

HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.



Bài 2: Cú pháp Java cơ bản

ONE LOVE. ONE FUTURE.

Mục tiêu bài học

- Nắm được quy định cơ bản về định danh, câu lệnh, chú thích, và biến trong Java
- Sử dụng thành thạo các kiểu dữ liệu nguyên thủy trong Java
- Nắm được các loại toán tử, các cấu trúc điều khiển, và cấu trúc dữ liệu kiểu mảng trong Java
- Hiểu được ý nghĩa ngôn ngữ mô hình hóa thống nhất UML, biết các loại biểu đồ thông dụng



Bài giảng e-learning

Bước 1 - Truy cập trang

[https://www.udacity.com/
course/java-
programming-basics--
ud282](https://www.udacity.com/course/java-programming-basics--ud282)

Bước 2: Đăng ký tài
khoản (free)

Bước 3: Nhấn nút START
FREE COURSE

The screenshot shows the Udacity website with a red box highlighting the URL in the address bar: udacity.com/course/java-programming-basics--ud282. The page title is "Programming and Development". The course title is "Java Programming Basics". A sidebar on the right lists features: "Industry-relevant content" (marked with a checkmark), "Project reviews" (marked with an X), "Mentorship" (marked with an X), "Certification" (marked with an X), and "Career Services / Job Assistance" (marked with an X). At the bottom right, a red box highlights the "START FREE COURSE" button.

udacity.com/course/java-programming-basics--ud282

UDACITY

SCHOOL OF

Programming and Development

Take your first steps towards becoming a Java developer! Learn Java syntax and create conditional statements, loops, and functions.

SAMPLE NANODEGREE PROGRAMS

- Front End Web Developer
- Full Stack Web Developer
- Java Developer

EXPLORE SCHOOL

THIS COURSE

FREE COURSE

Java Programming Basics

- ✓ Industry-relevant content
- ✗ Project reviews
- ✗ Mentorship
- ✗ Certification
- ✗ Career Services / Job Assistance

START FREE COURSE



Bài giảng e-learning

- Trong khóa học Java Programming Basics, SV học theo các bài 1, 2, và 4.
 - Lesson 1: Variables and Data Types
 - Lesson 2: Control Flow and Conditionals
 - Lesson 3: Functions (sẽ trình bày ở các bài giảng sau)
 - Lesson 4: Loops
 - Lesson 5: IntelliJ and Debugging (tham khảo)



Nội dung

1. Cơ bản về Java
2. Giới thiệu về UML



1. Cơ bản về Java

- 1.1. Các khái niệm cơ bản
- 1.2. Biến
- 1.3. Các kiểu dữ liệu cơ bản
- 1.4. Toán tử
- 1.5. Cấu trúc điều khiển
- 1.6. Mảng



1. Cơ bản về Java

1.1. Các khái niệm cơ bản

1.2. Biến

1.3. Các kiểu dữ liệu cơ bản

1.4. Toán tử

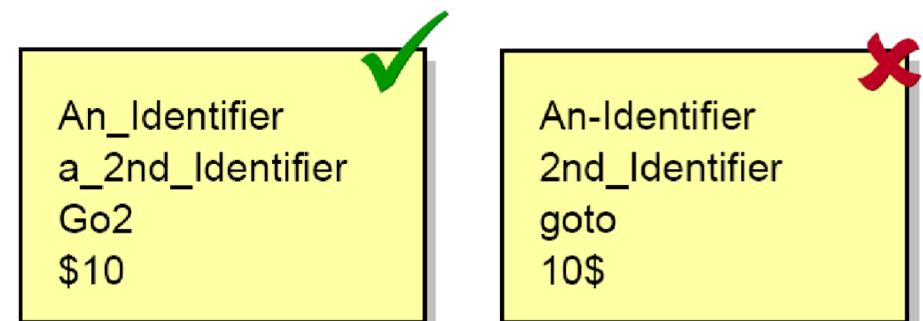
1.5. Cấu trúc điều khiển

1.6. Mảng



Định danh

- Định danh:
 - Xâu ký tự thể hiện tên các biến, các phương thức, các lớp và nhãn
 - là duy nhất trong chương trình
- Quy định với định danh hợp lệ (bắt buộc tuân thủ)
 - Gồm các ký tự có thể là chữ cái, chữ số, ký tự '\$' hoặc '_'
 - Không được phép:
 - Bắt đầu bởi một chữ số
 - Trùng với từ khóa
 - Chứa dấu cách
 - Phân biệt chữ hoa chữ thường
 - Yourname, yourname, YourName và yourName là 4 định danh khác nhau



Định danh (2)

- Quy ước với định danh - naming convention (Quy ước: không bắt buộc, nhưng nên làm theo)
 - Phải mang tính gợi nhớ
 - Ví dụ: nên dùng định danh “bookPrice” hơn là “bp” để lưu thông tin về giá 1 quyển sách
 - Bắt đầu bằng chữ cái
 - Gói (package): tất cả sử dụng chữ thường
 - theexample
 - Lớp (Class): viết hoa chữ cái đầu tiên trong các từ ghép lại
 - TheExample
 - Phương thức/thuộc tính (method/field): Bắt đầu bằng chữ thường, viết hoa chữ cái đầu tiên trong các từ còn lại
 - theExample
 - Hằng (constants): Tất cả viết hoa
 - THE_EXAMPLE



Các từ khóa

- Người lập trình không được phép sử dụng các từ khóa như một định danh
- Literals:
 - null true false
- Từ khóa (keyword):
 - abstract assert boolean break byte case catch char class continue default do double else extends final finally float for if implements import instanceof int interface long native new package private protected public return short static strictfp super switch synchronized this throw throws transient try void volatile while
- Từ dành riêng (reserved word):
 - byvalue cast const future generic goto inner operator outer rest var volatile



Câu lệnh

- Các câu lệnh kết thúc bởi dấu ;
- Nhiều lệnh có thể viết trên một dòng
- Một câu lệnh có thể viết trên nhiều dòng
 - Ví dụ:

```
System.out.println(  
    "This is part of the same line");
```

```
a=0; b=1; c=2;
```



Chú thích trong Java

- Java hỗ trợ ba kiểu chú thích như sau:
 - // Chú thích trên một dòng
 - // Không xuống dòng
 - /* Chú thích một đoạn */
 - /** Javadoc * chú thích dạng Javadoc */
- Chú thích dùng để mô tả thêm về mã nguồn (source code). Trình thông dịch sẽ bỏ qua các chú thích này.

```
/* This is a simple Java program.  
   FileName : "HelloWorld.java". */  
class HelloWorld  
{  
    // Your program begins with a call to main().  
    // Prints "Hello, World" to the terminal window.  
    public static void main(String args[])  
    {  
        System.out.println("Hello, World");  
    }  
}
```



1. Cơ bản về Java

1.1. Các khái niệm cơ bản

1.2. Biến (*Tham khảo Lesson 1 – Session 6*)

1.3. Các kiểu dữ liệu cơ bản

1.4. Toán tử

1.5. Cấu trúc điều khiển

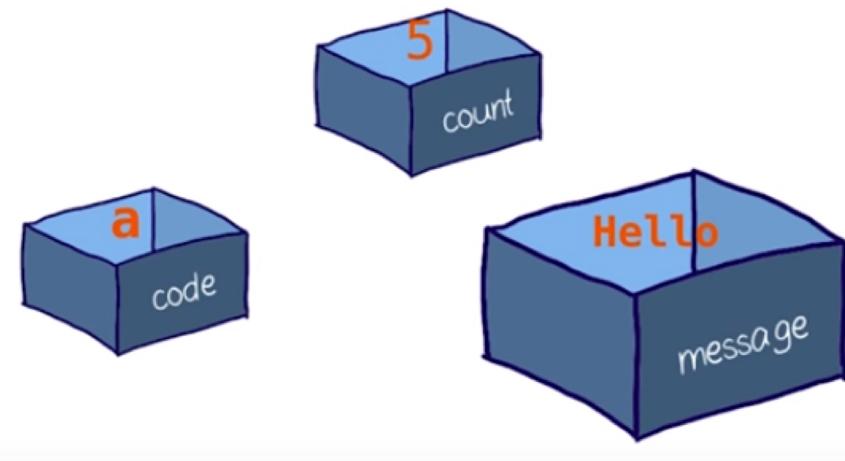
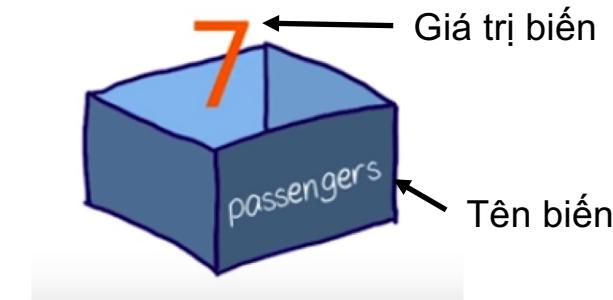
1.6. Mảng



Khái niệm biến

VARIABLES

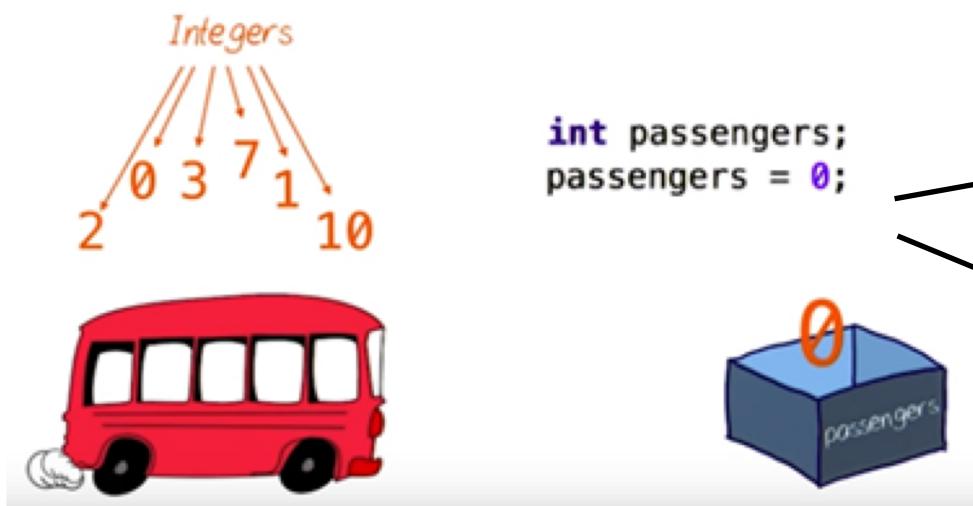
- Biến giống như 1 chiếc hộp trong bộ nhớ, chứa giá trị cho 1 đại lượng nào đó
 - Biến có tên không thay đổi
 - Biến được gán 1 giá trị, có thể thay đổi trong khi chạy
- Biến có thể chứa các giá trị kiểu số, ký tự, văn bản, hay đối tượng
 - và kiểu giá trị này của biến cũng không thay đổi, gọi là kiểu dữ liệu của biến



<https://www.youtube.com/watch?v=TGw5szyZk88>

Khai báo biến

- Biến khi dùng phải được khai báo tên (định danh) và gán cho một kiểu dữ liệu (số, ký tự, văn bản, đối tượng, v.v.)
- Các biến đơn cần phải được khởi tạo trước khi sử dụng



Lệnh khai báo 1 biến có tên `passengers`, có kiểu số nguyên, trong Java ký hiệu là `int`.

Lệnh khởi tạo giá trị biến `passengers = 0`.

Khai báo biến (2)

- Có thể kết hợp khai báo và khởi tạo cùng một lúc.
- Sử dụng toán tử = để gán (bao gồm cả khởi tạo)
- Ví dụ:

The diagram illustrates the process of declaring and initializing variables. It features three horizontal orange arrows pointing to the right. The first arrow is labeled "Declaring" above it. The second arrow is labeled "Initializing" above it. Below the arrows, there are three lines of C-like pseudocode:
int price = **0**;
int speed = **100**;
int stockPrice = **75**;



Sử dụng biến

```
int passengers;  
passengers = 0;
```



```
passengers = passengers + 5;
```



```
passengers = passengers - 3;
```



```
System.out.println(passengers); →
```

Lệnh in ra giá trị hiện tại của biến
passengers (không có "" quanh tên biến)
Nếu passengers chưa khởi tạo, sẽ báo lỗi

Phạm vi sử dụng của biến

- Phạm vi của biến là vùng chương trình mà trong đó biến có thể được tham chiếu đến, có thể sử dụng được.
- Phạm vi hoạt động (scope) của các biến cho phép xác định các nguyên lý của tạo biến, sử dụng biến và giải phóng biến
- Phân loại:
 - Biến toàn cục: phạm vi trong cả chương trình
 - Biến cục bộ: được khai báo trong một phương thức/khoi lệnh thì chỉ có thể truy cập trong phương thức/khoi lệnh đó.



Phạm vi sử dụng của biến (2)

```
boolean isLightGreen = ? ; //true or false  
  
if(isLightGreen) {  
    //traffic light is green  
    double carSpeed = 100; //in km/hr  
    System.out.println("Drive!");  
    System.out.println("Speed is: " + carSpeed);  
}
```

Block of code where a variable can be used

A set of curly braces defines a variable scope

{ scope }

isLightGreen scope carSpeed scope

Tham khảo Lesson 2 - Session 7



1. Cơ bản về Java

1.1. Các khái niệm cơ bản

1.2. Biến

Tham khảo Lesson 1 - Session 16, 12, 13

1.3. Các kiểu dữ liệu cơ bản

1.4. Toán tử

1.5. Cấu trúc điều khiển

1.6. Mảng



Các kiểu dữ liệu trong Java

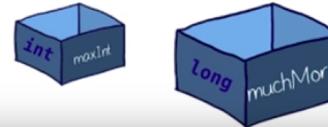
- Trong Java kiểu dữ liệu được chia thành hai loại:
 - Kiểu dữ liệu nguyên thủy (primitive)
 - Số nguyên (integer)
 - Số thực (float)
 - Ký tự (char)
 - Giá trị logic (boolean)
 - Kiểu dữ liệu tham chiếu (reference)
 - Mảng (array)
 - Đối tượng (object)
- Kích thước của các kiểu dữ liệu nguyên thủy được định nghĩa bởi JVM. Chúng giống nhau trên tất cả các platform
- Cần cân bằng giữa nhu cầu lưu trữ (độ lớn có thể của giá trị) và việc tiết kiệm bộ nhớ (không dư thừa ô nhớ)



Số nguyên

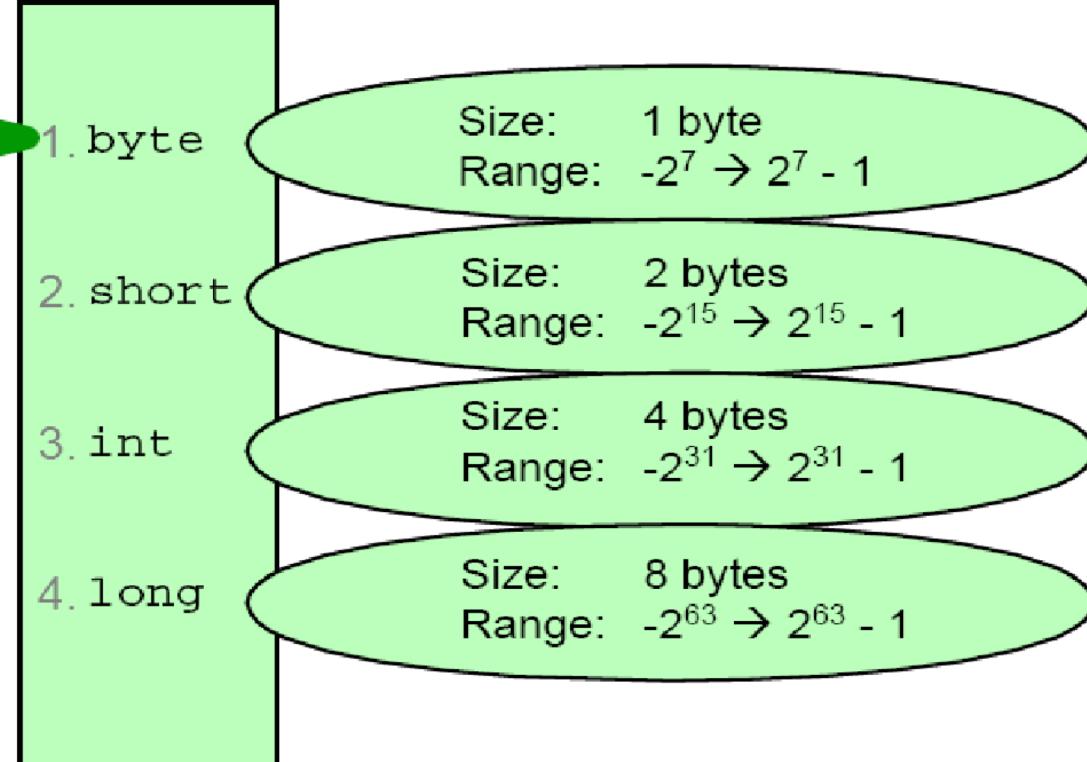
- Số nguyên có dấu
- Khởi tạo với giá trị 0

```
(1) Integer int maxInt = 2147483647;  
(2) Long long muchMore = 2147483647*10000000;
```

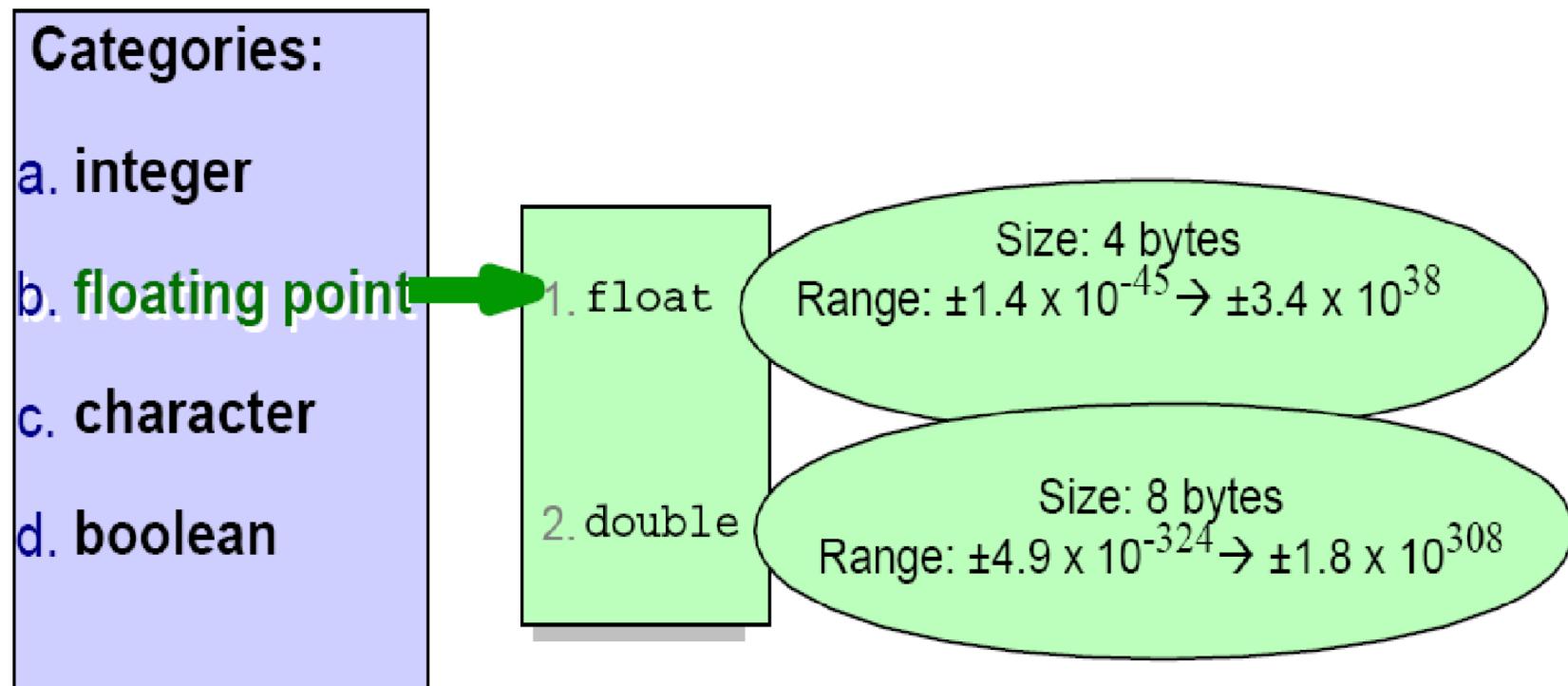


Categories:

- a. **integer**
- b. **floating point**
- c. **character**
- d. **boolean**

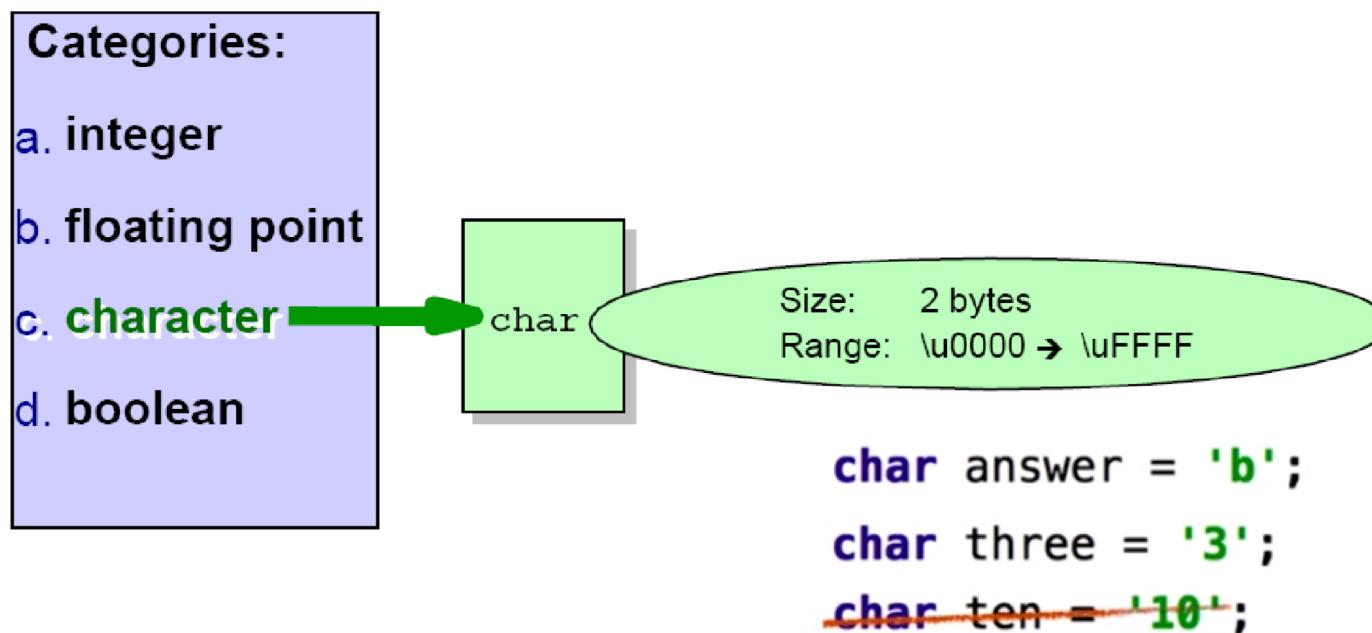


- Khởi tạo với giá trị 0.0



```
double fraction = 99.275;
```

- Ký tự Unicode không dấu, được đặt giữa hai dấu nháy đơn
- 2 cách gán giá trị:
 - Sử dụng các chữ số trong hệ 16: char uni = '\u05D0';
 - Sử dụng ký tự: char a = 'A';
- Giá trị mặc định là giá trị zero (\u0000)



- Giá trị boolean được xác định rõ ràng trong Java
 - Một giá trị int không thể sử dụng thay cho giá trị boolean
 - Có thể lưu trữ giá trị hoặc true hoặc false
- Biến boolean được khởi tạo là false

Categories:

- a. integer
- b. floating point
- c. character
- d. boolean

`boolean fact = true;`
`boolean condition = false;`

Size: 1 byte
Range: true | false

Giá trị hằng (literal)

- Literal là một giá trị của các kiểu dữ liệu nguyên thủy và xâu ký tự.
- Gồm 5 loại:
 - integer
 - floating point
 - boolean
 - character
 - String

Literals

integer.....	7
floating point...	7.0f
boolean.....	true
character.....	'A'
string.....	"A"



Hằng số nguyên

- Hệ cơ số 8 (Octals) bắt đầu với chữ số 0
 - $032 = 011\ 010(2) = 16 + 8 + 2 = 26(10)$
- Hệ cơ số 16 (Hexadecimals) bắt đầu với 0 và ký tự x
 - $0x1A = 0001\ 1010(2) = 16 + 8 + 2 = 26(10)$
- Kết thúc bởi ký tự “L” thể hiện kiểu dữ liệu long
 - 26L
- Ký tự hoa, thường cho giá trị bằng nhau
 - 0x1a , 0x1A , 0X1a , 0X1A đều có giá trị 26 trong hệ decimal



Hằng số thực

- float kết thúc bằng ký tự f (hoặc F)
 - 7.1f
- double kết thúc bằng ký tự d (hoặc D)
 - 7.1D
- e (hoặc E) được sử dụng trong dạng biểu diễn khoa học:
 - 7.1e2
- Một giá trị thực mà không có ký tự kết thúc đi kèm sẽ có kiểu là double
 - 7.1 giống như 7.1d



Hằng boolean, ký tự và xâu ký tự

- boolean:
 - true
 - false
- Ký tự:
 - Được đặt giữa 2 dấu nháy đơn
 - Ví dụ: 'a', 'A' hoặc '\uffff'
- Xâu ký tự:
 - Được đặt giữa hai dấu nháy kép
 - Ví dụ: “Hello world”, “Xin chao ban”,...



Escape sequence

- Các ký tự điều khiển nhấn phím
 - \b backspace
 - \f form feed
 - \n newline
 - \r return (về đầu dòng)
 - \t tab
- Hiển thị các ký tự đặc biệt trong xâu
 - \" quotation mark
 - \' apostrophe
 - \\ backslash



Chuyển đổi kiểu dữ liệu (Casting)

- Java là ngôn ngữ định kiểu chặt
 - Gán sai kiểu giá trị cho một biến có thể dẫn đến các lỗi biên dịch hoặc các ngoại lệ của JVM
- JVM có thể ngầm định chuyển từ một kiểu dữ liệu hẹp sang một kiểu rộng hơn
- Để chuyển sang một kiểu dữ liệu hẹp hơn, cần phải định kiểu rõ ràng.

```
int a, b;  
short c;  
a = b + c;
```

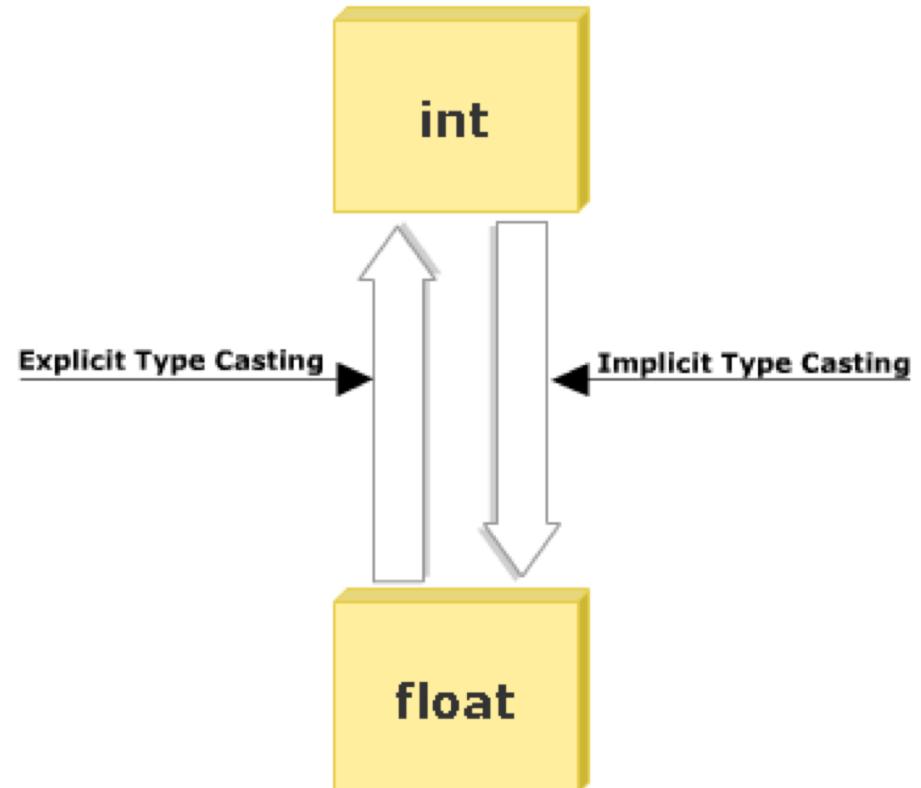
```
int d;  
short e;  
e = (short)d;
```

```
double f;  
long g;  
f = g;  
g = f; //error
```



Chuyển đổi kiểu dữ liệu (2)

- Chuyển đổi kiểu sẽ được thực hiện tự động nếu không xảy ra mất mát thông tin
 - byte → short → int → long → float
- Lưu ý: ép kiểu từ short về char, thì phải ép kiểu tường minh
- Ép kiểu trực tiếp (explicit cast) để giảm độ chính xác



Ví dụ - chuyển đổi kiểu

```
long p = (long) 12345.56; // p sẽ nhận giá trị 12345
int g = p; // không hợp lệ dù kiểu int có thể lưu giá
            trị 12345
char c = 't';
int j = c; // hợp lệ, tự động chuyển đổi
short k = c; // không hợp lệ, phải ép kiểu tường minh
short k = (short) c; // hợp lệ
float f = 12.35; // Báo lỗi do 12.35 là hằng double
float f = 0.0; // Báo lỗi do 0.0 là hằng double
float f = 0;
long l = 999999999999; //Báo lỗi: The literal
                        999999999999 of type int is out of range
short k = 99999999; // Báo lỗi: Type mismatch: cannot
                      convert from int to short
```



Ví dụ SV tự thử và kiểm tra kết quả

short i = 6, j=7;

i = i + j;

i += j;

short i, j = 5;

int n = 6;

i = (short)n + j;



1. Cơ bản về Java

- 1.1. Các khái niệm cơ bản
- 1.2. Biến
- 1.3. Các kiểu dữ liệu cơ bản
- 1.4. Toán tử** *Tham khảo Lesson 1 - Session 18*
- 1.5. Cấu trúc điều khiển
- 1.6. Mảng



Toán tử (Operators)

- Kết hợp các giá trị đơn hoặc các biểu thức con thành những biểu thức mới, phức tạp hơn và có thể trả về giá trị.
- Java cung cấp nhiều dạng toán tử sau:
 - Toán tử số học
 - Toán tử bit, toán tử quan hệ
 - Toán tử logic
 - Toán tử gán
 - Toán tử một ngôi

✓ A = B + C
✓ Z = Y * Y
✓ Result = (A+B) > (C+D)



- Toán tử số học
 - +, -, *, /, %
- Toán tử bit
 - AND: &, OR: |, XOR: ^, NOT: ~
 - Dịch bit: <<, >>
- Toán tử quan hệ
 - ==, !=, >, <, >=, <=
- Toán tử logic
 - &&, ||, !

- Toán tử một ngôi
 - Đảo dấu: +, -
 - Tăng giảm 1 đơn vị: ++, --
 - Phủ định một biểu thức logic: !
- Toán tử gán
 - =, +=, -=, %= tương tự với >>, <<, &, |, ^

Toán tử “/”

```
int i = 10/3;  
float f0 = 10;  
float f1 = (float) 10/3;  
float f2 = 10/3;  
float f3 = f0/3;  
  
System.out.println(i);      //3  
System.out.println(f1);    //3.3333333  
System.out.println(f2);    //3.0  
System.out.println(f3);    //3.3333333
```



Thứ tự ưu tiên của toán tử

- Cho biết toán tử nào thực hiện trước
- Được xác định bởi các dấu ngoặc đơn hoặc theo ngầm định như sau (ưu tiên từ trên xuống thực hiện trước):

1. Toán tử [] . ()
2. Toán tử x++ x--
3. Toán tử một ngôi: ++x --x +x -
x ~ !
4. Toán tử khởi tạo, toán tử chuyển kiểu: new (type) x
5. Nhân, chia: * / %
6. Cộng, trừ: + -
7. Dịch bit: << >> >>> (unsigned shift)
8. So sánh: < > <= >= instanceof
9. So sánh bằng == !=

10. Toán tử bit AND: &
11. Toán tử bit OR: ^
12. Toán tử bit XOR: |
13. Toán tử logic AND: &&
14. Toán tử logic OR: ||
15. Toán tử điều kiện: (ternary)
?:
16. Toán tử gán: = *= /= %= += -=
>>= <<= >>>= &= ^= |=



Thứ tự ưu tiên của toán tử - Ví dụ 1

```
double paid = 10;  
double change = 3.25;  
double tip = (paid-change)*0.2;
```

$$1.35 \quad \begin{array}{c} \text{tip} \\ \text{---} \\ \text{---} \end{array} = (\begin{array}{c} 10 \\ \text{paid} \\ \text{---} \end{array} - \begin{array}{c} 3.25 \\ \text{change} \\ \text{---} \end{array}) * 0.2$$
$$\begin{array}{c} \text{tip} \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} 10 \\ \text{paid} \\ \text{---} \end{array} - \begin{array}{c} 3.25 \\ \text{change} \\ \text{---} \end{array} * 0.2$$

9.35



Thứ tự ưu tiên của toán tử - Ví dụ 2

```
int i;  
System.out.println(i=5);           //5  
System.out.println(i+=4);         //9  
System.out.println(i++);          //9  
System.out.println(--i);          //9
```



1. Cơ bản về Java

- 1.1. Các khái niệm cơ bản
- 1.2. Biến
- 1.3. Các kiểu dữ liệu cơ bản
- 1.4. Toán tử
- 1.5. Cấu trúc điều khiển**
- 1.6. Mảng

Tham khảo Lesson 2 – Session 1..16



1.5. Cấu trúc điều khiển

- Là các cấu trúc lệnh nhằm chỉ định cho chương trình thực hiện các câu lệnh/đoạn lệnh khác nhau, tùy theo từng điều kiện nào đó.
- 2 loại cấu trúc điều khiển:
 - Câu lệnh điều kiện
 - Lệnh if – else,
 - Lệnh switch – case
 - Câu lệnh lặp
 - Vòng lặp for
 - Vòng lặp while
 - Vòng lặp do – while



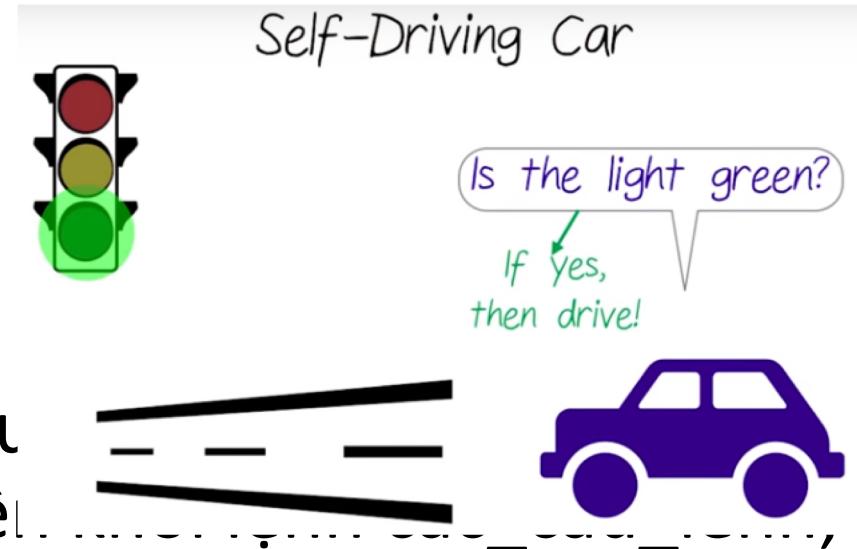
Lệnh if

- Cú pháp

```
if (dieu_kien) {  
    cac_cau_lenh;  
}
```

- Nếu biểu thức điều kiện dieu_kien nhận giá trị true thì thực hiện các câu lệnh.

```
boolean isLightGreen = ? ; //true or false  
  
if(isLightGreen) {  
    // traffic light is green  
    System.out.println("Drive!");  
}
```



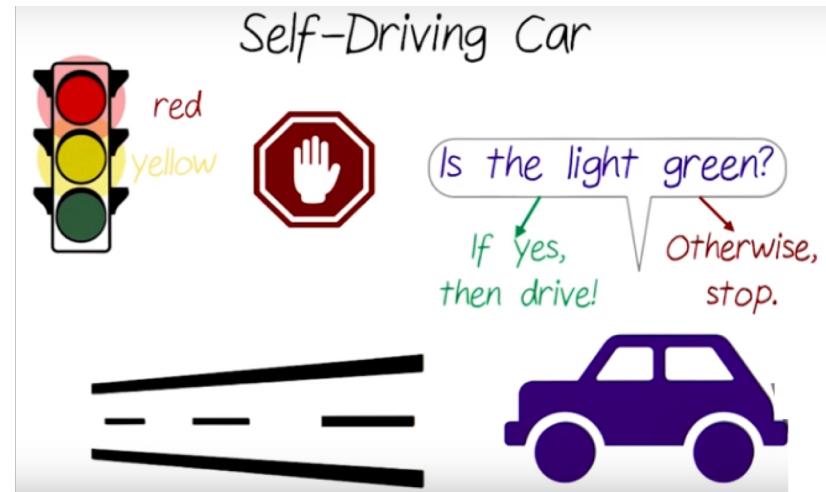
```
boolean isLightGreen = ? ; //true or false  
  
if(isLightGreen) {  
    // traffic light is green  
    System.out.println("Drive!");  
}  
  
skip down here
```

Lệnh if - else

❖ Cú pháp

```
if (dieu_kien) {  
    cac_cau_lenh_1;  
} else {  
    cac_cau_lenh_2;  
}
```

❖ Nếu biểu thức điều kiện (kiểu boolean) nhận giá trị true thì thực hiện khối lệnh cac_cau_lenh_1, là false thì thực hiện khối lệnh cac_cau_lenh_2.



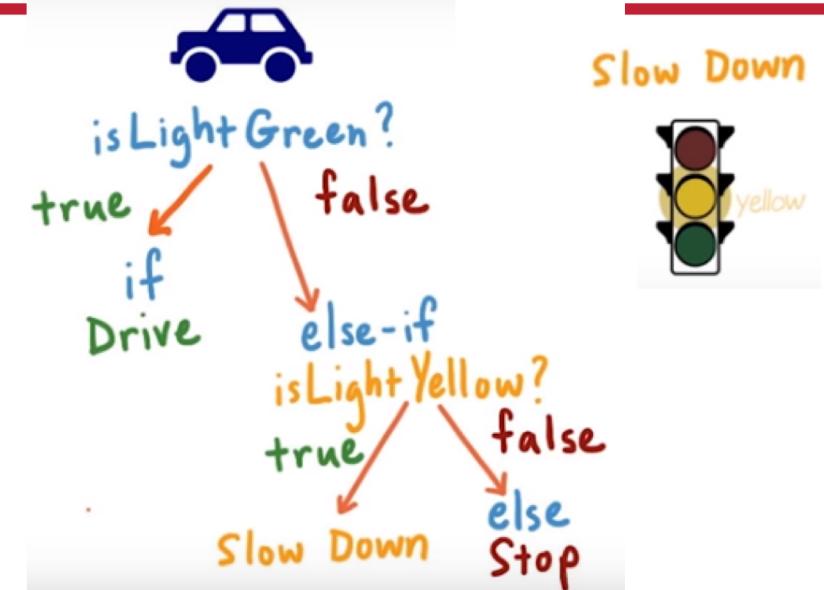
```
boolean isLightGreen = ? //true or false  
  
if(isLightGreen) {  
    //traffic light is green  
    System.out.println("Drive!");  
} else {  
    //light is NOT green  
    System.out.println("Stop.");  
}
```



Lệnh else-if

- Cú pháp

```
if (dieu_kien_1) {  
    cac_cau_lenh_1;  
} else if (dieu_kien_2) {  
    cac_cau_lenh_2;  
} else if (dieu_kien_3) {  
    cac_cau_lenh_3;  
} else {  
    cac_cau_lenh_n;  
}
```



Slow Down



Yellow

```
boolean isLightGreen = false; //true or false  
boolean isLightYellow = false; //true or false  
  
if(isLightGreen) {  
    //traffic light is green  
    System.out.println("Drive!");  
} else if(isLightYellow) {  
    //light is NOT green but is yellow  
    System.out.println("Slow down.");  
} else {  
    //light is neither green nor yellow  
    System.out.println("Stop.");  
}
```

- Có thể có nhiều else-if, chỉ có 1



Biểu thức điều kiện

- Toán tử so sánh

<u>Expression</u>	<u>Value</u>
<code>int x = 10;</code>	
<code>3 < 5</code>	true
<code>3 > 5</code>	false
<code>7 <= 6</code>	false
<code>x >= 10</code>	true
<code>x == 9</code>	false
<i>equality check</i>	
<code>x != 9</code>	true.
<i>NOT equal</i>	



Biểu thức điều kiện (2)

• Toán tử logic

Three main logical operators:

1) AND $3 < 5 \text{ && } 2 > 15 \rightarrow \text{false}$
 true false

2) OR $3 < 5 \text{ || } 2 > 15 \rightarrow \text{true}$

3) NOT $!(3 < 5) \rightarrow \text{false}$
 NOT true

turns a value into its opposite

Using multiple logical operators

&& will evaluate first, then ||

false && true || true $\rightarrow \text{true}$

false &&(true || true) $\rightarrow \text{false}$

Museum discount cases

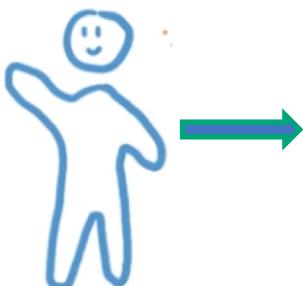
1. $\text{age} \leq 15$

ticket

$= \$5$

2. $\text{age} > 60$

3. isStudent



if(age ≤ 15 1 || age > 60 2 || isStudent 3) {

ticket = \$5

}

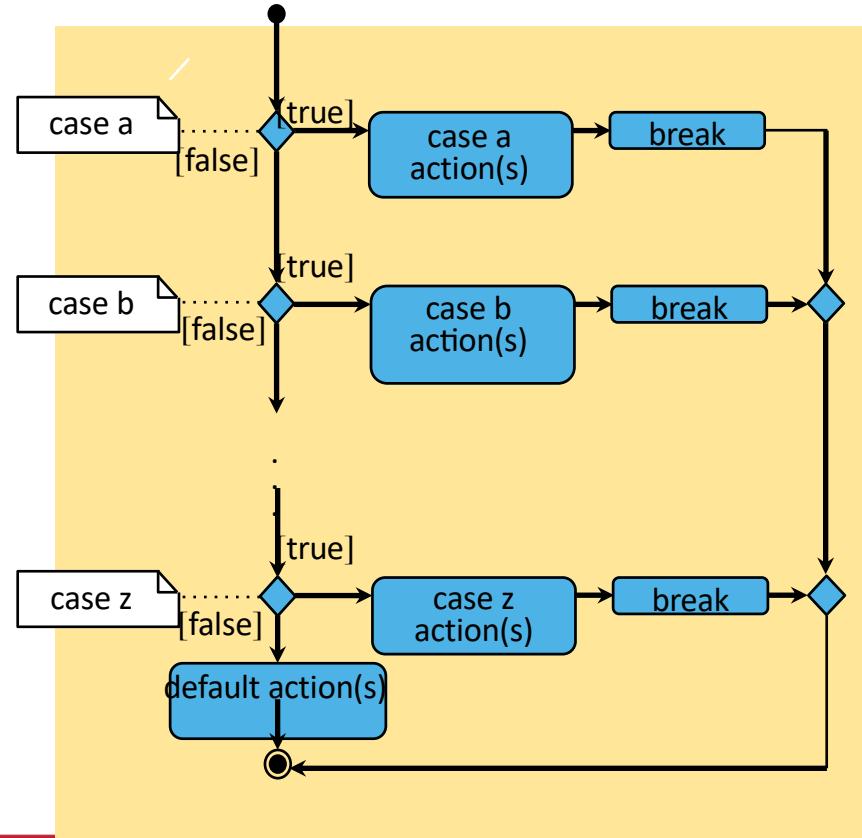
Ví dụ - Kiểm tra số chẵn – lẽ

```
class CheckNumber
{
    public static void main(String args[])
    {
        int num =10;
        if (num %2 == 0)
            System.out.println (num+ "la so chan");
        else
            System.out.println (num + "la so le");
    }
}
```



Lệnh switch - case

- Kiểm tra một biến đơn với nhiều giá trị khác nhau và thực hiện trường hợp tương ứng
 - break: Thoát khỏi lệnh switch-case
 - default kiểm soát các giá trị nằm ngoài các giá trị case:



Ví dụ - Lệnh switch - case (1)

```
public class Test {  
    public static void main(String args[]) {  
        int i = 2;  
  
        switch (i) {  
            case 1:  
                System.out.println("1");  
            case 2:  
                System.out.println("2");  
            case 3:  
                System.out.println("3");  
        }  
    }  
}
```



Ví dụ - Lệnh switch - case (2)

```
switch (day) {  
    case 0:  
    case 1:  
        rule = "weekend";  
        break;  
    case 2:  
    case 3:  
    case 4:  
    case 5:  
    case 6:  
        rule = "weekday";  
        break;  
    default:  
        rule = "error";  
}
```

```
if (day == 0 || day == 1) {  
    rule = "weekend";  
} else if (day > 1 && day < 7) {  
    rule = "weekday";  
} else {  
    rule = error;  
}
```



Bài tập: Tính số ngày trong tháng

- Input: Năm, tháng
- Output: số ngày trong tháng của năm đó
- Yêu cầu: sử dụng lệnh switch-case
- Gợi ý:
 - Tháng 1, 3, 5, 7, 8, 10, 12: 31 ngày
 - Tháng 4, 6, 9, 11: 30 ngày
 - Riêng tháng 2:
 - Năm thường: 28 ngày
 - Năm nhuận: 29 ngày (năm nhuận là “năm chia hết cho 4 và không chia hết cho 100”, hoặc là “năm chia hết cho 400”)



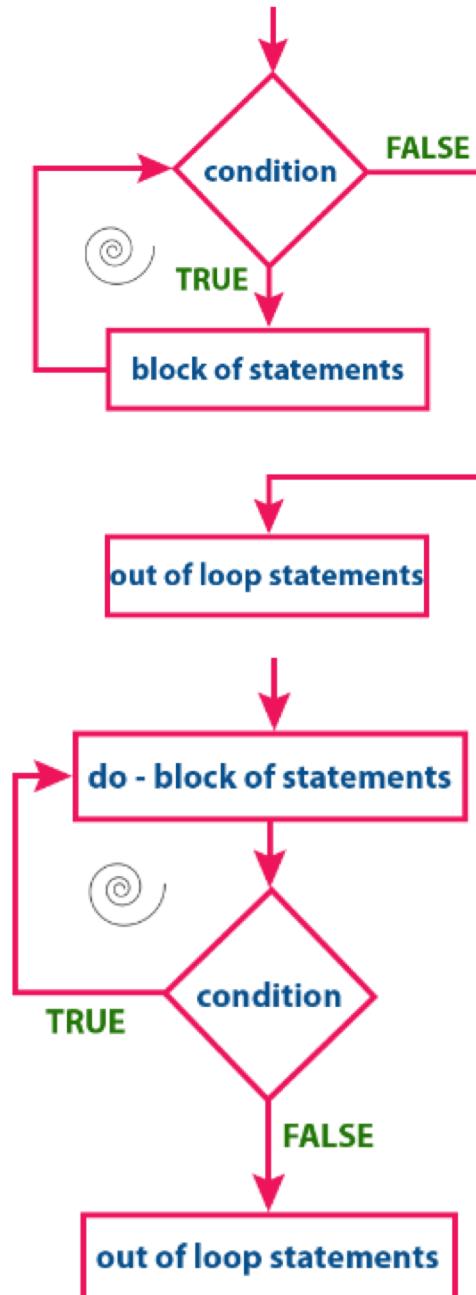
d. Vòng lặp while và do while

- Thực hiện một câu lệnh hoặc một khối lệnh
- khi điều kiện vẫn nhận giá trị true

```
while (condition) {  
    // code block to be executed  
}
```

```
do {  
    // code block to be executed  
}  
while (condition);
```

- while() thực hiện 0 hoặc nhiều lần
- do...while() thực hiện ít nhất một lần



Ví dụ - Vòng lặp while

```
int numOfWorkings = 5;  
int i = 1;           ← (1) loop counter  
while (i <= numOfWorkings) { ← (2) loop condition  
    System.out.println("Warning");  
    i++;               ← (3) loop increment  
}
```

Print output

```
Warning  
Warning  
Warning  
Warning  
Warning
```

i	i <= numOfWorkings
1	1 <= 5 (true)
2	2 <= 5 (true)
3	3 <= 5 (true)
4	4 <= 5 (true)
5	5 <= 5 (true)
6	6 <= 5 (false)

Chú ý: Cần tránh vòng lặp vô tận!



Ví dụ - Vòng lặp while (2)

```
class WhileDemo{  
    public static void main(String args[]) {  
        int a = 5, fact = 1;  
        while (a >= 1) {  
            fact *=a;  
            a--;  
        }  
        System.out.println("The Factorial of 5 is"+fact);  
    }  
}
```

Kết quả: “The factorial of 5 is 120” được hiển thị.

Viết thay lệnh while bằng lệnh do-while ?



Vòng lặp for

- Cú pháp:

```
for (start_expr; test_expr; increment_expr) {  
    // code to execute repeatedly  
}
```

- Ví dụ:

```
loop counter      loop condition      loop increment  
      (1)           (2)           (3)  
for(int i = 1; i <= numOfWorkWarnings ; i++){  
    System.out.println("Warning");  
}
```

- 3 biểu thức (1) (2) (3) đều có thể vắng mặt (thay bằng lệnh tương ứng trong khối lệnh)
- Có thể khai báo biến trong câu lệnh for
 - Thường sử dụng để khai báo một biến đếm
 - Thường khai báo trong biểu thức “start”
 - Phạm vi của biến giới hạn trong vòng lặp



Ví dụ - Vòng lặp for

```
class ForDemo {  
    public static void main(String args[]) {  
        int i=1, sum=0;  
        for (i=1;i<=10;i+=2) {  
            sum+=i;  
        System.out.println("Sum of first five odd numbers is" + sum);  
    }  
}
```

Kết quả: "Sum of first five odd numbers is 25" được hiển thị.



Vòng lặp for và while

- Các câu lệnh for và while cung cấp chức năng tương đương nhau
- Các cấu trúc lặp thường được sử dụng trong các tình huống khác nhau
 - while được sử dụng cho lặp từ đầu đến cuối
 - for được sử dụng để lặp với số vòng lặp xác định

```
int sum = 0;
for (int index = 1;index <= 10;index++)
{
    sum += index;
}
```

```
int sum = 0;
int index = 1;
while (index <= 10) {
    sum += index;
    index++;
}
```



Các lệnh thay đổi cấu trúc điều khiển

- break
 - Có thể được sử dụng để thoát ra ngoài câu lệnh switch
 - Kết thúc vòng lặp for, while hoặc do...while
 - Có hai dạng:
 - Gắn nhãn: Tiếp tục thực hiện câu lệnh tiếp theo sau vòng lặp được gắn nhãn
 - Không gắn nhãn: Thực hiện câu lệnh tiếp theo bên ngoài vòng lặp



4.5. Các lệnh thay đổi cấu trúc điều khiển (2)

- continue
 - Có thể được sử dụng cho vòng lặp for, while hoặc do...while
 - Bỏ qua các câu lệnh còn lại của vòng lặp hiện thời và chuyển sang thực hiện vòng lặp tiếp theo.



Ví dụ - break và continue

```
public int myMethod(int x) {  
    int sum = 0;  
    outer: for (int i=0; i<x; i++) {  
        inner: for (int j=i; j<x; j++) {  
            sum++;  
            if (j==1) continue;  
            if (j==2) continue outer;  
            if (i==3) break;  
            if (j==4) break outer;  
        }  
    }  
    return sum;  
}
```



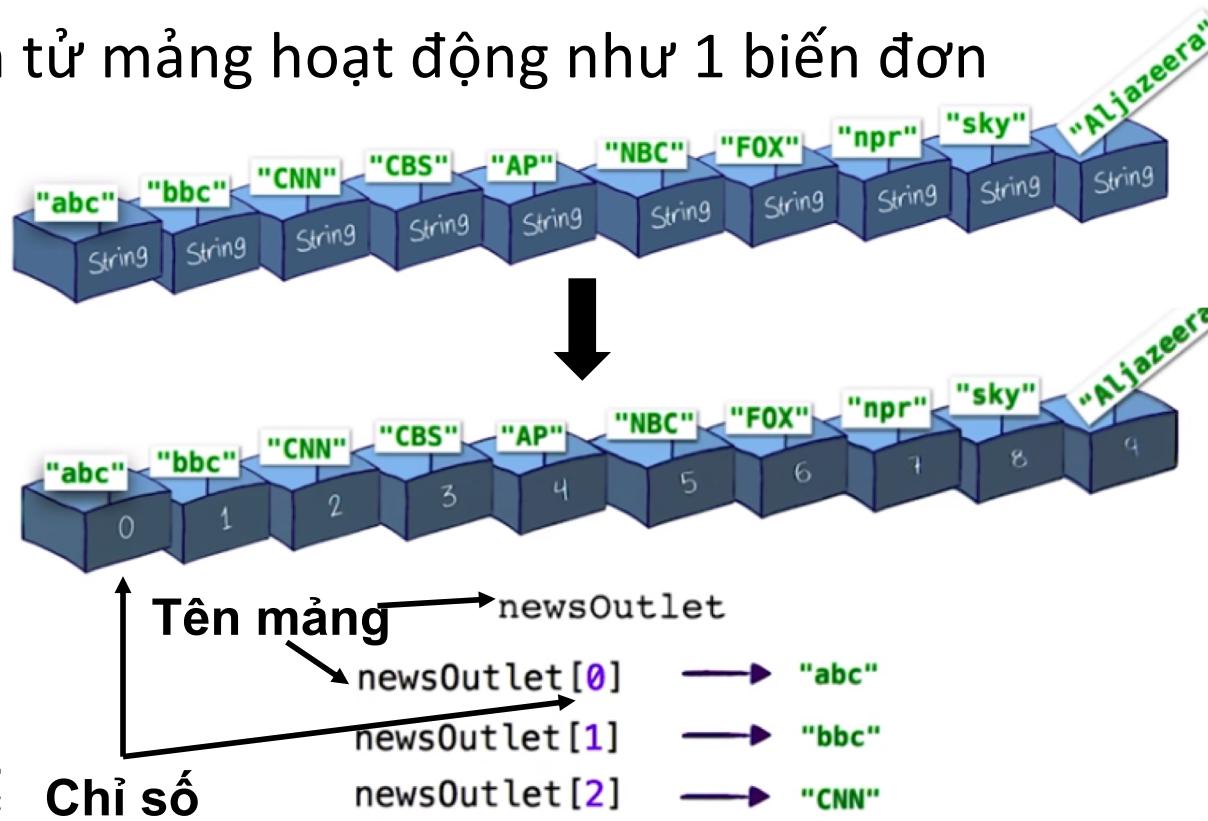
1. Cơ bản về Java

- 1.1. Các khái niệm cơ bản
- 1.2. Biến
- 1.3. Các kiểu dữ liệu cơ bản
- 1.4. Toán tử
- 1.5. Cấu trúc điều khiển
- 1.6. Mảng**



Khái niệm Mảng (array)

- Dùng để lưu một tập hợp hữu hạn các phần tử cùng kiểu (nguyên thuỷ hoặc đối tượng), liền kề nhau trong bộ nhớ.
 - Mỗi mảng có 1 tên gọi
 - Các phần tử được đánh số thứ tự, bắt đầu từ 0
 - Mỗi phần tử mảng hoạt động như 1 biến đơn



Khai báo và khởi tạo mảng

- Mảng phải khai báo trước khi sử dụng. Kích thước của một mảng sau khai báo sẽ không thể thay đổi
- Cú pháp:

```
kieu_dulieu[] ten_mang = new  
kieu_dulieu[KichThuoc_MANG];  
  
kieu_dulieu ten_mang[] = new  
kieu_dulieu[KichThuoc_MANG];  
kieu_dl[] ten_mang = {ds_gia_tri_cac_ptu};
```

- Nếu không khởi tạo → tất cả các phần tử của mảng nhận giá trị mặc định tùy thuộc vào kiểu dữ liệu.



Khai báo và khởi tạo mảng

Cách khai báo	Mô tả	Cú pháp	Ví dụ
Chỉ đơn thuần khai báo mảng	Chỉ đơn thuần khai báo mảng	Datatype identifier[]	char ch[] ; khai báo mảng ký tự có tên ch
Khai báo và tạo mảng	Khai báo và cấp phát bộ nhớ cho các phần tử mảng sử dụng toán tử new	Datatype identifier[] = new datatype [size]	char ch[] = new char [10] ; Khai báo một mảng ch và lưu trữ 10 ký tự
Khai báo và khởi tạo các phần tử mảng	Khai báo mảng, cấp phát bộ nhớ cho nó và gán các giá trị ban đầu cho các phần tử của mảng	Datatype identifier[] = {value1, value2 ... valueN} ;	char ch [] = { 'A' , 'B' , 'C' , 'D' } ; khai báo mảng ch và lưu 4 chữ cái kiểu ký tự



Ví dụ - mảng

Tên của mảng (tất cả các thành phần trong mảng có cùng tên, c)

c.length: cho biết độ dài của mảng c

Chỉ số (truy nhập đến các thành phần của mảng thông qua chỉ số)

c[0]
c[1]
c[2]
c[3]
c[4]
c[5]
c[6]
c[7]
c[8]
c[9]
c[10]
c[11]

-45
6
0
72
1543
-89
0
62
-3
1
6453
78



Khai báo và khởi tạo mảng

- Ví dụ:

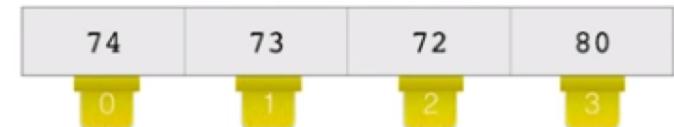
```
int MAX = 5;  
boolean bit[] = new boolean[MAX];  
float[] value = new float[2*3];  
int[] number = {10, 9, 8, 7, 6};  
System.out.println(bit[0]); // prints "false"  
System.out.println(value[3]); // prints "0.0"  
System.out.println(number[1]); // prints "9"
```



Làm việc với mảng

- Dùng thuộc tính `.length` để lấy kích thước của một mảng
- Lưu ý không truy cập vào các chỉ số không thuộc mảng, ví dụ chỉ số âm, chỉ số \geq kích thước mảng.
- Duyệt tất cả các phần tử trong mảng: dùng vòng lặp.

temperatures:



```
int size = temperatures.length; ← 4
```

```
System.out.println(temperatures[10]); Error!
```

ArrayIndexOutOfBoundsException

```
for(int i=0; i<size; i++){  
    // truy cập temperatures[i];  
}
```

loop counter: (0, 1, 2 ...)



Mảng 2 chiều

- Có thể khai báo và sử dụng một mảng nhiều chiều.
- Mảng 2 chiều giống một bảng với các dòng và cột.

	Gale	Tim	Sandra	Sam	Eric
Maths	87	93	99	75	60
English	93	70	98	90	75
Biology	82	75	95	80	66
Arts	90	75	99	85	70



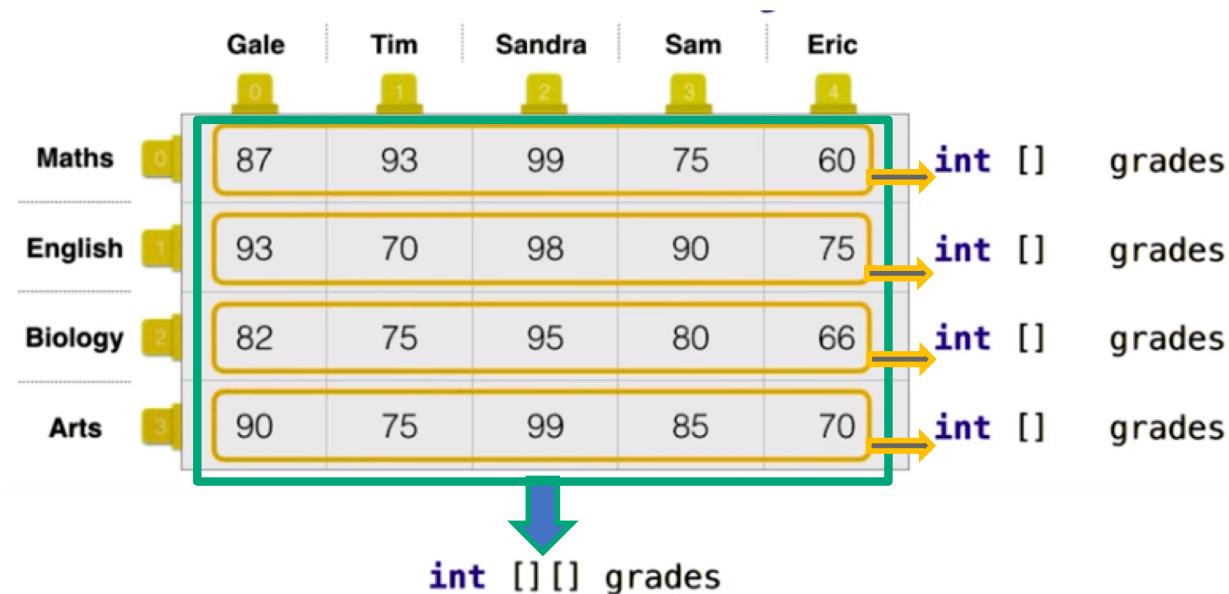
2D Array

	Gale	Tim	Sandra	Sam	Eric	
Maths	0	87	93	99	75	60
English	1	93	70	98	90	75
Biology	2	82	75	95	80	66
Arts	3	90	75	99	85	70

Mảng 2 chiều (2)

- Mảng 2 chiều: coi như 1 mảng của các phần tử A, mỗi phần tử A lại là 1 mảng các phần tử B.
- Khai báo mảng 2 chiều

```
kieu_dulieu[] [] ten_mang;
```



Mảng 2 chiều (3)

- Truy cập phần tử trong mảng:

ten_mang [chi_so_hang] [chi_so_cot]

- Duyệt tất cả các phần tử trong mảng:

- Dùng vòng lặp lồng nhau

	Gale	Tim	Sandra	Sam	Eric
Maths	87	93	99	75	60
English	93	70	98	90	75
Biology	82	75	95	80	66
Arts	90	75	99	85	70

grades [2] [1] = 75
array #2 (Biology)
item #1 (Tim)

```
for(int i=0; i<4; i++){  
    for (int j=0; j<5; j++){  
        // truy cập grades[i][j];  
    }  
}
```

Bài tập

- Bài tập 1: Viết chương trình tráo đổi ngẫu nhiên vị trí một dãy số cho trước
 - Để lấy một số int ngẫu nhiên từ 0 đến n-1 ta dùng lệnh
 - `int i = Random.nextInt(n);`
- Bài tập 2: Viết chương trình sắp xếp một dãy số theo thứ tự tăng dần, dãy số được nhập từ bàn phím.
- Bài tập 3: Viết chương trình nhập chiều cao h từ bàn phím, sau đó hiển thị các tam giác hình sao có chiều cao h như dưới đây. Chú ý có kiểm tra điều kiện của h: $2 \leq h \leq 10$. Nếu h nằm ngoài đoạn trên, yêu cầu người dùng nhập lại.
- Bài tập 4: Nhập vào kích thước ô vuông $n*n$, kiểm tra $3 \leq n \leq 8$. Hiển thị ra màn hình kết quả như ví dụ sau.

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7



2. Giới thiệu về UML

2.1. UML là gì

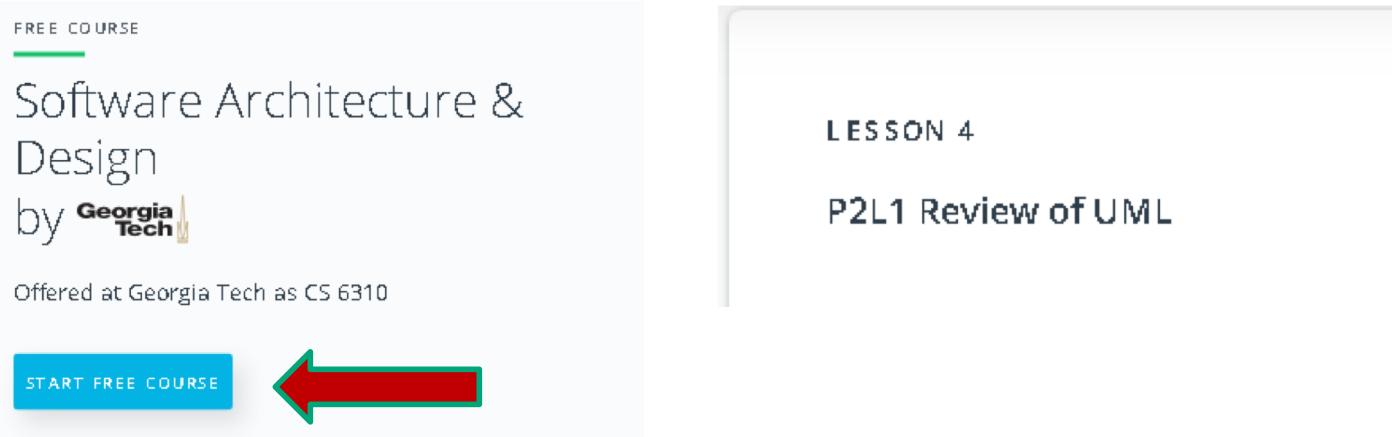
2.2. Các biểu đồ UML cơ bản

2.3. Ví dụ



Bài giảng e-learning tham khảo

- Đăng ký tài khoản trên <https://www.udacity.com>
- Theo dõi bài giảng “Software Architecture & Design”:
<https://www.udacity.com/course/software-architecture-design--ud821>
- Tập trung vào Lesson 4, các bài khác tham khảo thêm



The image shows a screenshot of the Udacity website for the "Software Architecture & Design" course. On the left, there's a large blue button labeled "START FREE COURSE". To the right of this button, a red arrow points towards the course content area. Inside this area, the word "LESSON 4" is at the top, followed by "P2L1 Review of UML". The course title "Software Architecture & Design" is displayed above the lesson, along with the provider "by Georgia Tech". Below the provider, it says "Offered at Georgia Tech as CS 6310".



Bài giảng e-learning

Cấu trúc bài giảng e-learning

The screenshot shows a user interface for an e-learning platform. On the left, there's a sidebar with a dark blue header "Lesson 4: P2L1 Review of UML". Below the header are three sections: "SEARCH" with a magnifying glass icon, "RESOURCES" with an upward arrow icon, and "CONCEPTS" with a downward arrow icon. A large orange arrow points downwards from the "CONCEPTS" section towards the main content area. The main content area has a title "Diagrams" and a "SEND FEEDBACK" button. It features a video of a man speaking, with a logo for "UML" consisting of three overlapping colored blocks (pink, yellow, purple) to his right. The word "Diagrams" is written in red below the logo. A subtitle at the bottom of the video frame reads: "way of doing that is with diagrams. This course focused on UML, using UML and". At the bottom of the video player are standard controls: a play button, a volume icon, a progress bar showing "0:10 / 1:05", and icons for "cc", "YouTube", and a share button.

Chọn phụ đề nếu cần



Bài giảng e-learning tham khảo

- Bài giảng của Smartdraw

- <https://www.smartdraw.com/uml-diagram/>
- <https://www.smartdraw.com/use-case-diagram/>
- <https://www.smartdraw.com/activity-diagram/>
- <https://www.smartdraw.com/sequence-diagram/>
- <https://www.smartdraw.com/class-diagram/>



2. Giới thiệu về UML

2.1. UML là gì

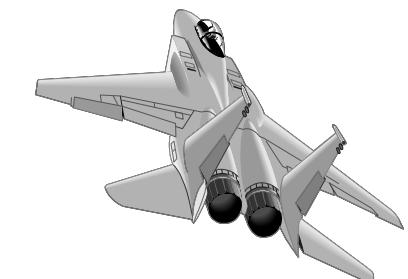
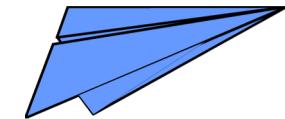
2.2. Các biểu đồ UML cơ bản

2.3. Ví dụ



Tầm quan trọng của phân tích và thiết kế

- Hướng tiếp cận không có phân tích – thiết kế:
 - Bắt đầu lập trình ngay khi có được yêu cầu
 - Mất rất nhiều thời gian và tạo đi tạo lại nhiều mã nguồn
 - Không có bất kỳ một kiến trúc nào
 - Phải chịu khổ với những lỗi phát sinh
- Hướng tiếp cận có phân tích – thiết kế:
 - Chuyển các yêu cầu của bài toán thành một bản thiết kế rõ ràng
 - Tập trung vào phân tích các YÊU CẦU và thiết kế các MÔ HÌNH cho hệ thống TRƯỚC khi lập trình



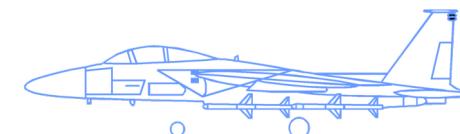
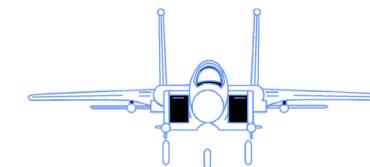
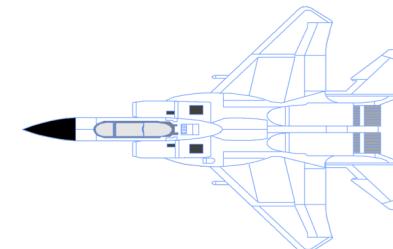
Tầm quan trọng của phân tích và thiết kế (2)

- Ưu điểm của việc PTTK hệ thống:
 - Đơn giản hóa thế giới thực bằng các mô hình
 - Mô tả đúng, đồng nhất cấu trúc, cách ứng xử của HT trong suốt quá trình xây dựng
 - Đảm bảo mục đích và yêu cầu của HT được thỏa mãn trước khi xây dựng
 - Cung cấp cho người dùng, khách hàng, kỹ sư phân tích, thiết kế, kỹ sư lập trình nhiều cái nhìn khác nhau về cùng một HT
 - Ghi lại các quyết định của nhà phát triển để sử dụng sau này

Máy bay phản lực



Các góc nhìn



- UML: viết tắt của “Unified Modeling Language” là một Ngôn ngữ mô hình hóa được thống nhất
- UML là ngôn ngữ trực quan để:
 - trực quan hóa (visualizing)
 - đặc tả (specifying)
 - xây dựng (constructing)
 - tài liệu hóa (documenting)các cấu phần của một hệ thống phần mềm
- Giúp công việc phát triển được xử lý nhất quán, giảm thiểu lỗi xảy ra
 - Giúp dễ hình dung hơn cấu trúc của hệ thống
 - Hiệu quả hơn trong việc liên lạc, trao đổi

Lịch sử phát triển UML

- Vào năm 1994, có hơn 50 phương pháp mô hình hóa hướng đối tượng:

- Fusion, Shlaer-Mellor, ROOM, Class-Relation,Wirfs-Brock, Coad-Yourdon, MOSES, Syntropy, BOOM, OOSD, OSA, BON, Catalysis, COMMA, HOOD, Ooram, DOORS ...
- Mô tả về mô hình “Meta-models” tương đồng với nhau
- Các ký pháp đồ họa khác nhau
- Quy trình khác nhau hoặc không rõ ràng

→ Cần chuẩn hóa và thống nhất các phương pháp

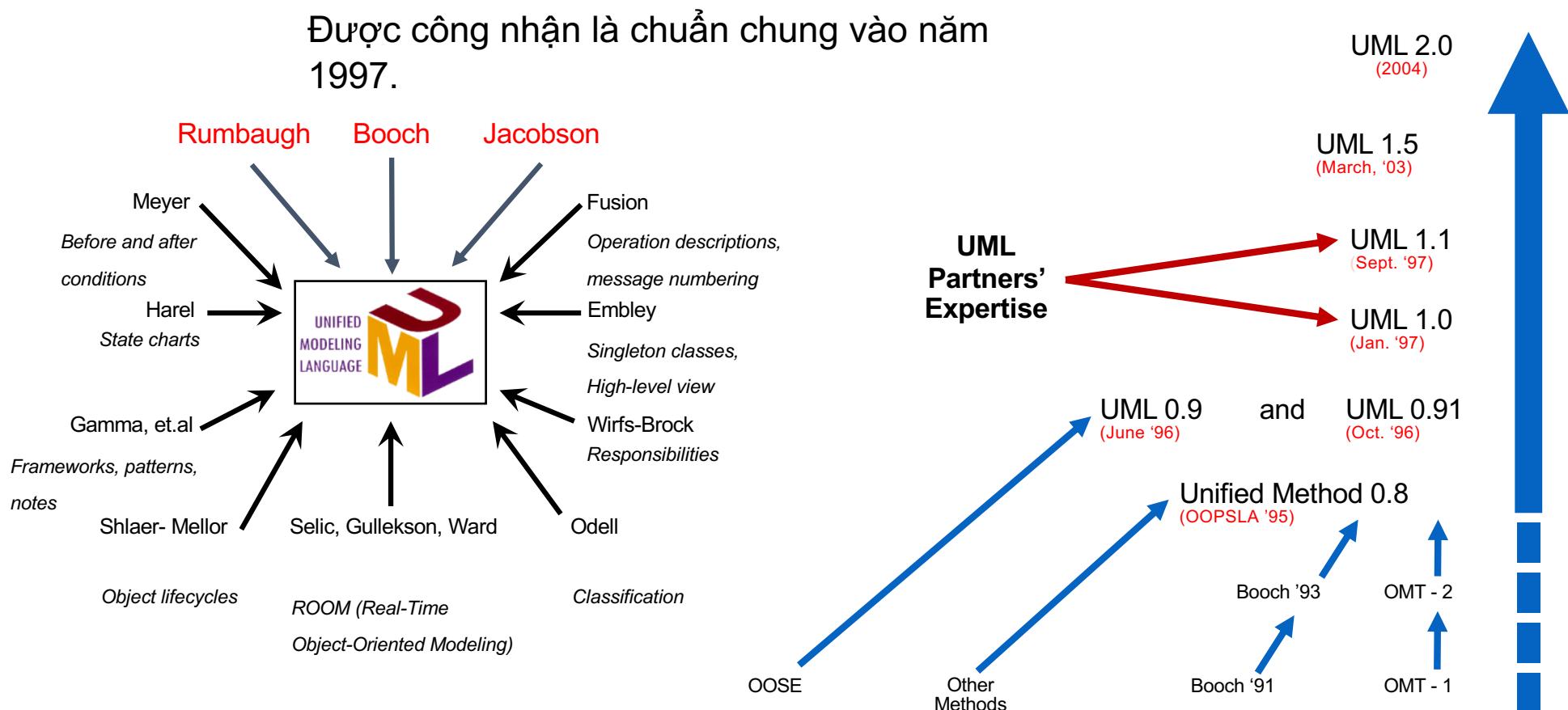
- UML được 3 chuyên gia hướng đối tượng hợp nhất các kỹ thuật của họ vào năm 1994:

- Booch91 (Grady Booch): Conception, Architecture
- OOSE (Ivar Jacobson): Use cases
- OMT (Jim Rumbaugh): Analysis



Lịch sử phát triển UML (2)

- UML là ngôn ngữ hợp nhất các mô hình khác nhau



- Các mô hình UML có thể kết nối trực tiếp với rất nhiều ngôn ngữ lập trình.
 - Ánh xạ sang Java, C++, Visual Basic...
 - Các bảng trong RDBMS hoặc kho lưu trữ trong OODBMS
 - Cho phép các kỹ nghệ xuôi (chuyển UML thành mã nguồn)
 - Cho phép kỹ nghệ ngược (xây dựng mô hình hệ thống từ mã nguồn)
- Các công cụ UML
 - Công cụ mã nguồn mở: EclipseUML, UmlDesigner, StarUML, Argo UML, ...
 - Công cụ thương mại: Enterprise Architect, IBM Rational Software Architect, Microsoft Visio, Visual Paradigm for UML, SmartDraw...

2. Giới thiệu về UML

2.1. UML là gì

2.2. Các biểu đồ UML cơ bản

2.3. Ví dụ



2.1. Biểu đồ UML

- Biểu đồ:
 - là các hình vẽ bao gồm các ký hiệu phần tử mô hình hóa
 - minh họa một thành phần cụ thể hay một khía cạnh cụ thể của hệ thống.
- Một mô hình hệ thống thường có nhiều loại biểu đồ, mỗi loại gồm nhiều biểu đồ khác nhau.
- Một biểu đồ là một thành phần của một hướng nhìn cụ thể
- Một số loại biểu đồ có thể là thành phần của nhiều hướng nhìn khác nhau
- UML thế hệ 2 có tới 13-14 loại biểu đồ. Trong một project, chỉ sử dụng những biểu đồ phù hợp nhất



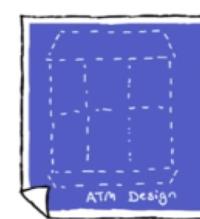
2.1. Biểu đồ UML (2)

- Phân biệt:

- Biểu đồ cấu trúc: mô tả thành phần tinh, luôn có của hệ thống và mối quan hệ giữa chúng
- Biểu đồ hành vi: mô tả cách hoạt động của hệ thống

Diagram Types

Two Main Categories:



Structural



Behavioral

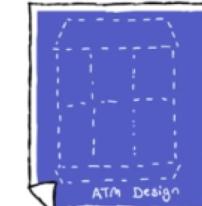
what is contained in a system

what must happen in a system



Biểu đồ cấu trúc

- Biểu đồ cấu trúc tĩnh
 - Biểu đồ lớp (Class Diagram)
 - Biểu đồ đối tượng (Object Diagram)
 - Biểu đồ gói (Package diagram)
- Biểu đồ thực thi
 - Biểu đồ thành phần (Component Diagram)
 - Biểu đồ triển khai (Deployment Diagram)
 - Biểu đồ cấu thành (Composite Diagram)
- Biểu đồ profile (Profile Diagram)



Structural



Biểu đồ hành vi

- Biểu đồ use case (Use Case Diagram)
- Biểu đồ hoạt động (Activity Diagram)
- Biểu đồ tương tác
 - Biểu đồ tổng quát (Interaction overview diagram)
 - Biểu đồ trình tự (Sequence Diagram)
 - Biểu đồ giao tiếp/cộng tác (Communication/Collaboration Diagram)
- Biểu đồ trạng thái (State Diagram)
- Biểu đồ thời gian (Timing Diagram)



Behavioral



2. Giới thiệu về UML

2.1. UML là gì

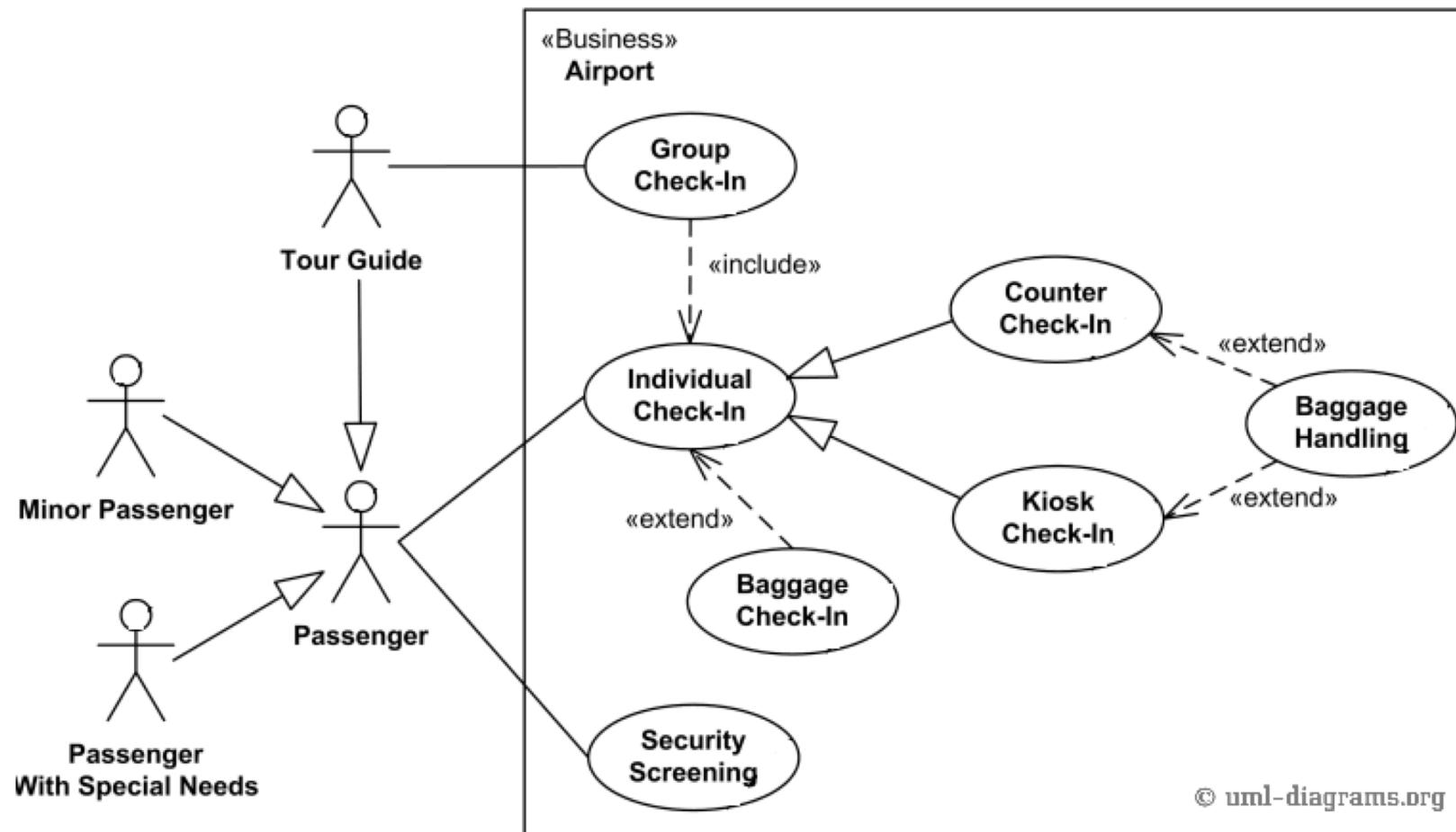
2.2. Các biểu đồ UML cơ bản

2.3. Ví dụ



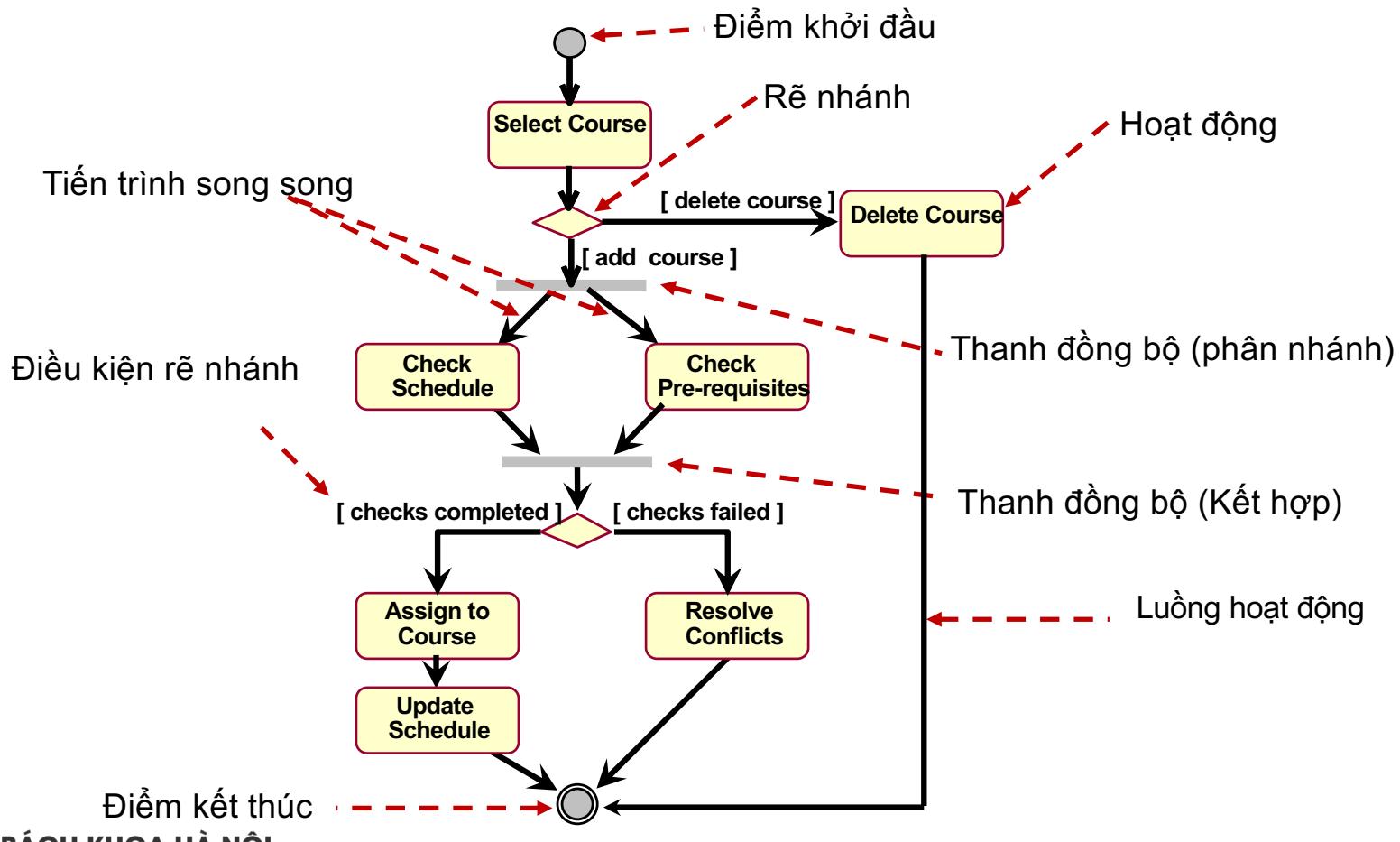
Biểu đồ Use case

- Mô hình chức năng hệ thống với các tác nhân và use case



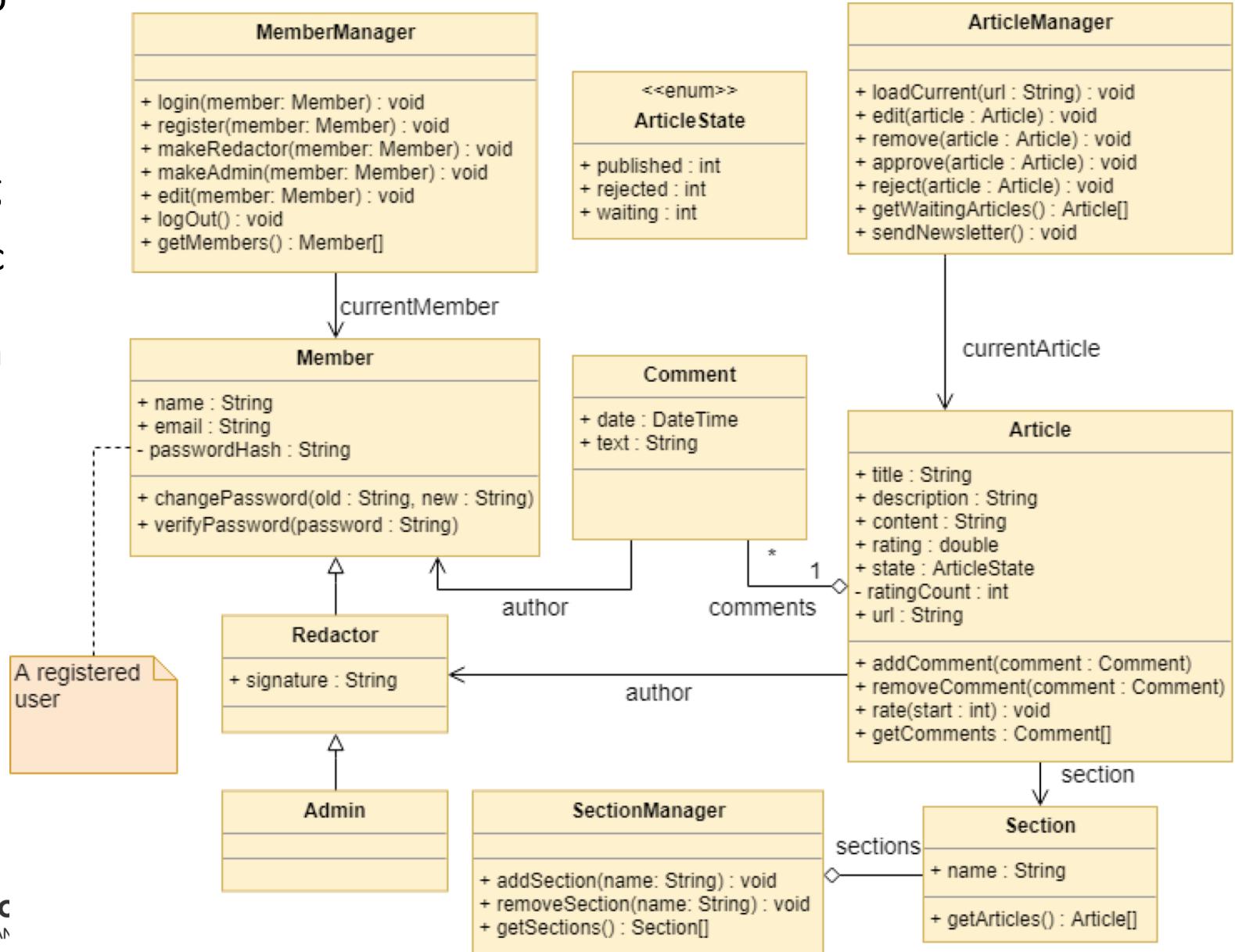
Biểu đồ hoạt động

- Biểu đồ hoạt động biểu diễn chuỗi các hoạt động hoặc luồng điều khiển có thứ tự của hệ thống thực hiện trong một use case



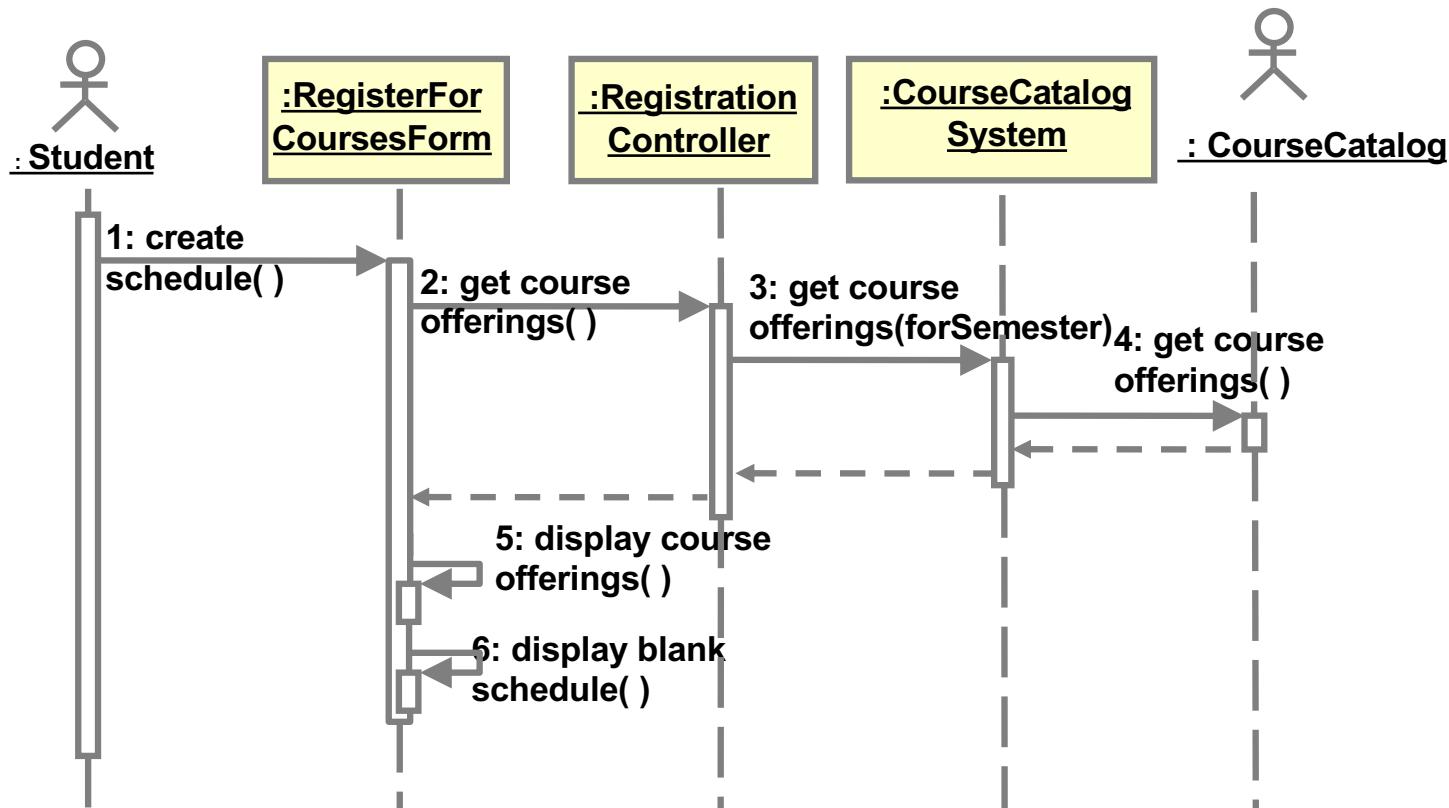
Biểu đồ lớp

- Dạng biểu đồ phổ biến nhất, mô tả cấu trúc tĩnh của hệ thống
- Biểu diễn các lớp và mối quan hệ giữa các lớp



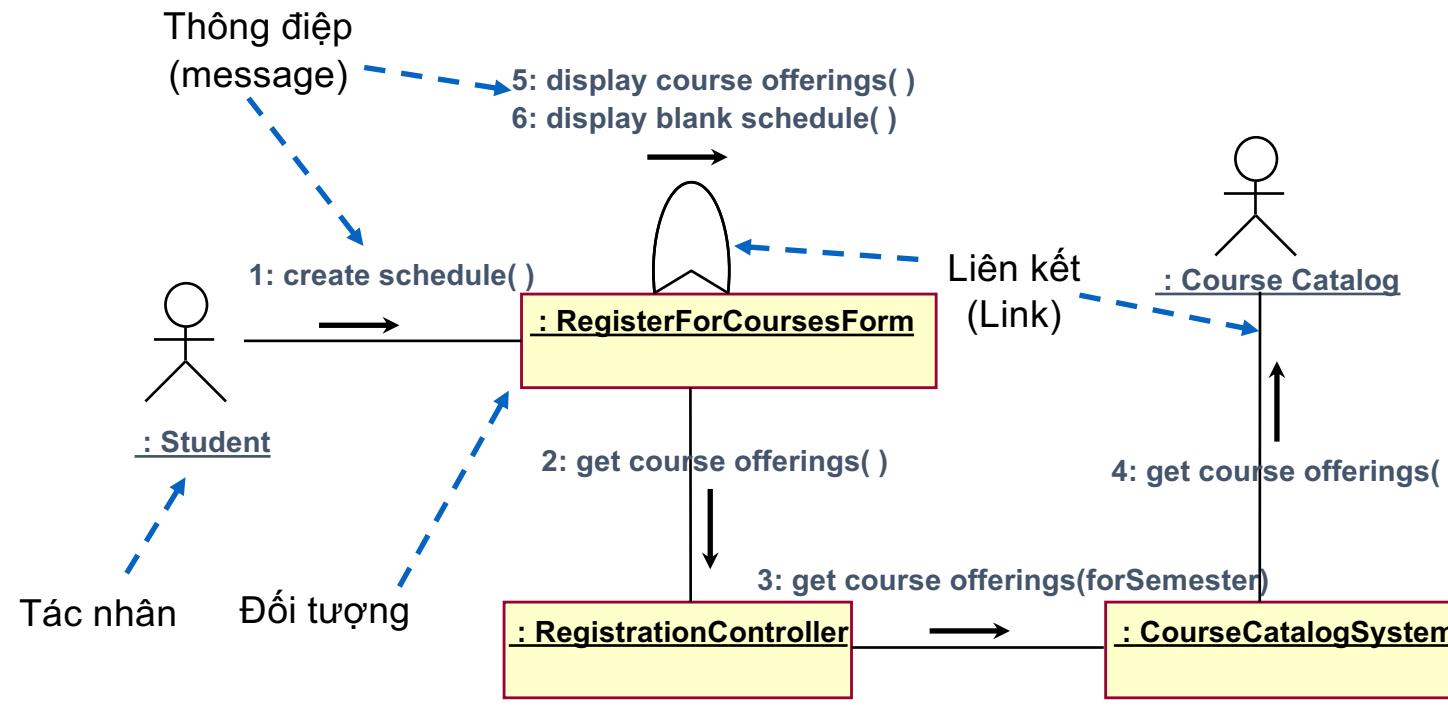
Biểu đồ tuần tự

- Là một loại biểu đồ tương tác, biểu diễn trình tự trao đổi thông điệp giữa các đối tượng theo thời gian trong một use case



Biểu đồ giao tiếp

- Cũng là một loại biểu đồ tương tác, nhưng nhấn mạnh vào việc tổ chức các đối tượng tham gia vào tương tác hơn là trình tự trao đổi thông điệp giữa các đối tượng





HUST

THANK YOU !