - Không được trùng với các từ khóa (keywords) của Java
  - Phân biệt chữ hoa và chữ thường
  - **Ví dụ:** `itemOrdered`, `noOfStudent`

# Biến và Định danh (Variable & Identifier)

---

- Ngoài các qui tắc đặt tên định danh, cần tham khảo thêm các **qui ước** đặt tên (**naming convention**):
  - Tên biến phải mang ý nghĩa
  - Tên biến trong Java thường dùng **Pascal case**: từ đầu tiên trong tên biến được viết thường (lower case), các từ sau viết hoa chữ cái đầu tiên (title case).
  - Thông thường, tên biến bao gồm các danh từ hoặc cụm danh từ
  - Các qui ước viết chương trình (coding convention) cho Java:  
<http://java.sun.com/docs/codeconv/html/CodeConventions.doc8.html>

# Phạm vi của biến

---

- Phạm vi của biến: các vị trí trong chương trình có thể truy xuất được biến
  - Một biến được khai báo trong khối lệnh nào thì chỉ được truy xuất bên trong khối lệnh đó
  - Phạm vi của biến là từ câu lệnh khai báo biến cho đến cuối khối lệnh chứa khai báo biến
  - Một biến sẽ bị hủy (vùng nhớ dành cho biến bị thu hồi) khi chương trình thực thi ra khỏi phạm vi của biến

# Phạm vi của biến

- Có thể khai báo ở bất kỳ vị trí nào trong chương trình
- Biến toàn cục: phạm vi toàn chương trình
- Biến cục bộ: phạm vi nó được khai báo
- Ví dụ:

```
int so = 5;
void GanSo ( int x) {
    so = x;
}
int NuaSo ( int x) {
    int c = 2;
    so = x/c ;
    return so;
}
```

# Phạm vi của biến

- Biến trong khối lệnh lồng nhau không được trùng tên

- Ví dụ:

```
class NestVar {  
    public static void main (String args[]) {  
        int count;  
        for (count = 0; count < 10; count = count+1) {  
            System.out.println ("This is count: " + count);  
            int count; // illegal!!!  
            for (count = 0; count < 2; count++)  
                System.out.println ( "This program is in error!");  
        }  
    }  
}
```

# Từ khóa (Keywords)

- Từ khóa là các từ dành riêng cho Java:

abstract	double	instanceof	static
assert	else	int	super
boolean	enum	interface	switch
break	extends	long	synchronized
byte	false	native	this
case	for	new	throw
catch	final	null	throws
char	finally	package	transient
class	float	private	true
const	goto	protected	try
continue	if	public	void
default	implements	return	volatile
do	import	short	while

# Các kiểu dữ liệu nguyên thủy

- Java có 8 kiểu dữ liệu nguyên thủy (primitive datatype):
  - Kiểu số nguyên:
    - byte
    - short
    - int
    - long
  - Kiểu số thực:
    - float
    - double
  - Kiểu luận lý: `boolean`
  - Kiểu ký tự: `char`
  - Kiểu chuỗi ký tự: `String`

# Kiểu dữ liệu số (numeric datatype)

Kiểu DL	Kích thước	Miền giá trị
byte	1 byte	-128 to +127
short	2 bytes	-32,768 to <u>+32,767</u>
int	4 bytes	-2,147,483,648 to +2,147,483,647
long	8 bytes	-9,223,372,036,854,775,808 to <u>+9,223,372,036,854,775,807</u>
float	4 bytes	$\pm 3.410 \cdot 10^{-38}$ to <u><math>\pm 3.41038</math></u> , with 7 digits of accuracy
double	8 bytes	$\pm 1.710 \cdot 10^{-308}$ to <u><math>\pm 1.710308</math></u> , with 15 digits of accuracy

- **Kích thước:** dung lượng bộ nhớ được cấp phát cho 1 biến có kiểu tương ứng.



# Kiểu dữ liệu số (numeric datatype)

- Hằng giá trị số (numeric literal):

- Số nguyên: 0, 5, 10, 12, -25, -30,...
- Số thực: các hằng giá trị số thực có kiểu mặc nhiên là `double`
- Ví dụ: 

```
int a;           // Khai báo biến a kiểu int
float b;         // Khai báo biến b kiểu float
a = 123;
b = 123.5;       // Sai!
```

Giá trị kiểu `double` không tương thích với biến số kiểu `float` vì độ chính xác khác nhau (7 vs. 15 số lẻ).

⇒ `b = 123.4F;`

- Có thể dùng cách biểu diễn bằng ký hiệu **E** (E-notation):  
`number = 1.234E2F;`

# Kiểu dữ liệu số (numeric datatype)

```
// This program has variables of several of the integer types.
public class IntegerVariables {
    public static void main(String[] args) {
        int checking;    // Declare an int variable named checking.
        byte miles;      // Declare a byte variable named miles.
        short minutes;   // Declare a short variable named minutes.
        long days;       // Declare a long variable named days.

        checking = -20;
        miles = 105;
        minutes = 120;
        days = 185000;
        System.out.println("We've made a journey of " + miles +
                           " miles.");
        System.out.println("It took us " + minutes + " minutes.");
        System.out.println("Our account balance is $" + checking);
    }
}
```

# Kiểu dữ liệu số (numeric datatype)

```
// This program demonstrates the double data type.
```

```
public class Sale
{
    public static void main(String[] args)
    {
        double price, tax, total;

        price = 29.75;
        tax = 1.76;
        total = 31.51;
        System.out.println("The price of the item " +
                           "is " + price);
        System.out.println("The tax is " + tax);
        System.out.println("The total is " + total);
    }
}
```

# Kiểu dữ liệu luận lý (boolean datatype)

---

- Giá trị luận lý có thể mang 1 trong 2 giá trị:
  - `true`
  - `false`
- Một biến luận lý chỉ có thể được gán 1 trong hai giá trị này
- Thông thường, các biến/giá trị luận lý được sử dụng trong các câu lệnh điều kiện (conditional statement) và lặp (loop)

# Kiểu dữ liệu luận lý (boolean datatype)

*// A program for demonstrating boolean variables*

```
public class TrueFalse
{
    public static void main(String[] args)
    {
        boolean bool;

        bool = true;
        System.out.println(bool);
        bool = false;
        System.out.println(bool);
    }
}
```

# Kiểu dữ liệu ký tự (char datatype)

- Kiểu dữ liệu ký tự cho phép thao tác trên **1 ký tự**
- Một hằng giá trị kiểu ký tự được bao trong một cặp dấu ngoặc đơn ` `
  - Ví dụ: ``a'`, ``Z'`, ``\n'`, ``\t'`, ``2'`
- Mỗi ký tự có một mã ký tự (character code):
  - Ví dụ: ký tự ``a'` có mã là 65, ``b'` có mã là 66,...
  - Khi thao tác trên ký tự, ta có thể sử dụng ký tự được bao trong dấu ngoặc đơn hoặc mã ký tự.
  - Mã ký tự có giá trị từ 0 – 65.335 ( $2^{16} - 1$ )
- Kích thước của mỗi ký tự là 2 bytes (Unicode)

# Kiểu dữ liệu ký tự (char datatype)

*// This program demonstrates the char data type.*

```
public class Letters
{
    public static void main(String[] args)
    {
        char ch;

        ch = 'A';
        System.out.println(ch);
        ch = 66;           //ch = 'B';
        System.out.println(ch);
    }
}
```

# Gán giá trị cho biến

- Để gán giá trị cho biến, ta dùng toán tử gán =

**<biến> = <biến | hằng giá trị | biểu thức>**

```
// This program shows variable assignment
```

```
public class Initialize {  
    public static void main(String[] args) {  
        int month, days;  
  
        month = 2;  
        days = 28;  
        System.out.println("Month " + month + " has " +  
            days + " days.");  
    }  
}
```



# Khởi tạo giá trị cho biến

- Một biến có thể được khởi tạo giá trị ngay khi khai báo  
**<kiểu DL> <tên biến> [= giá trị];**

```
// This program shows variable initialization

public class Initialize {
    public static void main(String[] args) {
        int month = 2, days = 28;

        System.out.println("Month " + month + " has " +
                           days + " days.");
    }
}
```

- **Lưu ý:** biến phải được khởi tạo hoặc gán giá trị trước khi được truy xuất giá trị.

# Toán tử toán học (arithmetic operators)

Toán tử	Ý nghĩa	Ví dụ
+	Cộng	<code>total = cost + tax;</code>
-	Trừ Trừ một ngôi	<code>cost = total - tax;</code> <code>a = -b;</code>
*	Nhân	<code>tax = cost * rate;</code>
/	Chia	<code>salePrice = original / 2;</code>
%	Chia dư	<code>remainder = value % 5;</code>
++	Tăng	<code>a++;</code> <code>++a;</code>
--	Giảm	<code>a--;</code> <code>--a;</code>

- Toán tử `/` sẽ là toán tử **chia nguyên** nếu cả hai toán hạng đều là kiểu số nguyên, ngược lại sẽ là phép **chia thực**.
- Độ **ưu tiên** của các toán tử tương tự như **ngôn ngữ C**

# Các toán tử khác

- Toán tử so sánh (comparisons):

==      !=      <      <=      >      >=

- Toán tử luận lý (boolean operators):

&&      ||      !

- Toán tử trên bit (bitwise operators):

&      |      ^

- Toán tử kết hợp (combined operators): kết hợp các toán tử toán học và phép gán vào cùng 1 toán tử.

+=      -=      \*=      /=      %=  
&=      |=      ^=

# Biểu thức (expression)

- Biểu thức là một sự kết hợp giữa toán tử với các biến, hằng hay các biểu thức khác.

- Ví dụ:

`-5`

`8 - 7`

`2 + 3 * 5`

`(b * b) + (4 * a * c)`

} Biểu thức số học

`(month > 0) && (month <= 12)`

`(year % 100) == 0`

} Biểu thức luận lý

# Biểu thức (expression)

- Chú ý quy tắc chuyển đổi kiểu của biểu thức toán học:
  - Nếu tất cả các toán hạng là kiểu nguyên thì biểu thức trả về kiểu nguyên
  - Nếu có ít nhất một toán hạng kiểu số thực thì biểu thức trả về kiểu thực
  - Biểu thức được định giá theo thứ tự ưu tiên của các toán tử
- Ví dụ:
  - $(3/2 + 4) / 2 = (1 + 4) = 2$
  - $(3/2 + 4.0) / 2 = (1 + 4.0) / 2 = 5.0 / 2 = 2.5$
  - $14.6 / 2 + 5 = 7.3 + 5 = 12.3$

# Ép kiểu (type casting)


- Ép kiểu dữ liệu (type casting) của một giá trị từ kiểu này sang kiểu khác.
  - Ép kiểu tự động (implicit): Java tự động ép kiểu các toán hạng trong một biểu thức khi có sự không tương thích về kiểu
  - Ép kiểu tường minh (explicit): người lập trình yêu cầu ép kiểu một cách tường minh
- Cú pháp: **(kiểu dữ liệu) <biểu thức>**
  - `"a" = " + 3 = "a = " + "3" = "a = 3"`
  - `3 / 2 + 4.0 = 1 + 4.0 = 1.0 + 4.0 = 5.0`
  - `(float) 3 / 2 + 4.0 = 1.5 + 4.0 = 5.5`
  - `(int) 11 * 0.3 = ?` ~~11~~ ~~×~~ ~~0.3~~

# Ép kiểu (type casting)

- Java API cũng cung cấp một số hàm để chuyển đổi kiểu
- Một số hàm chuyển đổi kiểu thông dụng:
  - `int` `Integer.parseInt(String s)`: trả về giá trị số nguyên tương ứng với một chuỗi số.
  - `float` `Float.parseFloat(String s)`: trả về giá trị số thực (float) tương ứng với một chuỗi số.
  - `double` `Double.parseDouble(String s)`: trả về giá trị số thực (double) tương ứng với một chuỗi số.
  - `String` `Integer.toString(int a)`: trả về một chuỗi tương ứng với một số nguyên
  - ...

# Ép kiểu (type casting)

```
public class TypeCasting {  
    public static void main(String[] args) {  
        System.out.println("(int)(7.9) = " + (int)(7.9));  
        System.out.println("(int)(3.3) = " + (int)(3.3));  
        System.out.println("(double)(5 + 3) = " + (double)(5 + 3));  
        System.out.println("(double)(15)/2 = " + ((double)(15)/2));  
        System.out.println("(double)(15/2) = " + ((double)(15/2)));  
        System.out.println("(int)(7.8 + (double)(15)/2) = " +  
            ((int)(7.8 + (double)(15)/2)));  
        System.out.println("(int)(7.8 + (double)(15/2)) = " +  
            ((int)(7.8 + (double)(15/2))));  
    }  
}
```





# Kiểu dữ liệu chuỗi (String datatype)

- Một chuỗi được xem như là một dãy các ký tự
- Một hằng chuỗi ký tự được bao trong cặp ngoặc kép `""`
  - Ví dụ: `"Hello World"`, `"Chào bạn"`
- Mỗi ký tự trong chuỗi có 1 vị trí với vị trí của ký tự đầu tiên của chuỗi được đánh chỉ số từ 0
- Kiểu dữ liệu chuỗi trong Java: `String`
- Toán tử trên chuỗi: `+` (cộng, ghép chuỗi)  
`System.out.println("The sum = " + 12 + 26);`
- **Lưu ý:** `String` là một lớp  $\Rightarrow$  hỗ trợ nhiều phương thức để thao tác trên chuỗi, sẽ được giới thiệu sau.

# Kiểu dữ liệu chuỗi (String datatype)

```
// This program demonstrates a few of the String methods.
```

```
public class StringMethods {  
    public static void main(String[] args) {  
        String message = "Java is Great Fun!";  
        String upper = message.toUpperCase();  
        String lower = message.toLowerCase();  
        char letter = message.charAt(2);  
        int stringSize = message.length();  
  
        System.out.println(message);  
        System.out.println(upper);  
        System.out.println(lower);  
        System.out.println(letter);  
        System.out.println(stringSize);  
    }  
}
```

# Nhập/Xuất căn bản

# Xuất dữ liệu ra màn hình

---

- `System.out.println(String s)`: xuất + xuống dòng
- `System.out.print(String s)`: xuất, không xuống dòng
- `System.out.printf(String format, Object... args)`: hiển thị dữ liệu có định dạng, tương tự như hàm `printf()` của C

```
System.out.printf("Phuong trinh co nghiem %.2f",  
                  -(float)b/a);
```

- Cú pháp định dạng có thể tham khảo tại:

<https://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html#syntax>

# Đọc dữ liệu từ bàn phím

- Muốn nhập dữ liệu từ bàn phím, ta dùng lớp `Scanner`, kết hợp với `System.in` như sau:
  - Tạo một Scanner:

```
Scanner keyboard = new Scanner(System.in);
```
  - Lớp `Scanner` được định nghĩa trong `java.util`, vì vậy ta phải thêm vào lệnh sau ở đầu chương trình:

```
import java.util.Scanner;
```
  - Đọc dữ liệu từ bàn phím: dùng các hàm
    - `String nextLine()`: đọc một chuỗi ký tự
    - `int nextInt()`: đọc một số nguyên kiểu `int`
    - `long nextLong()`: đọc một số nguyên kiểu `long`
    - `float nextFloat()`: đọc một số thực kiểu `float`
    - ...

# Đọc dữ liệu từ bàn phím

```
import java.util.Scanner;

public class PTB1 {
    public static void main(String args[]) {
        //Tạo đối tượng thuộc Lớp Scanner để nhập dữ liệu từ bàn phím
        Scanner keyboard = new Scanner(System.in);

        int a, b;
        System.out.print("a = ");
        a = keyboard.nextInt();           //đọc một số nguyên
        System.out.print("b = ");
        b = keyboard.nextInt();
        System.out.printf("Nghiem cua PT x = %.2f", -(float)b/a);
    }
}
```

- **Lưu ý:** g/sử a, b khác 0 và phương trình luôn có nghiệm

# Đọc dữ liệu từ bàn phím

- Chú ý tr/hợp dữ liệu còn sót trong bộ đệm bàn phím

```
import java.util.Scanner;
public class ScannerFlush {
    public static void main(String args[]) {
        //Tạo đối tượng thuộc Lớp Scanner để nhập dữ liệu từ bàn phím
        Scanner keyboard = new Scanner(System.in);

        String name;
        long ID;
        System.out.print("Enter your ID: ");
        ID = keyboard.nextLong();

        • ← keyboard.nextLine();
        System.out.print("Enter your name: ");
        name = keyboard.nextLine();

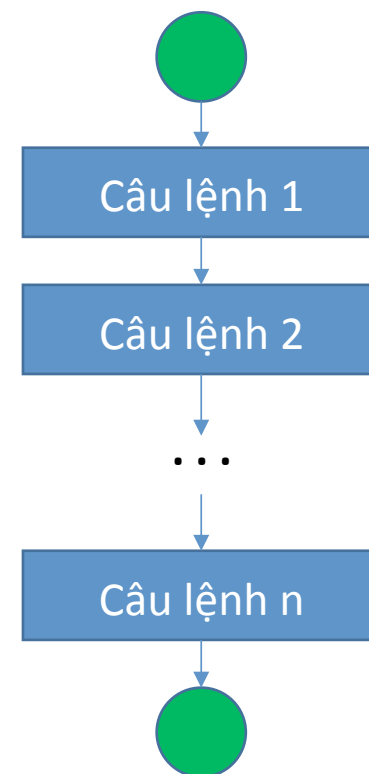
        System.out.println("ID :" + ID + ", name: " + name);
    }
}
```

# Cấu trúc điều khiển trong Java



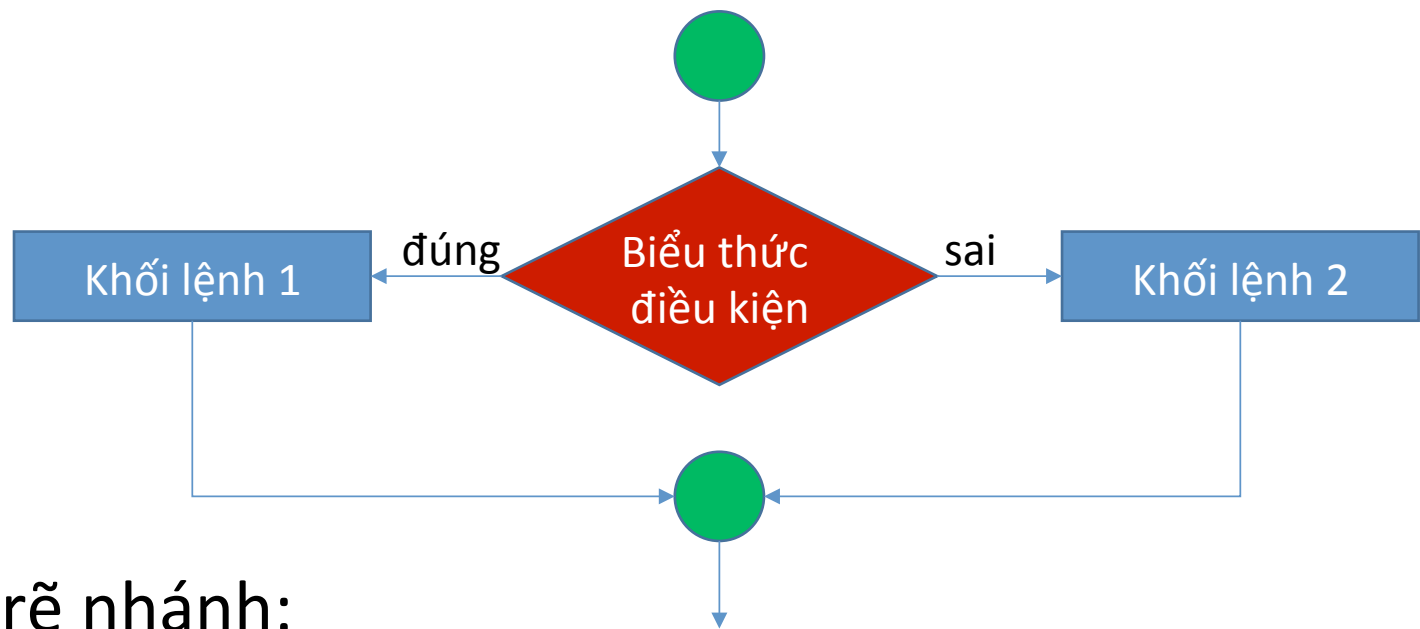
# Cấu trúc điều khiển (control structure)

- Cấu trúc điều khiển: điều khiển cách thức thực hiện các lệnh trong chương trình.
- Có 3 cấu trúc điều khiển:
  - Tuần tự (sequence, **mặc nhiên**)
  - Lựa chọn (selection)
  - Lặp (repetition)



# Cấu trúc rẽ nhánh (selection structure)

- Lựa chọn 1 trong 2 công việc (khối lệnh) để thực hiện dựa trên 1 điều kiện cho sẵn.



- Các lệnh rẽ nhánh:
  - `if ... else`
  - `switch ... case`

# Lệnh rẽ nhánh `if ... else`

---

- Lệnh `if` đầy đủ:

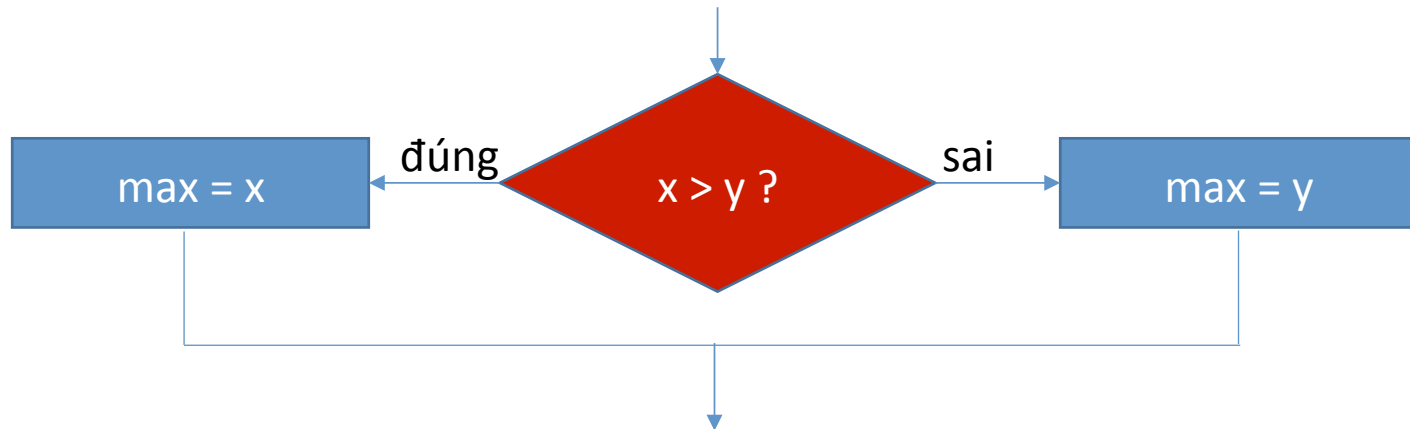
```
if (biểu thức điều kiện) {  
    //statement T;  
}  
else {  
    //statement F;  
}
```

- Nếu điều kiện đúng, thực hiện `statement T`, ngược lại thực hiện `statement F`.
- `Statement T/F` có thể là một hoặc nhiều câu lệnh
- Trong trường hợp chỉ có 1 câu lệnh thì không cần cặp dấu ngoặc nhọn

# Lệnh rẽ nhánh `if ... else`

- Ví dụ: tìm giá trị lớn nhất (max):

```
if (x > y) {  
    max = x;  
}  
else {  
    max = y;  
}
```



# Lệnh rẽ nhánh `if ... else`

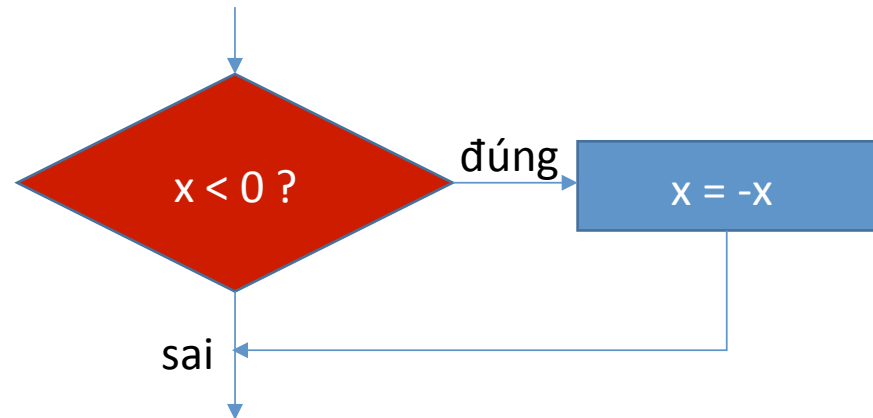
- Lệnh rẽ nhánh `if` không có mệnh đề `else`:

```
if (biểu thức điều kiện) {  
    statement T;  
}
```

- Nếu điều kiện đúng, thực hiện `statement T`

Tìm giá trị tuyệt đối

```
if (x < 0) {  
    x = -x;  
}
```

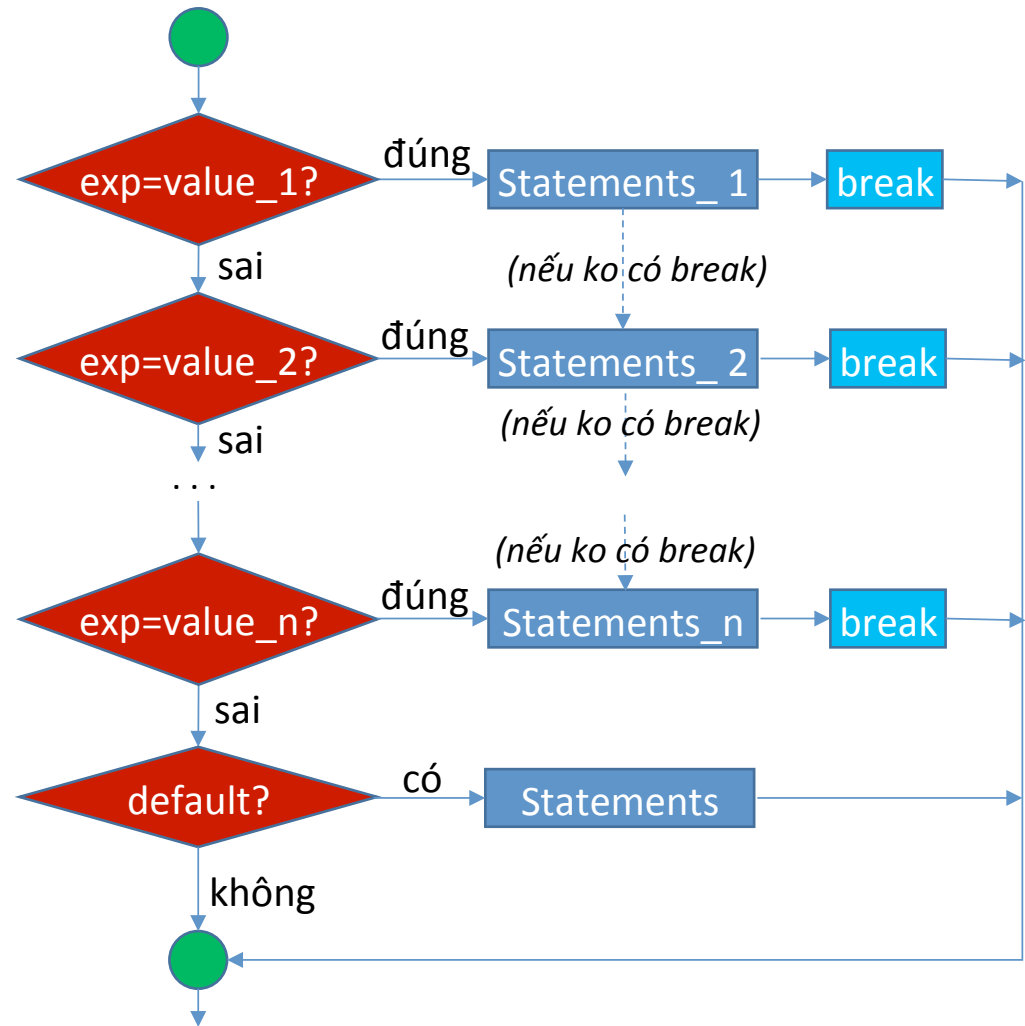


# Lệnh rẽ nhánh if ... else

```
import java.util.Scanner;
```

# Lệnh lựa chọn switch ... case

```
switch (exp) {  
  case value1:  
    statements_1;  
    break;  
  case value_2:  
    statements_2;  
    break;  
  ...  
  case value_n:  
    statements_n;  
    break;  
  default:           } Không bắt buộc  
    statements;  
}
```



# Lệnh lựa chọn `switch ... case`

- Giải thích:

- `exp` phải là một biểu thức có giá trị **kiểu số nguyên** hoặc kiểu **ký tự** (từ Java 7 trở đi có thể là kiểu chuỗi)
- **Mệnh đề `case`** nào có giá trị bằng với giá trị của `exp`, thì các câu lệnh từ mệnh đề `case` đó cho đến hết lệnh `switch` sẽ được thực hiện, hoặc cho đến khi gặp lệnh `break`
- Lệnh **`break`** dùng để thoát ra khỏi cấu trúc `switch`
- Các câu lệnh trong **mệnh đề `default`** (không bắt buộc) sẽ được thực hiện nếu giá trị của `exp` không nằm trong các giá trị được liệt kê trong các mệnh đề `case`



# Lệnh lựa chọn switch ... case

```
switch (grade)
{
    case 'A':
        System.out.println("The grade is A.");
        break;
    case 'B':
        System.out.println("The grade is B.");
        break;
    case 'C':
        System.out.println("The grade is C.");
        break;
    case 'D':
        System.out.println("The grade is D.");
        break;
    case 'F':
        System.out.println("The grade is F.");
        break;
    default:
        System.out.println("The grade is invalid.");
}
```

# Lệnh lựa chọn switch ... case

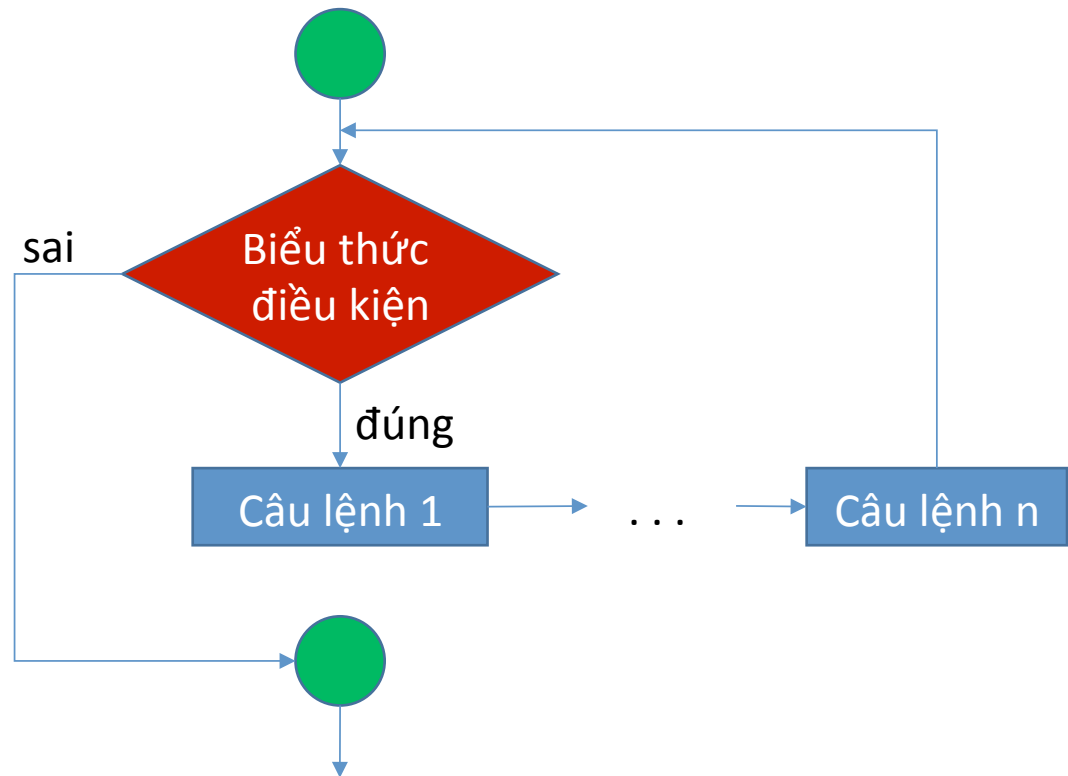
```
switch (month)
{
case 1:
case 3:
case 5:
case 7:
case 8:
case 10:
case 12:
    System.out.println("31-day month");
    break;
case 4:
case 6:
case 9:
case 11:
    System.out.println("30-day month");
    break;
default:
    System.out.println("28/29-day month");
}
```

# Cấu trúc lặp (repetition/loop)

- Được sử dụng để thực hiện **lặp đi lặp lại** 1 tác vụ (khối lệnh) 1 số lần dựa vào điều kiện cho trước

- Các lệnh lặp:

- `while`
- `for`
- `do ... while`

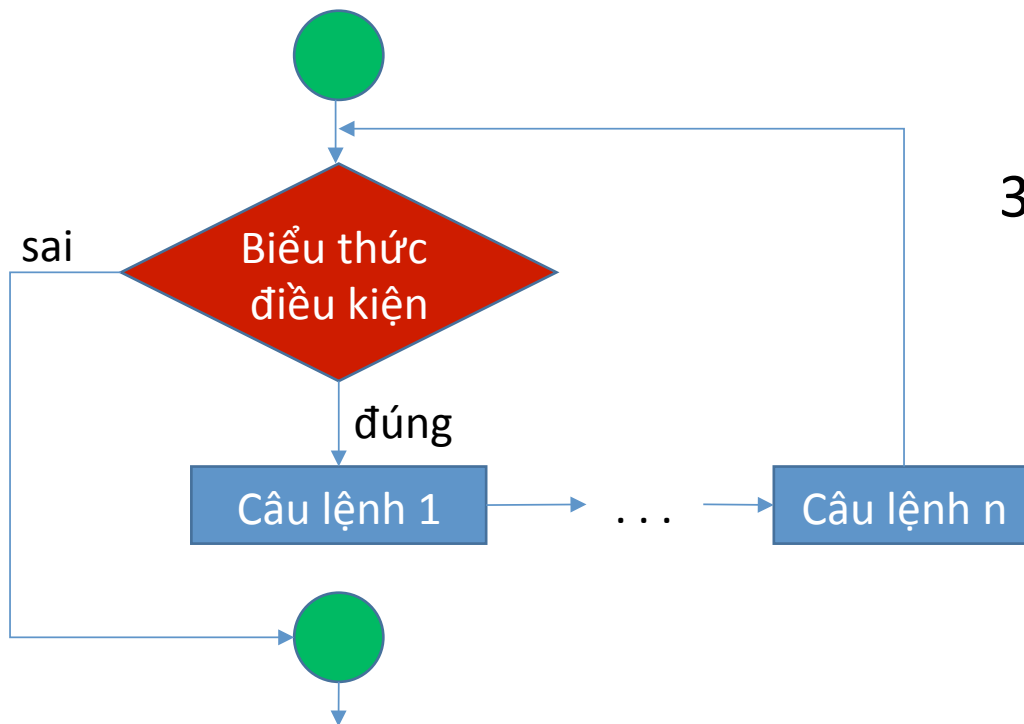


# Lệnh lặp `while`

```
while (biểu thức điều kiện) {  
    câu lệnh 1;  
    . . .  
    câu lệnh n;  
}
```

**thân vòng lặp**

1. Kiểm tra “biểu thức điều kiện”
2. Nếu đúng, thực hiện thân vòng lặp  
Ngược lại, thoát khỏi vòng lặp
3. Trở lại bước 1.



# Lệnh lặp `while`

- Lưu ý:

- Điều kiện được kiểm tra trước khi thực hiện thân vòng lặp  
⇒ Thân vòng có thể không được thực hiện lần nào cả nếu điều kiện sai từ đầu
- Vòng lặp với số lần lặp vô tận được gọi là vòng lặp vô tận (infinite loop)
- Để không bị trường hợp lặp vô tận, trong thân vòng lặp phải có ít nhất 1 lệnh thay đổi giá trị của biểu thức điều kiện
- Vòng lặp `while` thường được sử dụng cho các trường hợp chưa xác định số lần lặp

```
int x = 20;  
while (x > 0)  
    System.out.println("x is greater than 0");
```

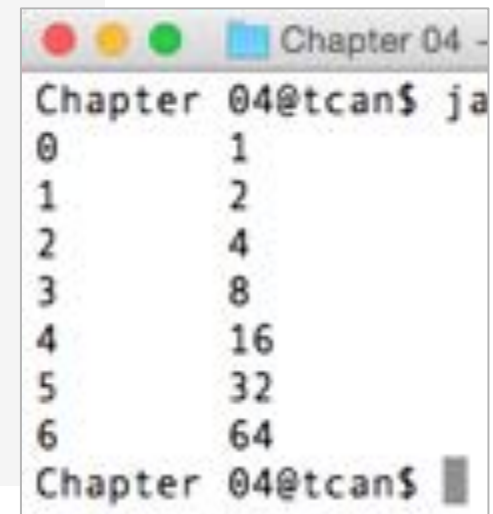
# Lệnh lặp `while`

- Ví dụ: Hiển thị các giá trị  $2^i$  với  $2^i \leq 2^N$ ,  $N$  được truyền qua đối số dòng lệnh.

```
public class PowersOfTwo {  
    public static void main(String[] args) {  
        int N = Integer.parseInt(args[0]);  
  
        int i = 0;  
        int v = 1;  
        while (i <= N) {  
            System.out.println(i + "\t" + v);  
            i = i + 1;  
            v = 2 * v;  
        }  
    }  
}
```

`args[0]`

`$java PowersOfTwo 6`



The screenshot shows a terminal window titled "Chapter 04 -". The prompt is "Chapter 04@tcan\$". The command "java PowersOfTwo 6" has been executed, resulting in the following output:

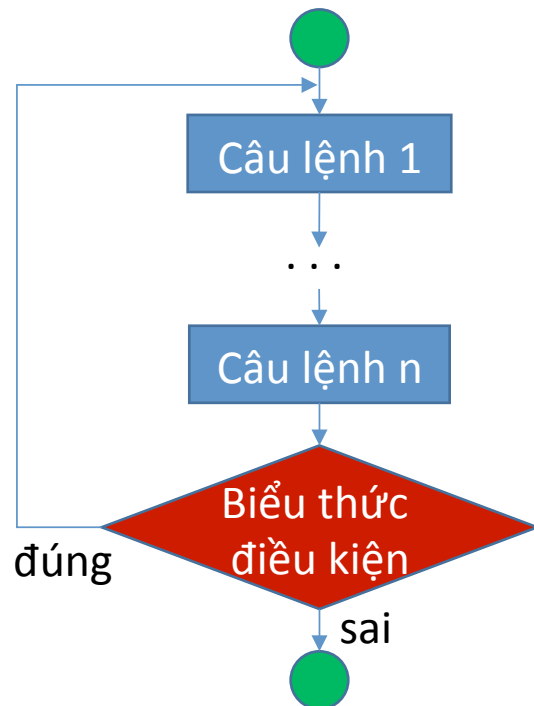
i	2 <sup>i</sup>
0	1
1	2
2	4
3	8
4	16
5	32
6	64

The terminal prompt is now "Chapter 04@tcan\$".

# Lệnh lặp do ... while

```
do {  
    câu lệnh 1;  
    . . .  
    câu lệnh n;  
} while (biểu thức điều kiện);
```

**thân vòng lặp**



1. Thực hiện thân vòng lặp
2. Định giá “biểu thức điều kiện”:
  - a) Nếu đúng, trở lại bước 1
  - b) Ngược lại, thoát khỏi vòng lặp

- **Nhận xét:**

- Thân vòng lặp luôn được thực hiện **ít nhất 1 lần**
- Chú ý trường hợp vòng lặp vô tận
- Thường được sử dụng trong trường hợp số lần lặp chưa xác định

# Lệnh lặp do ... while

```
import java.util.Scanner;    // Needed for the Scanner class
public class DoWhileSqrt {
    public static void main(String[] args) {
        int n;                // number that will be calculated the square root
        char repeat;          // To hold 'y' or 'n'

        Scanner keyboard = new Scanner(System.in);
        do {
            System.out.print("Enter a number: ");
            n = keyboard.nextInt();
            keyboard.nextLine();

            System.out.println("SQRT of " + n + " is " + Math.sqrt(n));

            System.out.print("Continue (y/n)? ");
            repeat = keyboard.nextLine().charAt(0); // Read a line.
        } while (repeat == 'Y' || repeat == 'y');
    }
}
```

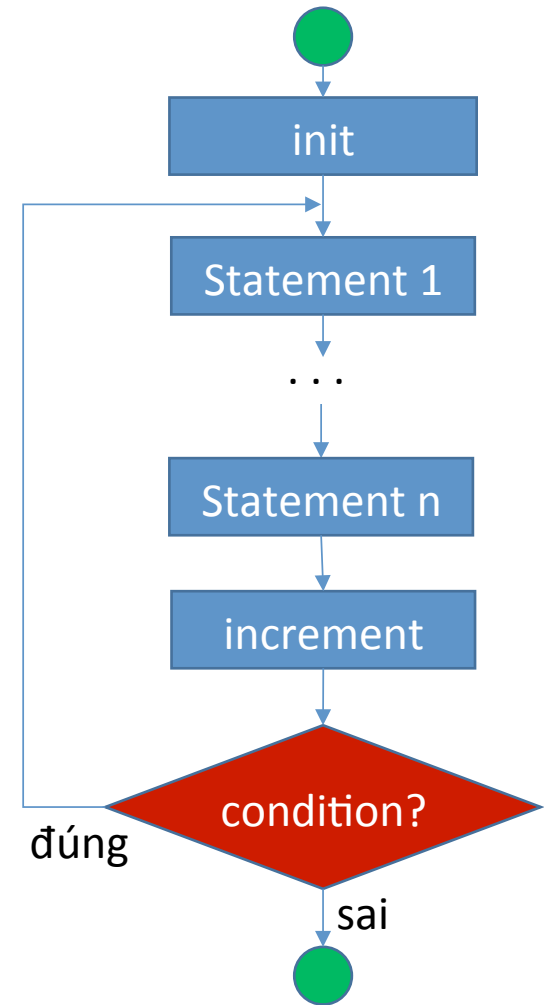


# Lệnh lặp `for`

```
for (init; condition; increment) {  
    statement 1;  
    ...  
    statement n;  
}
```

**thân vòng lặp**

1. Thực hiện biểu thức khởi tạo
2. Thực hiện thân vòng lặp
3. Thực hiện biểu thức tăng/lặp
4. Định giá “biểu thức điều kiện”:
  - a) Nếu đúng, trở lại **bước 2**
  - b) Ngược lại, thoát khỏi vòng lặp



# Lệnh lặp `for`

- Nhận xét:

- Biểu thức **khởi tạo** (init) được thực hiện 1 lần duy nhất trước khi vào vòng lặp.
  - Có thể khai báo biến trong biểu thức khởi tạo
- Chức năng của biểu thức **điều kiện** tương tự lệnh lặp `while`
- Biểu thức **lặp/tăng** (increment) được thực hiện sau mỗi lần thực hiện xong thân vòng lặp
- Cả ba biểu thức trong lệnh lặp đều **không bắt buộc**
- Lệnh lặp `for` thường được sử dụng trong trường hợp biết trước số lần lặp
- Lệnh lặp `for` chuyển về lệnh lặp `while` và ngược lại
- Biểu thức khởi tạo và lặp có thể chứa nhiều lệnh, cách nhau bằng dấu **,**

# Lệnh lặp `for`

```
import java.util.Scanner;    // Needed for the Scanner class

public class ForSum_1_N {
    public static void main(String[] args) {
        int n;
        Scanner keyboard = new Scanner(System.in);

        // Get the maximum value to display.
        System.out.print("n = ");
        n = keyboard.nextInt();

        // Calculate the sum
        int sum = 0;
        for (int i = 1; i <= n; i++)
            sum += i;

        System.out.println("1 + 2 + ... + " + n + " = " + sum);
    }
}
```

# Lệnh lặp `for`

- Chuyển đổi giữa lệnh lặp `for` và `while`

```
for (init; condition; increment) {  
    statement 1;  
    ...  
    statement n;  
}
```



```
init;  
while (condition) {  
    statement 1;  
    ...  
    statement n;  
    increment;  
}
```

```
while (condition) {  
    statement 1;  
    ...  
    statement n;  
}
```



```
for (; condition; ) {  
    statement 1;  
    ...  
    statement n;  
}
```

# Lệnh lặp `for`

khởi tạo

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + "\t" + v);
    v = 2 * v;
    i = i + 1;
}
```

Biểu thức tăng

```
for (int i=0, int v=1; i <= N; i++, v *= 2) {
    System.out.println(i + "\t" + v);
}
```

Thông thường

```
int v = 1;
for (int i=0; i <= N; i++) {
    System.out.println(i + "\t" + v);
    v = 2 * v;
}
```

# Các cấu trúc điều khiển lồng nhau

- Các cấu trúc điều khiển có thể được lồng nhau:

```
if (a == 0) {  
    if (b == 0)  
        System.out.println("PT vo nghiem");  
    else  
        System.out.println("PT vo so nghiem");  
}  
else {  
    ...  
}
```

*Giải PT bậc 1*

*Hiển thị tam giác các dấu \**

```
for (int i=0; i<5; i++) {  
    for (int j=0; j <= i; j++)  
        System.out.print("*");  
  
    System.out.println("");  
}
```

*Tính tổng các số chẵn từ 1..N*

```
int sum = 0;  
for (int i=0; i < N; i++) {  
    if (i % 2 == 0)  
        sum += i;  
}
```

```
*  
* *  
* * *  
* * * *  
* * * * *
```

# Lệnh `break` và `continue`

- Lệnh `break`: Kết thúc một vòng lặp
  - Khi gặp lệnh `break`, vòng lặp sẽ kết thúc ngay lập tức, bất kể giá trị của biểu thức điều kiện
  - Lệnh này còn được sử dụng trong câu lệnh `switch...case`
- Lệnh `continue`: Bắt đầu chu kỳ lặp mới
  - Khi gặp lệnh `continue`, chương trình sẽ trở về đầu vòng lặp để thực hiện chu kỳ lặp mới
  - Các lệnh dưới lệnh `continue` sẽ bị bỏ qua
- **Chú ý:** Chỉ sử dụng 2 lệnh này khi thật cần thiết vì chúng phá vỡ tính cấu trúc của chương trình.

# Lựa chọn lệnh lặp

---

- Lệnh lặp `while`:
  - Kiểm tra điều kiện trước (pre-test loop)
  - Sử dụng trong trường hợp ta không muốn thực hiện thân vòng lặp nếu điều kiện sai từ đầu
- Lệnh lặp `do...while`:
  - Kiểm tra điều kiện sau (post-test loop)
  - Sử dụng trong trường hợp thực hiện lặp ít nhất 1 lần
- Lệnh lặp `for`:
  - Kiểm tra điều kiện trước (pre-test loop)
  - Thường được sử dụng trong trường hợp lặp với biến đếm (counting variable)

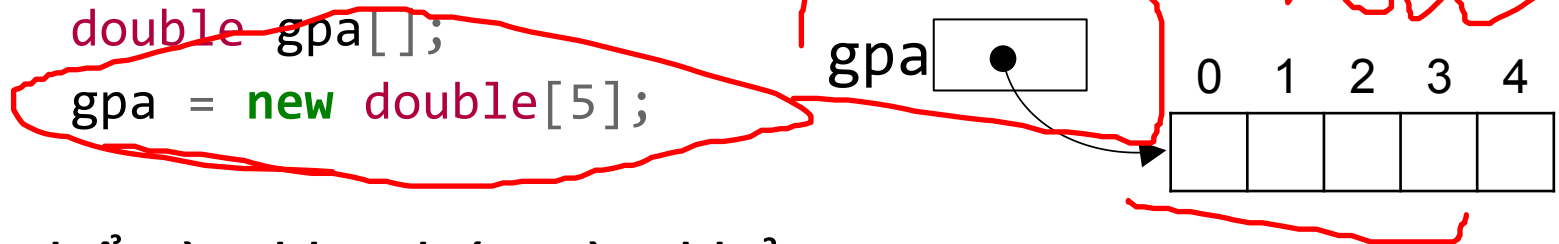


# Mảng (array)

# Mảng (array)

- Là một dãy các phần tử (giá trị) có cùng kiểu dữ liệu
- Là một biến **kiểu tham chiếu**:

- Khai báo: <kiểu dữ liệu> <tên mảng>[];
- Tạo mảng: <tên mảng> = **new** <kiểu dữ liệu>;



- Có thể vừa khai báo vừa khởi tạo

```
int []number = {2, 4, 6, 8};  
String []monthName = {"Jan", "Feb", "March",...};
```

# Mảng (array)

- Các phần tử của mảng được truy xuất thông qua chỉ số
- Chỉ số của mảng bắt đầu từ 0

```
public class Fibonacci {  
    public static void main(String args[]) {  
        int fibo[] = new int [10];  
  
        fibo[0] = fibo[1] = 1;  
  
        for (int i=2; i< 10; i++)  
            fibo[i] = fibo[i-1] + fibo[i-2];  
  
        for (int i=0; i< 10; i++)  
            System.out.print(fibo[i] + " ");  
    }  
}
```

# Mảng (array)

## • Mảng các đối tượng:

```

Person []prs,      ①
prs = new Person[5]; ②
prs[0] = new Person(); ③
//...

```

```

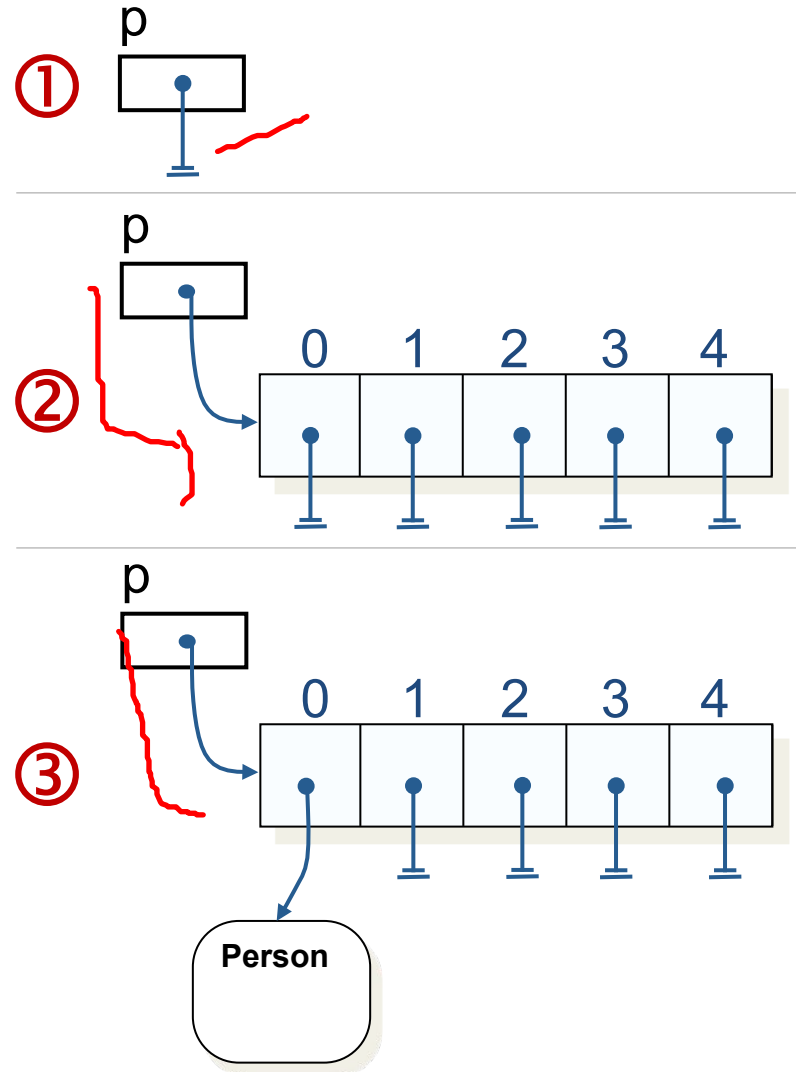
prs[0].setName("Tommy");
prs[0].setAge(20);

```

```

for (Person t : prs) {
    System.out.println(t.getName());
}

```



# Mảng (array)

- Java hỗ trợ mảng đa chiều (hai hoặc lớn hơn hai chiều).
- Ví dụ khai báo mảng hai chiều chứa một bảng dữ liệu kiểu **int** gồm 3 dòng và 4 cột như sau:

```
class TwoD {
    public static void main (String args[]) {
        int h, c;
        int table[][] = new int [3][4];
        for (h=0; h < 3; ++h) {
            for (c=0; c < 4; ++c) {
                table[h][c] = (h*4) + c + 1;
                System.out.print(table[h][c] + " ");
            } System.out.println();
        }
    }
}
```

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

table[1][2]

- 
- Viết chương trình tạo ra một mảng 2 chiều có 4 dòng 5

---

# Chuỗi ký tự

# Chuỗi ký tự

- Khai báo và khởi tạo giá trị cho chuỗi ký tự:

- `String str1 = new String( );`
- `String str11 = "Java strings are powerful";`

*//str1 chứa một chuỗi rỗng.*

- `String str2 = new String("Hello World");`
- `String str21 = new String(str2);`

*//str2 và str21 cùng chứa "Hello World"*

- `char ch[] = {'A','B','C','D','E'};`
- `String str3 = new String(ch);`

*//str3 chứa "ABCDE"*

- `String str4 = new String(ch,0,2);`

*//str4 chứa "AB" vì 0- tính từ ký tự bắt đầu, 2- là số lượng ký tự kể từ ký tự bắt đầu.*



# Chuỗi ký tự

- Lớp String có một vài phương thức thường dùng:
- ***boolean equals(String str):** trả về giá trị đúng nếu như chuỗi gọi hàm có thứ tự các ký tự trùng với chuỗi str truyền vào:*

```
String name = new String ("Java Language");  
String name1 = new String ("Java Languages");  
if ( name.equals(name1) )  
    System.out.println("equal");  
else    System.out.println ("not equal");
```

# Chuỗi ký tự

- ***int compareTo(<sup>1</sup>String <sup>1</sup>str)***: trả về giá trị **nhỏ hơn 0** nếu như chuỗi gọi hàm xếp trước chuỗi **str** theo thứ tự bảng chữ cái, **lớn hơn 0** nếu chuỗi gọi hàm đứng sau chuỗi **str** và **bằng 0** nếu hai chuỗi bằng nhau. Ví dụ:

```
if ( name.compareTo(name1) > 0)
    System.out.println("chuỗi 1 đứng trước 2");
else if ( name.compareTo(name1) < 0)
    System.out.println(" chuỗi 1 đứng sau 2 ");
else
    System.out.println(" chuỗi 1 bằng chuỗi 2 ");
```

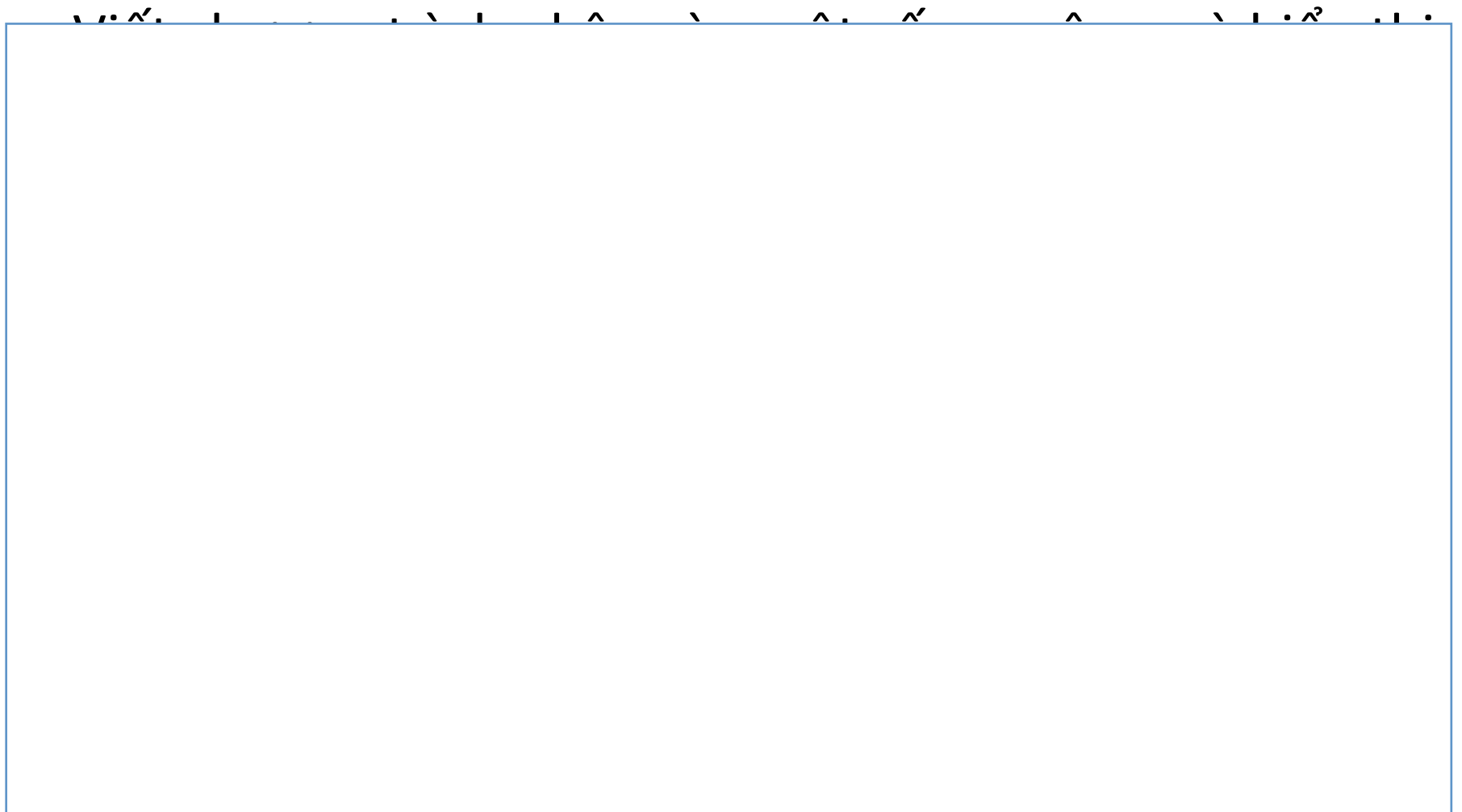
# Chuỗi ký tự

- `char charAt(int index)`
- `int length()`
- `String substring(int beginIndex)`
- `String substring(int beginIndex, int endIndex)`
- `boolean contains(CharSequence s)`
- `boolean equals(Object another)`
- `boolean isEmpty()`
- λ • `String concat(String str)`
- ✂ • `String replace(String oldStr, String newStr)`
- `String trim()`
- `int indexOf(String substring)`
- `int indexOf(String substring, int fromIndex)`
- `String toLowerCase()`
- `String toUpperCase()`

# Lớp Math

---

- Cung cấp các hàm toán học
- Hầu hết các phương thức là tĩnh (static, gọi từ lớp)
- Một số hàm thông dụng như:
  - `double/float/int/long abs(double/float/int/long);`
  - `double log/log10(double);`
  - `long/int round(double/float);`
  - `double sqrt(double);`
  - `double random();`
- Một số hàm khác: `sin`, `cos`, `asin`, `acos`, `exp`, `floor`, `ceil`, `pow`, `min`, `max`



# Xử lý ngoại lệ

# Xử lý ngoại lệ (exception handling)

---

- Ngoại lệ: là một lỗi xảy ra khi thực thi chương trình:
  - Ví dụ: chia cho 0, mở tập tin không tồn tại,...
- Khi một lỗi xảy ra, một **đối tượng** ngoại lệ sẽ sinh ra và chương trình phải “bắt” và xử lý ngoại lệ
- Bắt ngoại lệ: dùng câu lệnh `try...catch...finally`
- Xử lý ngoại lệ:
  - Truyền tiếp ngoại lệ: “quảng” ngoại lệ cho nơi khác trong chương trình xử lý
  - Khắc phục lỗi để chương trình có thể tiếp tục

# Xử lý ngoại lệ (exception handling)

- Bắt ngoại lệ:

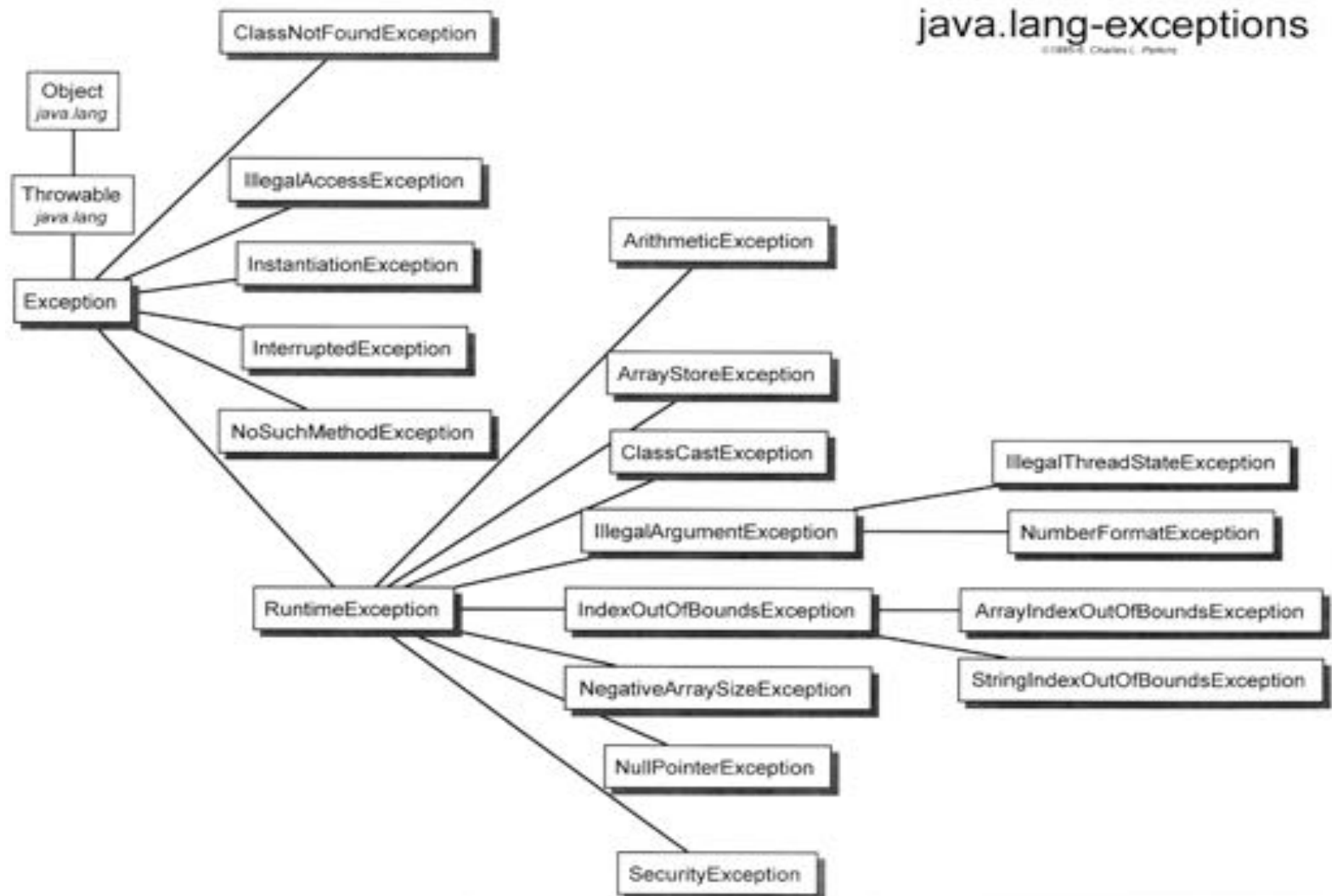
```
try {  
    //các câu lệnh có khả năng  
    //sinh ra lỗi  
}  
catch (<loại ngoại lệ 1>) {  
    //xử lý ngoại lệ 1  
}  
...  
catch (<loại ngoại lệ n>) {  
    //xử lý ngoại lệ n  
}  
finally {  
    //các lệnh sẽ được thực hiện  
    //cả trong trường hợp có và  
    //không có ngoại lệ xảy ra  
}
```

- Lưu ý:

- Khối lệnh **finally** là không bắt buộc
- Các loại ngoại lệ trong Java được tổ chức theo dạng cây phân cấp với gốc cây phân cấp là **Exception**



# Xử lý ngoại lệ (exception handling)



# Xử lý ngoại lệ (exception handling)

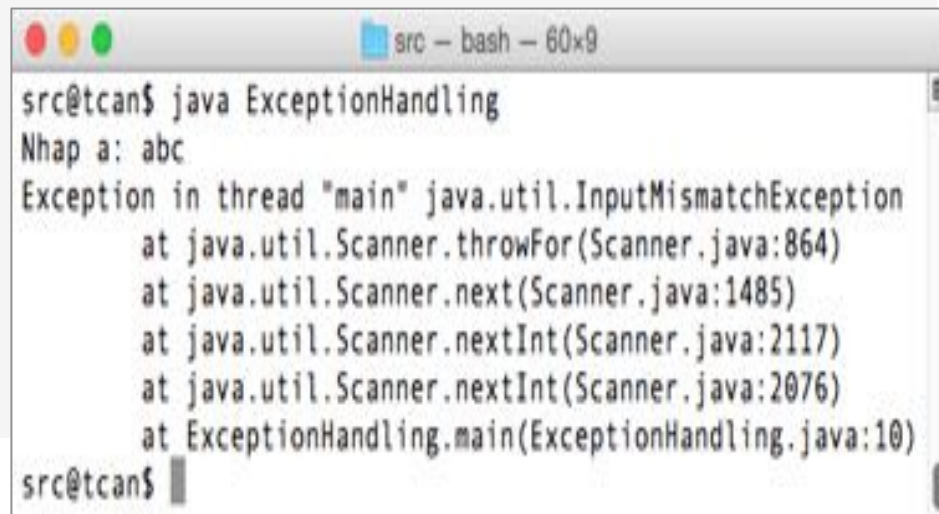
```
import java.util.Scanner;
import java.util.InputMismatchException;

class ExceptionHandling {
    public static void main(String args[]) {
        int a, b;
        Scanner s = new Scanner(System.in);

        System.out.print("Nhập a: ");
        a = s.nextInt();
        s.nextLine();

        System.out.print("Nhập b: ");
        b = s.nextInt();
        s.nextLine();

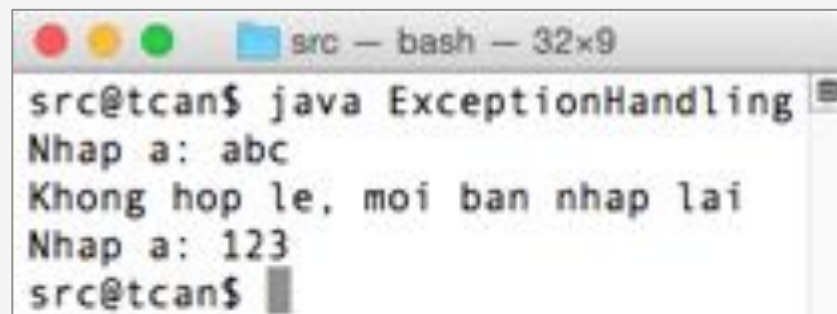
        //...
    }
}
```



```
src - bash - 60x9
src@tcan$ java ExceptionHandling
Nhập a: abc
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
    at java.util.Scanner.nextInt(Scanner.java:2117)
    at java.util.Scanner.nextInt(Scanner.java:2076)
    at ExceptionHandling.main(ExceptionHandling.java:10)
src@tcan$
```

# Xử lý ngoại lệ (exception handling)

```
import java.util.Scanner;
import java.util.InputMismatchException;
class ExceptionHandling {
    public static void main(String args[]) {
        int a, b;
        Scanner s = new Scanner(System.in);
        while (true) {
            try {
                System.out.print("Nhập a: ");
                a = s.nextInt();
            }
            catch (InputMismatchException e) {
                System.out.println("Không hợp lệ, mời bạn nhập lại");
                continue;
            }
            finally {
                s.nextLine();
            }
            break;
        } //while
        //tiếp tục nhập b, ...
    } //main
}
```



```
src — bash — 32x9
src@tcan$ java ExceptionHandling
Nhập a: abc
Không hợp lệ, mời bạn nhập lại
Nhập a: 123
src@tcan$
```



Question?

CT176 – LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG