

# Java Core #1

Lục Ngọc



**Lộ trình**



## Nội dung học phần

- ☐ Buổi 1 - Java cơ bản 1: Giới thiệu, cài đặt, biến, Toán tử
- ☐ Buổi 2 - Java cơ bản 2: Chuỗi, ngày giờ, xuất nhập
- ☐ Buổi 3 - Java cơ bản 3: Câu lệnh rẽ nhánh, vòng lặp
- ☐ Buổi 4 - Java cơ bản 4: Mảng, phương thức, thực hành
- ☐ Buổi 5 - OOP 1: Lớp, đối tượng, thuộc tính, phương thức
- ☐ Buổi 6 - OOP 2: Đóng gói, kế thừa
- ☐ Buổi 7 - OOP 3: Trừu tượng, đa hình
- ☐ Buổi 8 - Thực hành lập trình OOP



# Cấu trúc mỗi buổi học

## Trên lớp

- Tóm tắt lại nội dung bài trước
- Hướng dẫn nội dung bài mới
- Thực hành hoặc chữa bài tập
- Giải đáp thắc mắc

## Về nhà

- Xem chi tiết bài học trên [techmaster.vn](https://techmaster.vn)
- Làm bài tập về nhà
- Xem trước nội dung cho buổi sau
- Hỏi đáp với mentor trên group chung

# Buổi 1: Java cơ bản 1



## Nội dung buổi học

- Giới thiệu về ngôn ngữ java
- Hướng dẫn cài đặt môi trường
- Tạo chương trình đầu tiên
- Biến & Kiểu dữ liệu
- Toán tử trong java
- Các phép toán học

# Giới thiệu về java

# Nội dung bài học bao gồm:

- ☐ Ngôn ngữ java và các sự thật thú vị
- ☐ Cài đặt môi trường
- ☐ Cấu trúc của một chương trình java
- ☐ Chương trình java đầu tiên
- ☐ Các thức code java chạy



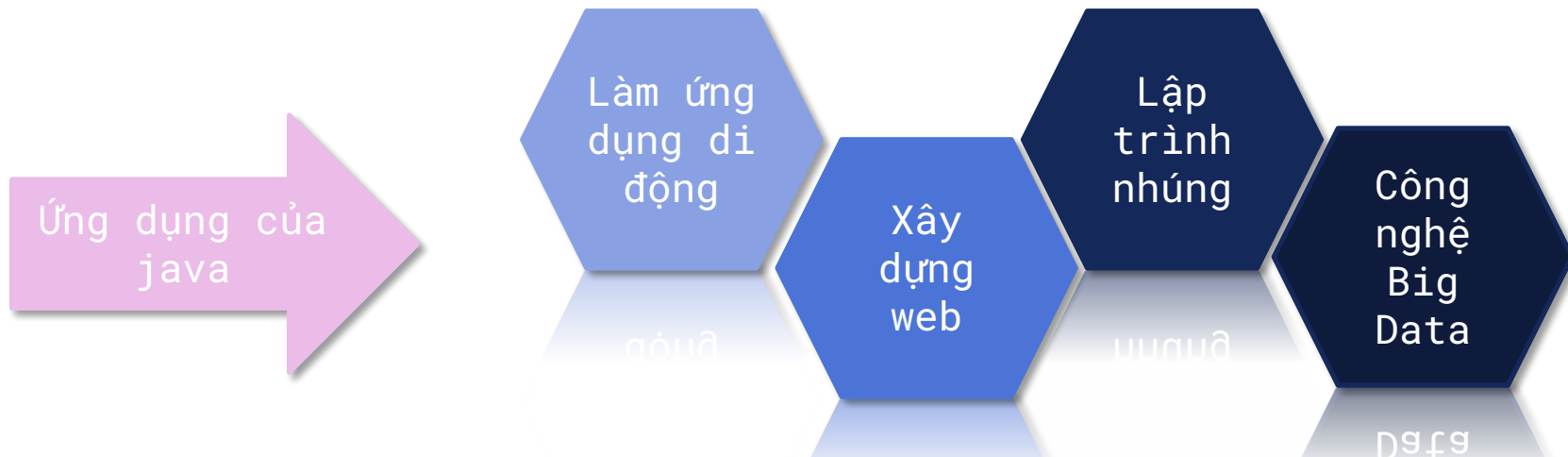




# Ngôn ngữ java và các sự thật thú vị

Java là một trong những ngôn ngữ lập trình hướng đối tượng phổ biến nhất hiện nay

Nó giúp các lập trình viên phát triển các ứng dụng có thể chạy trên nhiều thiết bị phần cứng và hệ điều hành khác nhau



**1991**

Do Jame Gosling và các  
đồng nghiệp của công  
ty Sun Microsystems  
khởi xướng

**1995**

Chính thức  
phát hành

**Now**

Trở thành một  
trong những  
ngôn ngữ lập  
trình phổ biến  
nhất hiện nay

**Lịch sử phát triển java**



# Đặc tính của Java

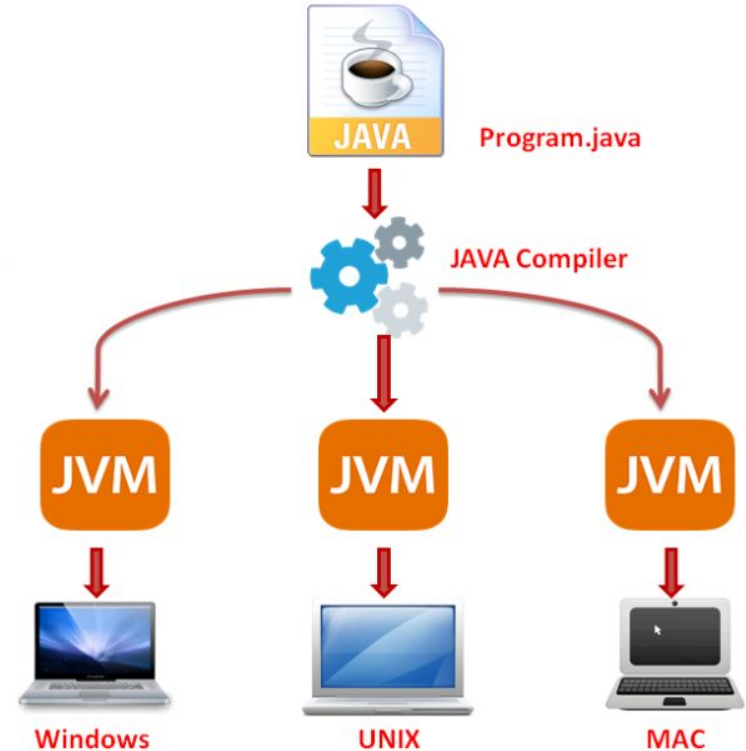




# JVM & bytecode

Máy ảo java (Java Virtual Machine – JVM) là một máy ảo cho phép chạy các chương trình Java cũng như các chương trình khác được viết bằng ngôn ngữ khác mà biên dịch sang mã máy

Ngôn ngữ dành cho máy ảo java được gọi là java bytecode, hay ngắn gọn là bytecode



## Các nền tảng java



**01** Java card

**02** Java Platform, Micro Edition

**03** Java Platform, Standard Edition

**04** Java Platform, Enterprise Edition



# Cài đặt môi trường

Cài đặt JDK:

[https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html](https://www.oracle.com/java/technologies/javase/jdk17-archhive-downloads.html)

Cài đặt Visual Studio Code:

<https://code.visualstudio.com/download>



# Cấu trúc của một chương trình java

```
package <package_name>;  
import <other_package>;  
  
public class <Class_name> {  
    <Variables>;  
    <Method>;  
}
```

Trong đó:

**package**: Một package (gói) mô tả không gian tên có chứa các lớp của java, ta có thể xem package như một thư mục

**import**: Nhằm sử dụng để xác định các class hoặc package được sử dụng trong lớp này

**public**: Xác định phạm vi truy cập của lớp

**class**: Từ khóa nhằm định nghĩa lớp của java

**Variables**: Biến

**Method**: Phương thức



# Chương trình đầu tiên

- Nội dung chương trình:

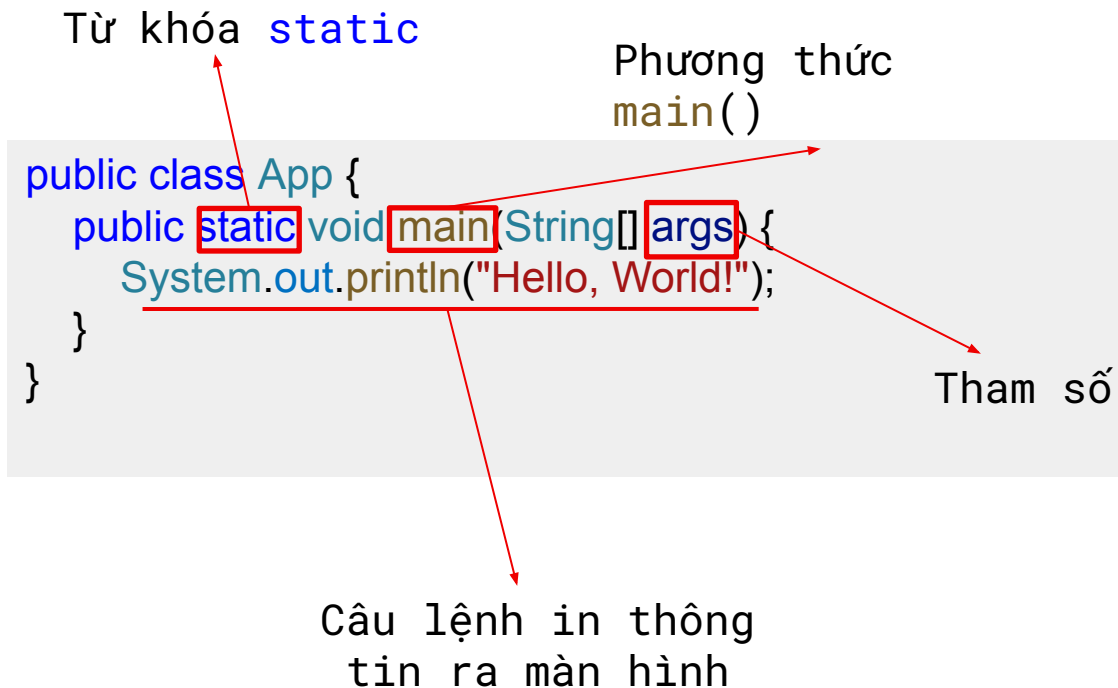
```
public class App {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

- Kết quả nhận được:

Hello, World!



# Cách thức code java chạy



The diagram illustrates the components of a Java code snippet. It features a light gray rectangular box containing the following code:

```
public class App {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Four red arrows point from external text labels to specific parts of the code:

- An arrow points from the text "Từ khóa `static`" to the `static` keyword in the `main` method signature.
- An arrow points from the text "Phương thức `main()`" to the `main` method name.
- An arrow points from the text "Tham số" to the `args` parameter in the `main` method signature.
- An arrow points from the text "Câu lệnh in thông tin ra màn hình" to the `System.out.println("Hello, World!");` statement.

# Types

## (biến và kiểu dữ liệu)

Biến là vùng nhớ dùng để lưu trữ các giá trị của chương trình. Mỗi biến gắn liền với một kiểu dữ liệu và một định danh duy nhất gọi là tên biến.





# Biến

Cú pháp khai báo biến:

```
<Kiểu dữ liệu> <Tên biến>;
```

Gán giá trị cho biến:

```
<Tên biến> = <Giá trị>;
```



# Biến

## Biến toàn cục

Được khai báo trong một lớp, bên ngoài constructor, phương thức, block. Được sử dụng với phạm vi truy cập

## Biến cục bộ

Biến chỉ có thể truy xuất trong khối lệnh nó khai báo



# Biến

Biến toàn cục

```
public class Variable {  
    int globalVariable;  
    public static void main(String[] args){  
        int localVariable;  
    }  
}
```

Biến cục bộ



# Kiểu dữ liệu

1

Các kiểu dữ liệu nguyên thủy (Primitive Types)



2

Các kiểu dữ liệu tham chiếu (Reference Types)



# Kiểu primitive







## Kiểu số nguyên

Java cung cấp 4 kiểu số nguyên khác nhau là: byte, short, int, long.

Kiểu dữ liệu	Miền giá trị	Giá trị mặc định	Kích cỡ mặc định
byte	-128 đến 127	0	1 byte
short	-32768 đến 32767	0	2 byte
int	$-2^{31}$ đến $2^{31}-1$	0	4 byte
long	$-2^{63}$ đến $2^{63}-1$	0L	8 byte



# Kiểu số nguyên

Ví dụ:

```
byte a = 5;  
short b = 10;  
int c = 20;  
long d = 100L;
```



## Kiểu số thực

Đối với kiểu dấu chấm động hay kiểu thực, java hỗ trợ hai kiểu dữ liệu là float và double.

Số kiểu dấu chấm động không có giá trị nhỏ nhất cũng không có giá trị lớn nhất.

Kiểu dữ liệu	Giá trị mặc định	Kích cỡ mặc định
float	0.0f	4 byte
double	0.0d	8 byte



## Kiểu số thực

```
float a = 2.5f;
```

```
double b = 2.5d;
```

```
double c = 2.5; //Vì double là kiểu mặc định cho kiểu số thực, nên có thể viết gọn hơn
```



# Kiểu ký tự

Kiểu ký tự trong ngôn ngữ lập trình java có kích thước là 2 bytes và chỉ dùng để biểu diễn các ký tự trong bộ mã Unicode. Như vậy kiểu char trong java có thể biểu diễn tất cả  $2^{16} = 65536$  ký tự khác nhau.

Giá trị mặc định cho một biến kiểu char là null

Giá trị nhỏ nhất của một biến kiểu ký tự là 0 và giá trị lớn nhất là 65535

```
char a = 'u';  
char b = '5';  
char c = 65; //c == 'A'
```



# Kiểu luận lý

Kiểu boolean chỉ nhận 1 trong 2 giá trị: true hoặc false.

Trong java kiểu boolean không thể chuyển thành kiểu nguyên và ngược lại.

Giá trị mặc định của kiểu boolean là false

```
boolean a = true;  
boolean b = false;
```



## Kiểu reference



Kiểu reference (Kiểu dữ liệu tham chiếu) là kiểu dữ liệu của đối tượng.

Một số kiểu dữ liệu cụ thể như mảng (Array), lớp đối tượng (Class) hay kiểu giao tiếp (Interface), kiểu String, ...

# Ép kiểu



Ép kiểu là cách chuyển biến thuộc kiểu dữ liệu này thành biến thuộc kiểu dữ liệu khác  
Ý nghĩa:

- Việc chuyển kiểu dữ liệu sẽ đến lúc phải cần trong quá trình xử lý chương trình
- Có thể định dạng đúng kiểu dữ liệu mình mong muốn







## Các cách ép kiểu

Ép kiểu trong kiểu dữ liệu nguyên thủy được chia làm 2 loại:

01 Chuyển đổi kiểu ngầm định (implicit)

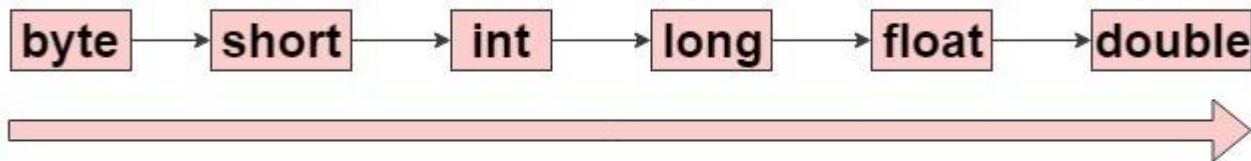
02 Chuyển đổi kiểu tường minh(explicit)



## Kiểu chuyển đổi ngầm định (**implicit**)

Việc chuyển đổi sẽ tự thực hiện bởi compiler và chúng ta không cần làm gì. Việc chuyển đổi này chỉ dành cho kiểu dữ liệu nhỏ sang kiểu dữ liệu lớn hơn. Ta có thể xem chiều từ nhỏ sang lớn như sau:

### Automatic Type Conversion (Widening - implicit)





## Kiểu chuyển đổi ngầm định (implicit)

Ví dụ :

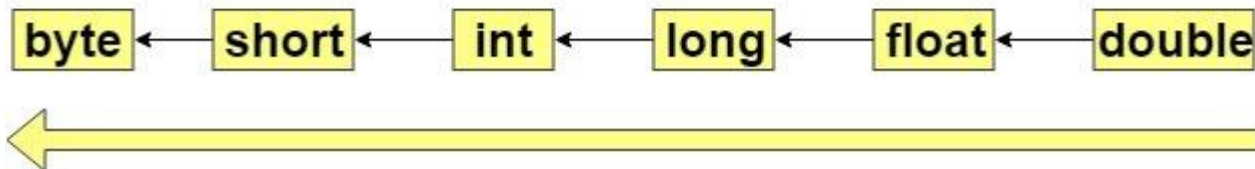
```
int a = 5;  
long b = a;  
System.out.print(b);
```



## Kiểu chuyển đổi tường minh (explicit)

Ngược lại với cách chuyển đổi ngầm định, việc chuyển đổi tường minh là chiều ngược lại từ kiểu dữ liệu lớn hơn sang kiểu dữ liệu nhỏ hơn (với điều kiện giá trị đó kiểu dữ liệu sẽ thay đổi có thể lưu trữ được trong kiểu dữ liệu mới).

### Narrowing (explicit)





## Kiểu chuyển đổi tường minh (explicit)

Ví dụ:

```
long a = 6;  
int b = (int) a;  
System.out.print(a);
```



# Kiểu enumerator

Enum là một từ khóa trong java, là một kiểu dữ liệu đặc biệt được sử dụng để đại diện cho hằng số cố định.

Một Enum có thể chứa các thuộc tính, phương thức và hàm tạo (Constructor)

Bởi vì các giá trị của enum là hằng số nên tên của các trường kiểu enum thường là các chữ cái in hoa.

```
enum Season{  
    SPRING, SUMMER, AUTUMN, WINTER;  
}
```

```
Season season = Season.WINTER;  
System.out.println(season);
```

WINTER



## Hằng số (Constant)

Hằng là một giá trị không đổi trong suốt chương trình, tất nhiên ta đã khởi tạo giá trị ngay từ ban đầu

Lý do sử dụng hằng:

- ☐ Tạo ra những giá trị không thay đổi , làm chương trình an toàn hơn
- ☐ Sử dụng với các giá trị như PI, gia tốc trọng trường,...
- ☐ Sẽ cảnh báo nếu người dùng cố tình thay đổi giá trị sau này. Đảm bảo tính nguyên vẹn của giá trị





## Một số hằng ký tự đặc biệt

Ký tự	Ý nghĩa
\b	Xoá lùi
\t	Tab
\n	Xuống hàng
\r	Dấu enter
\”	Nháy kép
\’	Nháy đơn
\\	Số ngược
\f	Đẩy trang
\uxxxx	Ký tự unicode



# Operators (Toán tử)





# Toán tử trong java

Giả sử ta có:

```
int a = 30;
```

```
int b = 10;
```

Toán tử	Ý nghĩa	Ví dụ
+	Cộng	$a + b = 40$
-	Trừ	$a - b = 20$
*	Nhân	$a * b = 300$
/	Chia nguyên	$a / b = 3$
%	Chia dư	$a \% b = 0$
++	Tăng 1	$a++ = 31$
--	Giảm 1	$b-- = 9$

## Toán tử trên bit

Giả sử ta có:

`int a = 30;` (00011110)

`int b = 10;` (00001010)

Toán tử	Ý nghĩa	Ví dụ
<code>&amp;</code>	AND	$a \& b = 10$ (00001010)
<code> </code>	OR	$a   b = 30$ (00010100)
<code>^</code>	XOR	$a ^ b = 20$ (00010100)
<code>&lt;&lt;</code>	Dịch trái	$a << 2 = 120$ (01111000)
<code>&gt;&gt;</code>	Dịch phải	$a >> 2 = 7$ (111)
<code>&gt;&gt;&gt;</code>	Dịch phải và điền 0 vào bit trống	$a >>> 2 = 7$ (00000111)
<code>~</code>	Bù bit	$\sim a = -31$

## Toán tử quan hệ

Giả sử ta có:

```
int a = 30;
```

```
int b = 10;
```

Toán tử	Ý nghĩa	Ví dụ
==	So sánh bằng	$a == b \Rightarrow \text{false}$
!=	So sánh khác	$a != b \Rightarrow \text{true}$
>	So sánh lớn hơn	$a > b \Rightarrow \text{true}$
<	So sánh nhỏ hơn	$a < b \Rightarrow \text{false}$
>=	So sánh lớn hơn hay bằng	$a \geq b \Rightarrow \text{true}$
<=	So sánh nhỏ hơn hay bằng	$a \leq b \Rightarrow \text{false}$

## Toán tử logic

Giả sử ta có:

```
boolean c = true;
```

```
boolean d = false;
```

Toán tử	Ý nghĩa	Ví dụ
&&	Toán tử và	$c \ \&\& \ d \Rightarrow \text{false}$
	Toán tử hoặc	$c \    \ d \Rightarrow \text{true}$
!	Toán tử phủ định	$!c \Rightarrow \text{false}$

## Toán tử gán

Toán tử	Miêu tả	Ví dụ
=	Toán tử gán đơn giản. Gán giá trị toán hạng bên phải cho toán hạng trái.	
+=	Thêm giá trị toán hạng phải tới toán hạng trái và gán giá trị đó cho toán hạng trái.	$c += a \Rightarrow c = c + a$
-=	Trừ đi giá trị toán hạng phải từ toán hạng trái và gán giá trị này cho toán hạng trái.	$c -= a \Rightarrow c = c - a$
*=	Nhân giá trị toán hạng phải với toán hạng trái và gán giá trị này cho toán hạng trái.	$c *= a \Rightarrow c = c * a$
/=	Chia toán hạng trái cho toán hạng phải và gán giá trị này cho toán hạng trái.	$c /= a \Rightarrow c = c / a$

## Toán tử gán

Toán tử	Miêu tả	Ví dụ
<code>%=</code>	Lấy phần dư của phép chia toán hạng trái cho toán hạng phải và gán cho toán hạng trái.	$c \% = a \Rightarrow c = c \% a$
<code>&lt;&lt;=</code>	Dịch trái toán hạng trái sang số vị trí là giá trị toán hạng phải.	$c << = a \Rightarrow c = c << a$
<code>&gt;&gt;=</code>	Dịch phải toán hạng trái sang số vị trí là giá trị toán hạng phải.	$c >> = a \Rightarrow c = c >> a$
<code>&amp;=</code>	Phép AND bit	$c \& = a \Rightarrow c = c \& a$
<code>^=</code>	Phép OR loại trừ bit	$c \wedge = a \Rightarrow c = c \wedge a$
<code> =</code>	Phép OR bit.	$c   = a \Rightarrow c = c   a$



## Toán tử điều kiện

**Cú pháp:** <điều kiện> ? <biểu thức 1> : <biểu thức 2>;

Nếu điều kiện đúng thì thực hiện <biểu thức 1>, còn ngược lại là <biểu thức 2>.

Trong đó:

- <điều kiện>: là một biểu thức logic
- <biểu thức 1>, <biểu thức 2>: có thể là hai giá trị, hai biểu thức hoặc hai hành động.

Ví dụ:

```
int a = 4;  
int b = 2;  
String s = (a%b==0) ? "a chia het cho b":"a khong chia het cho b";  
System.out.println(s);
```

Kết quả thực thi:

```
a chia het cho b
```

Thứ tự ưu tiên của các phép toán tính từ trái qua phải, từ trên xuống dưới

()	[]	.	
++	--	~	!
*	/	%	
+	-		
>>	>>>	<<	
>	>=	<	<=
==	!=		
&			
^			
&&			
?:			
=	<Toán tử>=		



# Math

## (Các phép toán)





# Lớp Math

Một số phương thức, hàm toán học của thư viện Math:

**Math.PI** hằng số pi:

```
double a = Math.PI * 5; //15.707963267948966
```

**Math.abs()** trả về giá trị tuyệt đối của tham số

```
int a = Math.abs(10); //10
```

**Math.ceil()** trả về giá trị double là số làm tròn tăng bằng giá trị số nguyên gần nhất

```
double a = Math.ceil(4.8); //5.0
```

**Math.floor()** trả về giá trị double là số làm tròn giảm

```
double a = Math.floor(4.3); //4.0
```

**Math.max()** trả về số lớn nhất trong hai số

```
int a = Math.max(4,6); //6
```

**Math.min()** trả về số nhỏ nhất trong hai số

```
int a = Math.min(4,6); //4
```

**Math.pow()** lấy lũy thừa (cơ số, số mũ)

```
double a = Math.pow(4,2); //16.0
```

**Math.sqrt()** khai căn

```
double a = Math.sqrt(9); //3.0
```

**Math.sin(), Math.cos()** sin và cos của đơn vị góc

```
double a = Math.sin(Math.PI/2); //1.0  
double b = Math.cos(Math.PI); // -1.0
```

**Math.random()** sinh số double ngẫu nhiên từ 0 đến 1

```
double a = Math.random();
```

**Math.toDegrees()** đổi góc radian thành độ

```
double a = Math.toDegrees(Math.PI); //180.0
```

**Math.toRadians()** đổi góc đơn vị độ ra radian

```
double b = Math.toRadians(45); //0.7853981633974483
```





## Exercise

Tính cạnh huyền của tam giác vuông khi biết:

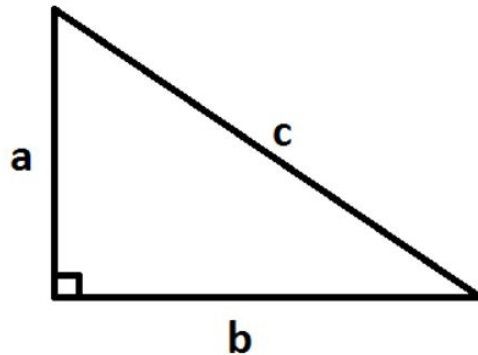
$$a = 3$$

$$b = 4$$

$$c = ?$$

Gợi ý:

- Sử dụng định lý Pythagore
- Sử dụng lớp Math



# Buổi 2: Java cơ bản 2



## Nội dung buổi học

- String (xâu, chuỗi)
- Date & Time
- Scanner
- Nhập xuất dữ liệu
- Thực hành

# String (Xâu, chuỗi)



# Kiểu xâu (String)

Trong java, String là một đối tượng biểu diễn một chuỗi các giá trị char

```
char[] ch = {'T','e','c','h','M','a','s','t','e','r'};  
String str = new String(ch);
```

```
String str = "TechMaster";
```



# Kiểu xâu (String)

Có hai cách để tạo đối tượng String:

01

Sử dụng String Literal

02

Sử dụng từ khóa new



# Sử dụng String Literal

String literal được tạo bằng cách sử dụng dấu nháy kép:

```
String s = "Ngoc Ban Quyen";
```

Các đối tượng String được lưu trữ trong một khu vực bộ nhớ đặc biệt được gọi là String Constant Pool

Sử dụng String literal giúp cho việc sử dụng bộ nhớ hiệu quả hơn vì nếu chuỗi đã tồn tại trong Pool thì sẽ không có đối tượng mới được tạo ra



## Sử dụng từ khóa new

```
String s = new String("TechMaster");
```

Trong trường hợp này, JVM sẽ tạo một đối tượng mới như bình thường trong bộ nhớ Heap (Không phải Pool) và hằng Techmaster sẽ được đặt trong Pool. Biến sẽ tham chiếu tới đối tượng trong Heap (Không phải Pool)





# Lớp String trong java

Lớp `java.lang.String` cung cấp rất nhiều phương thức để xử lý chuỗi. Các phương thức này giúp chúng ta thực hiện nhiều thao tác như cắt, ghép, chuyển đổi, so sánh, thay thế các chuỗi, ...



# Phương thức toUpperCase() và toLowerCase()

Phương thức toUpperCase() chuyển đổi chuỗi thành dạng chữ hoa và phương thức toLowerCase() chuyển đổi chuỗi thành dạng chữ thường.

```
String s="Hello Java";  
System.out.println(s.toUpperCase());//HELLO JAVA  
System.out.println(s.toLowerCase());//hello java  
System.out.println(s);//Hello Java
```



## Phương thức trim()

Phương thức trim() được sử dụng để xóa khoảng trắng ở đầu và cuối của chuỗi

```
String s = "  Java  ";  
System.out.println(s); // Java  
System.out.println(s.trim()); //Java
```



# Phương thức `length()`

Phương thức `length()` trả độ dài của chuỗi.

```
String s="Hello Java";  
System.out.println(s.length());//10
```



## Phương thức equals()

Phương thức equals() được sử dụng để so sánh nội dung của 2 chuỗi.

- ❑ **public boolean equals(Object another)**
- ❑ **public boolean equalsIgnoreCase(String another)**

```
String s1 = "Hello";  
String s2 = "HELLO";  
String s3 = "Hello";
```

```
System.out.println(s1.equals(s2));//false  
System.out.println(s1.equals(s3));//true  
System.out.println(s1.equalsIgnoreCase(s2));//true
```



# **Date & Time** **(Thời gian)**





## Date & Time

Java không có sẵn lớp Date nhưng ta có thể import *package java.time* để làm việc với ngày và giờ, gói này bao gồm nhiều lớp về Date&Time. Ví dụ:

Class	Mô tả
LocalDate	Đại diện cho ngày (year, month, day (yyyy-MM-dd))
LocalTime	Biểu thị cho giờ (hour, minute, second and nanoseconds (HH-mm-ss-ns))
LocalDateTime	Đại diện cho cả ngày và giờ (yyyy-MM-dd-HH-mm-ss-ns)
DateTimeFormatter	Định dạng để hiển thị và phân tích cú pháp các đối tượng ngày - giờ



# Hiển thị ngày hiện tại

Để lấy ngày giờ hiện tại, ta sử dụng lớp `java.time.LocalDate` và sử dụng phương thức `now()` của lớp đó

```
import java.time.LocalDate; // import lớp LocalDate

public class App {
    public static void main(String[] args) {
        LocalDate currentDate = LocalDate.now(); // Tạo đối tượng currentDate
        System.out.println(currentDate); // In ra màn hình ngày hôm nay
    }
}
```





# Hiển thị giờ hiện tại

Để lấy giờ hiện tại, cần thực hiện import lớp `java.time.LocalDateTime` và sử dụng phương thức `now()` của lớp này

```
import java.time.LocalDateTime; // import lớp LocalDateTime

public class App {
    public static void main(String[] args) {
        LocalDateTime currentTime = LocalDateTime.now(); //Tạo đối tượng currentTime
        System.out.println(currentTime); //In ra màn hình thời gian hiện tại
    }
}
```



# Hiển thị ngày giờ hiện tại

Để lấy ngày và giờ hiện tại, ta import lớp `java.time.LocalDateTime` và sử dụng phương thức `now()` của lớp đó

```
import java.time.LocalDateTime; // import lớp LocalDateTime

public class App {
    public static void main(String[] args) {
        LocalDateTime currentDateTime = LocalDateTime.now(); //Tạo đối tượng currentDateTime
        System.out.println(currentDateTime); //In ra ngày giờ hiện tại
    }
}
```



## Định dạng ngày và giờ

Bạn có thể sử dụng lớp `DateTimeFormatter` và phương thức `ofPattern()` trong cùng một package để định dạng hoặc phân tích cú pháp các đối tượng `date-time`.

```
import java.time.LocalDateTime; // Import lớp LocalDateTime
import java.time.format.DateTimeFormatter; // Import lớp DateTimeFormatter

public class App {
    public static void main(String[] args) {
        LocalDateTime myDateObj = LocalDateTime.now();
        System.out.println("Before formatting: " + myDateObj);
        DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss"
);

        String formattedDate = myDateObj.format(myFormatObj);
        System.out.println("After formatting: " + formattedDate);
    }
}
```

# Nhập xuất trong java



# Lớp Scanner

Lớp Scanner của package `java.util` được sử dụng để đọc dữ liệu đầu vào từ các nguồn khác nhau như người dùng nhập thông tin từ bàn phím, đọc file, ...





# Tạo đối tượng Scanner

Cú pháp:

```
Scanner <Tên biến tham chiếu> = new Scanner(Tham số truyền vào);
```

Ví dụ:

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

Lấy thông tin từ bàn phím

Tên biến tham chiếu

# Một số phương thức lớp Scanner

Phương thức	Mô tả
<b>public String next()</b>	Trả về kết quả nội dung trước khoảng trắng (String)
<b>public String nextLine()</b>	Trả về kết quả nội dung của một chuỗi nhập vào (String)
<b>public byte nextByte()</b>	Trả về kiểu dữ liệu byte
<b>public short nextShort()</b>	Trả về kiểu dữ liệu short
<b>public int nextInt()</b>	Trả về kiểu dữ liệu int
<b>public long nextLong()</b>	Trả về kiểu dữ liệu long
<b>public float nextFloat()</b>	Trả về kiểu dữ liệu float
<b>public double nextDouble()</b>	Trả về kiểu dữ liệu double





# Một số phương thức lớp Scanner

```
int n; // Khai báo biến n có kiểu dữ là int  
n = scanner.nextInt(); // Nhập dữ liệu kiểu số nguyên cho n từ bàn phím
```

```
String str; // khai bao str có kiểu dữ liệu là String  
str = scanner.nextLine(); // Nhập dữ liệu cho chuỗi str  
// hoặc  
str = scanner.next();
```

```
// Khai báo biến d có kiểu dữ liệu là double  
// nhập dữ liệu kiểu số thực cho d  
double d = scanner.nextDouble();
```



# Xuất dữ liệu

Trong Java có tới 3 cách in ra màn hình thì nên chọn cái nào trong trường hợp nào cho phù hợp.

- Với **Print**: Xuất kết quả ra màn hình nhưng con trỏ chuột không xuống dòng.
- Với **Println**: Xuất kết quả ra màn hình đồng thời con trỏ chuột nhảy xuống dòng tiếp theo.
- Với **Printf**: Xuất ra màn hình kết quả đồng thời có thể định dạng được kết quả đó nhờ vào các đối số thích hợp.

Hãy thử chạy những đoạn code này và xem kết quả:

Sử dụng print

```
System.out.print("Ơ mây zing, ");  
System.out.print("Gút chóp!!!");
```

Sử dụng println


```
System.out.println("Đưa tay đây nào ");  
System.out.println("mãi bên nhau bạn nhớ");
```

Sử dụng printf

```
System.out.printf("%s, %s", "Hải", " quay xe!");
```



# Bộ định dạng có sẵn trong Printf

- `%c` : Ký tự
- `%d` : Số thập phân (số nguyên) (cơ sở 10)
- `%e` : Dấu phẩy động theo cấp số nhân
- `%f` : Dấu phẩy động
- ~~`%i` : Số nguyên (cơ sở 10)~~ 
- `%o` : Số bát phân (cơ sở 8)
- `%s` : Chuỗi
- ~~`%u` : Số thập phân (số nguyên) không dấu~~
- `%x` : Số trong hệ thập lục phân (cơ sở 16)
- `%%` : Dấu phần trăm
- `\%` : Dấu phần trăm

```
int i = 5;  
System.out.printf("Số nguyên: %d\n", i);
```

```
float f = 6.8f;  
System.out.printf("Số thực: %f\n", f);
```

```
String str = "Java";  
System.out.printf("Chuỗi: %s\n", str);
```

```
char c = 'a';  
System.out.printf("Ký tự: %c\n", c);
```

```
Số nguyên: 5  
Số thực: 6.800000  
Chuỗi: Java  
Ký tự: a
```

```
int i = 5;  
int j = 7;  
System.out.printf("Tổng của %d và %d là: %d\n ", i, j, i+i);
```

Tổng của 5 và 7 là: 10

```
double d = 3.14159265;  
System.out.printf("PI = %.2f", d);
```

PI = 3.14

```
Date date = new Date(0);  
System.out.println("In ngày với println: "+date);  
System.out.printf("Lấy giờ với printf: %tT\n",date);  
System.out.printf("Lấy ngày với printf: %td/%tm/%ty\n", date, date, date);  
System.out.printf("Lấy giờ với printf: %tH:%tM", date, date);
```

```
In ngày với println: 1970-01-01  
Lấy giờ với printf: 07:00:00  
Lấy ngày với printf: 01/01/70  
Lấy giờ với printf: 07:00
```



## Exercise

Viết chương trình tính chỉ số BMI, với cân nặng và chiều cao nhập từ bàn phím. Trong đó:

- Cân nặng tính theo kg
- Chiều cao tính theo met

In ra màn hình chỉ số BMI



# Buổi 3: Java cơ bản 3



## Nội dung buổi học

- Câu lệnh rẽ nhánh (if else)
- Vòng lặp for
- Vòng lặp while
- Thực hành

# Branching

## (Câu lệnh rẽ nhánh)

## Câu lệnh rẽ nhánh

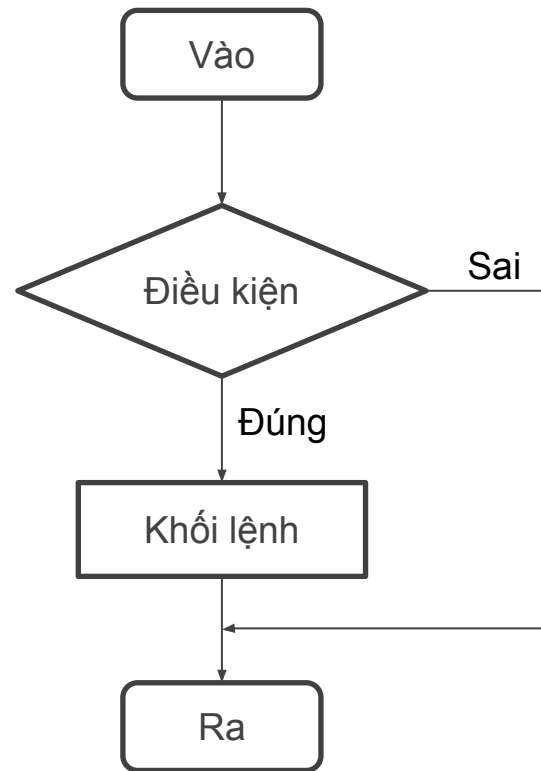


# Câu lệnh if

Dạng 1:

```
if(<Điều kiện>{  
    <Khối lệnh>;  
}
```

*“Nếu trời mưa thì tôi ở nhà”*



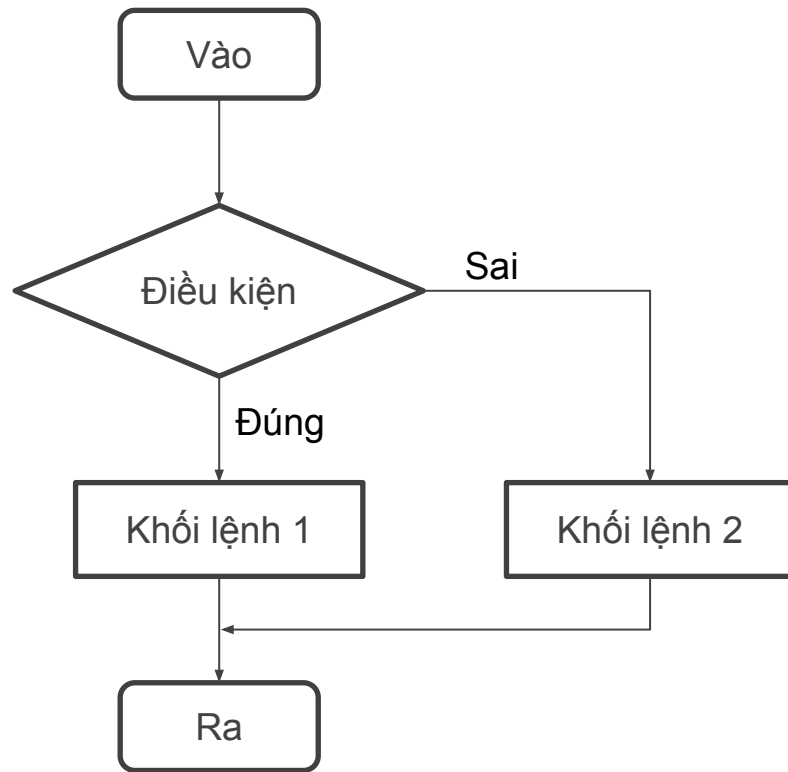
```
int i = 10;  
if(i%2==0){  
    System.out.println(i + " là số chẵn");  
}
```

```
String s1 = "Java";  
String s2 = "Java";  
if(s1.equals(s2)){  
    System.out.println("Hai chuỗi giống nhau");  
}
```

# Câu lệnh if else

Dạng 2:

```
if(<Điều kiện>){  
    <Khối lệnh 1>;  
}else{  
    <Khối lệnh 2>;  
}
```



```
int i = 10;  
if(i%2==0){  
    System.out.println(i + " là số chẵn");  
}else{  
    System.out.println(i + " là số lẻ");  
}
```

```
String s1 = "Java";  
String s2 = "HTML";  
if(s1.equals(s2)){  
    System.out.println("Hai chuỗi giống nhau");  
}else{  
    System.out.println("Hai chuỗi khác nhau");  
}
```





## Lồng các lệnh if else

Ta hoàn toàn có thể thực hiện lồng các câu lệnh if else, nghĩa là sử dụng một lệnh if hoặc else if bên trong lệnh if hoặc else if khác.

Ví dụ:

```
int a = 4;  
int b = 2;  
int c = 6;  
if(a > b){  
    if(b > c){  
        System.out.println("a lớn hơn c");  
    }  
}
```



## Lệnh if ... else if ... else

Một lệnh if có thể được theo sau bởi else if ... else tùy ý, nó rất hữu ích để kiểm tra các điều kiện đa dạng bởi sử dụng lệnh if ... else if đơn

Ví dụ:

```
int x = 30;
if( x == 10 ){
    System.out.print("Giá trị của x là 10");
}else if( x == 20 ){
    System.out.print("Giá trị của x là 20");
}else if( x == 30 ){
    System.out.print("Giá trị của x là 30");
}else{
    System.out.print("Không phải giá trị của x");
}
```



## Exercise

Viết chương trình tính chỉ số BMI, với cân nặng và chiều cao nhập từ bàn phím.

Trong đó:

Cân nặng tính theo kg

Chiều cao tính theo met

In ra màn hình chỉ số BMI và thông báo kết quả nếu:

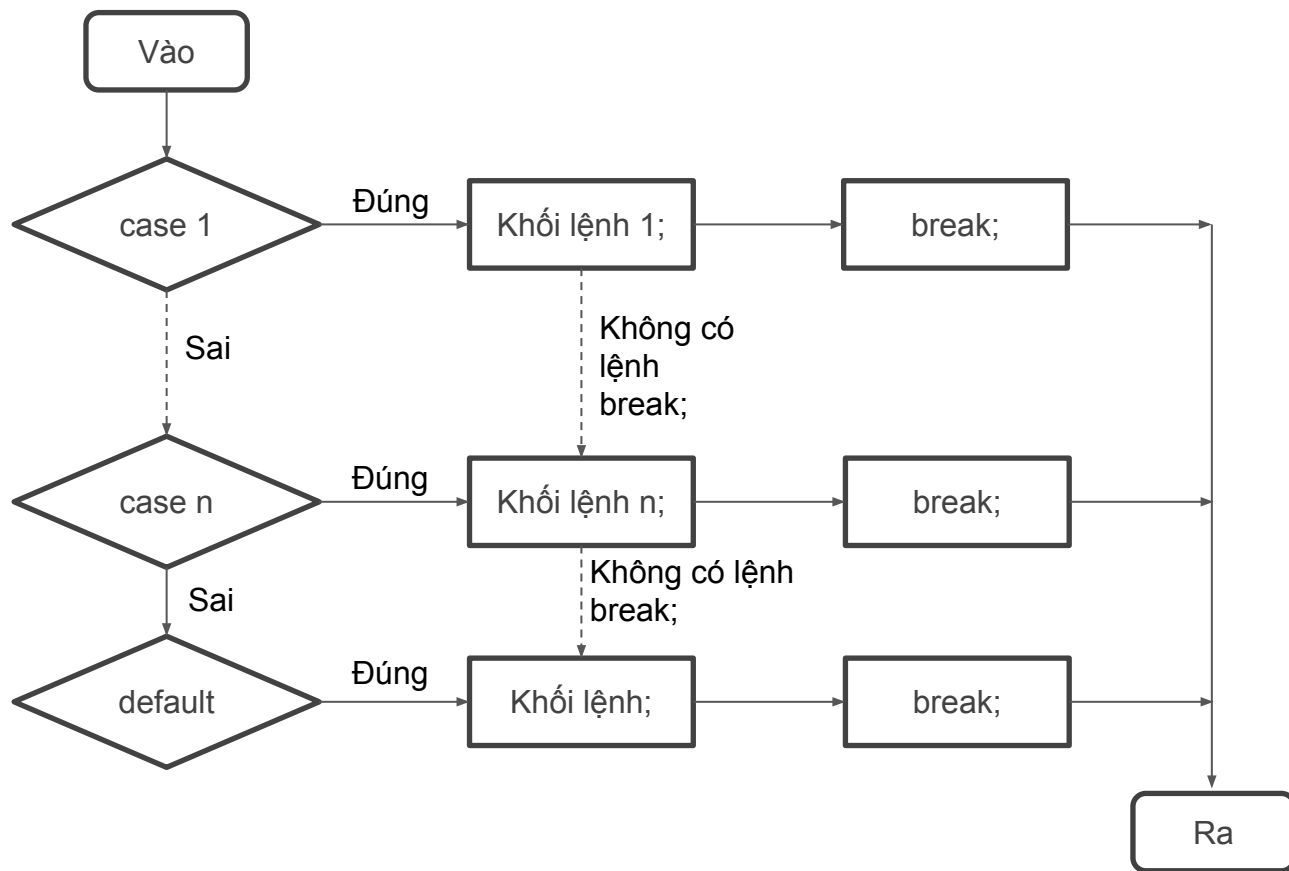
- $BMI < 18,5$ : Thiếu cân
- $18,5 \leq BMI \leq 24,9$ : Cân đối
- $BMI > 24,9$ : Thừa cân



# switch ... case

Cấu trúc lệnh switch...case

```
switch (<Biến>) {  
    case <Giá trị 1>:  
        <Khởi lệnh 1>;  
        break;  
    case <Giá trị 2>:  
        <Khởi lệnh 2>;  
        break;  
    ...  
    case <Giá trị n>:  
        <Khởi lệnh n>;  
        break;  
  
    default:  
        <Khởi lệnh>;  
        break;  
}
```



Ví dụ:

Hãy thử bỏ break  
tại đây và xem kết quả

```
int number = 2;
switch (number) {
case 1:
    System.out.println("One");
    break;
case 2:
    System.out.println("Two");
    break;
case 3:
    System.out.println("Three");
    break;
case 4:
    System.out.println("Four");
    break;
default:
    System.out.println("Không có số này");
    break;
```



## Exercise

Nhập vào một tháng bất kỳ và in ra màn hình thông tin tháng đó có bao nhiêu ngày.

Biết:

Tháng 1, 3, 5, 7, 8, 10, 12 có 31 ngày

Tháng 2 có 28 hoặc 29 ngày

Tháng 4, 6, 9, 11 có 30 ngày

# Loops (Vòng lặp)

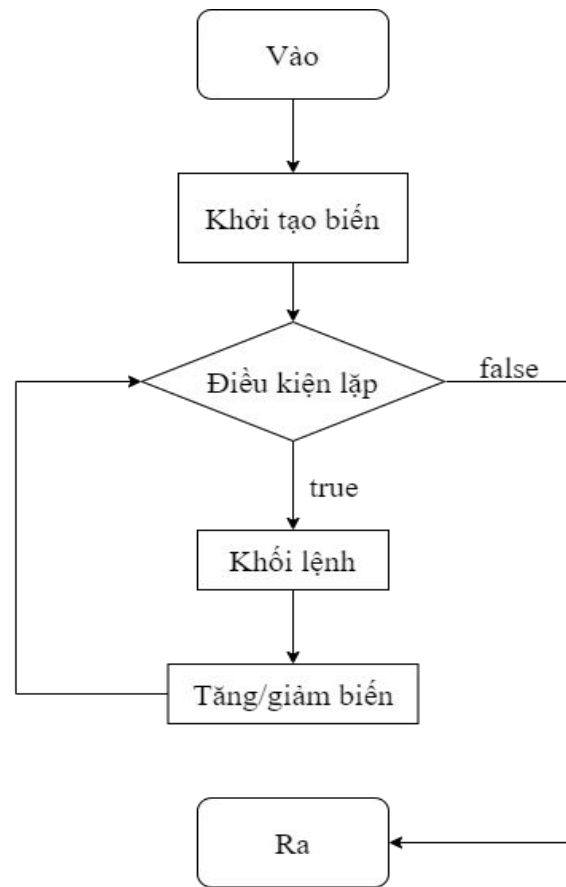




# Vòng lặp for

Cú pháp:

```
for (<Khởi tạo biến>; <Điều kiện>; <Tăng/giảm biến>){  
    <Khối lệnh>;  
}
```



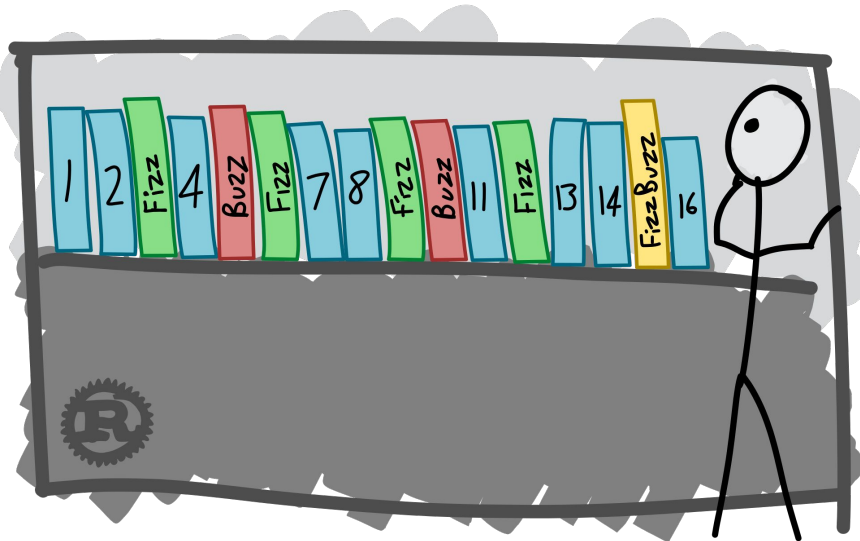
```
for(int i = 0; i < 5; i++){  
    System.out.println("À thế à...");  
}
```

Kết quả thực thi:

```
À thế à...  
À thế à...  
À thế à...  
À thế à...  
À thế à...
```

## Exercise

Viết một chương trình để in ra các số từ 1 đến 100. Trong đó những số nào chia hết cho 3 thì in chữ "Fizz", những số chia hết cho 5 thì in chữ "Buzz", còn những số chia hết cho cả 3 và 5 thì in chữ "FizzBuzz"

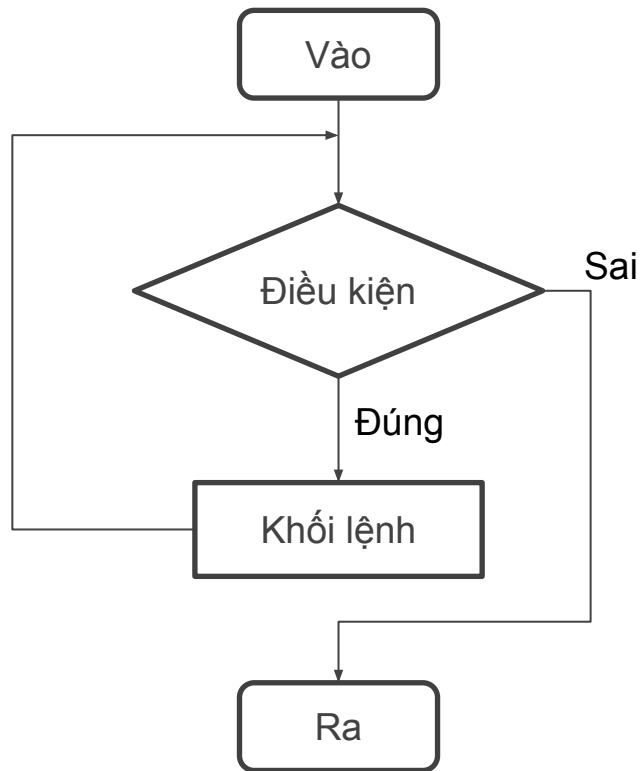




# Vòng lặp while

Cú pháp:

```
while (<Điều kiện lặp>) {  
    <Khối lệnh>;  
}
```



```
int i = 1;  
while (i <= 5) {  
    System.out.print(i + "\t");  
    i++;  
}
```

1    2    3    4    5



## Exercise

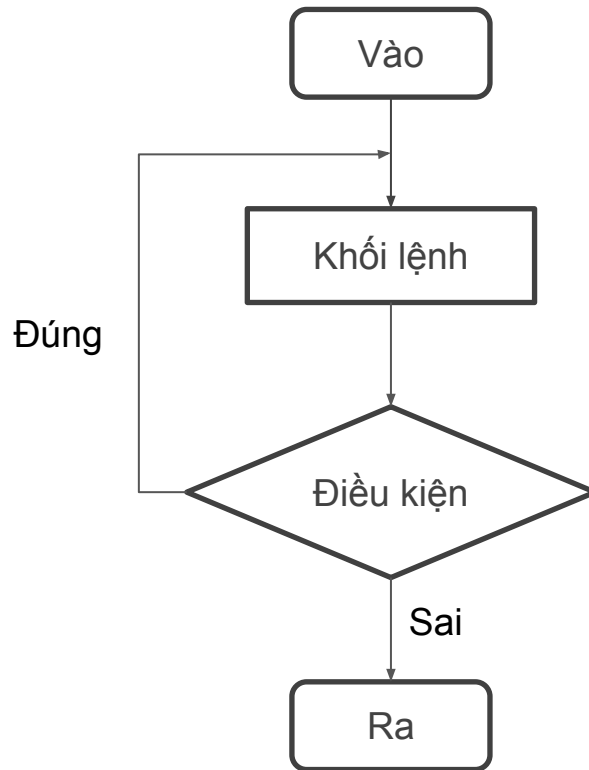
- 1, Viết chương trình cho phép nhập vào một số nguyên dương  $n$ , tính tổng tất cả số chẵn trong khoảng từ  $0 - n$ .
- 2, Viết chương trình liệt kê  $n$  số nguyên tố đầu tiên.



# Vòng lặp do while

Cú pháp:

```
do {  
    <Khối lệnh>;  
}while (condition);
```



```
int i = 1;  
do{  
    System.out.print(i + "\t");  
    i++;  
}while(i <= 5);
```

1    2    3    4    5





## Từ khóa break

Từ khóa break được sử dụng để dừng thực thi vòng lặp hoặc trong mệnh đề switch tại điều kiện đã được chỉ định. Đối với vòng lặp bên trong vòng lặp khác, nó chỉ stop vòng lặp bên trong đó

```
for(int i = 0; i < 10; i++){  
    if(i == 5)  
        break;  
    System.out.print(i+"\t");  
}
```

0 1 2 3 4



## Từ khóa continue

Từ khóa continue được sử dụng để tiếp tục vòng lặp tại điều kiện đã được xác định. Với điều kiện đó, khối lệnh phía sau từ khóa continue sẽ không được thực thi

Đối với vòng lặp bên trong vòng lặp khác, continue chỉ có tác dụng với vòng lặp bên trong đó

```
for (int i = 2; i < 10; i++){  
    if(i%2 == 0){  
        continue;  
    }  
    System.out.println(i);  
}
```

3 5 7 9



# Exercise

Viết chương trình thực hiện:

- 1 - Sinh một số nguyên ngẫu nhiên rdNumber
  - 2 - Nhập vào bàn phím số nguyên bất kỳ number
- Nếu `number > rdNumber` thì thông báo bạn đoán lớn hơn rồi, và cho nhập lại
  - Nếu `number < rdNumber` thì thông báo bạn đoán nhỏ hơn rồi và cho nhập lại
  - Nếu `number == rdNumber` thì thông báo bạn đoán trúng rồi và kết thúc chương trình

# Buổi 4: Java cơ bản 4



# Nội dung buổi học

- Mảng một chiều
- Mảng hai chiều
- Duyệt mảng
- Phương thức
- Phạm vi của biến
- Tiêu chuẩn coding
- Thực hành



# Array

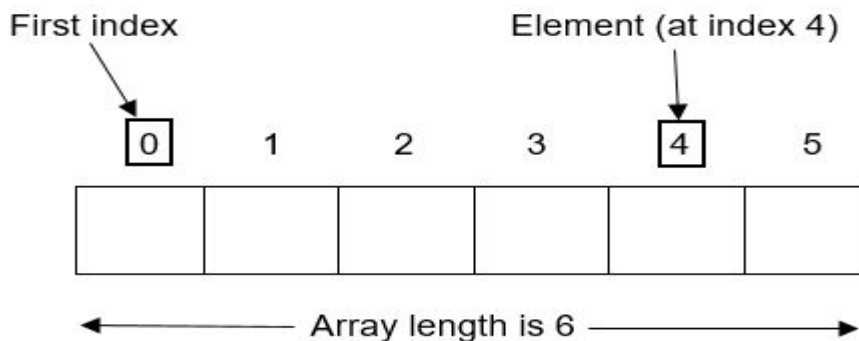




# Mảng (Array)

Mảng là tập hợp nhiều phần tử có cùng tên, cùng kiểu dữ liệu và mỗi phần tử trong mảng được truy xuất thông qua các chỉ số của nó trong mảng

Mảng trong java lưu các phần tử theo chỉ số, chỉ số của phần tử đầu tiên là 0





# Mảng (Array)

Mảng có hai loại:



Mảng một chiều



Mảng đa chiều





# Mảng một chiều

Khai báo mảng:

```
<Kiểu dữ liệu> <Tên mảng>[];
```

Hoặc:

```
<Kiểu dữ liệu>[] <Tên mảng>;
```

Cú pháp cấp phát bộ nhớ cho mảng:

```
<Tên mảng> = new <kiểu dữ liệu>[<Kích thước mảng>];
```

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

Khai báo mảng a có kiểu dữ liệu là int

Mảng a có kích thước là 3

Gán giá trị cho mảng a

Lấy kích thước mảng

Mang a:

5  
2  
1



# Mảng đa chiều

Mảng đa chiều chỉ là tăng số chiều lưu trữ nhiều chiều hơn, hay còn gọi là ma trận. Thông thường ta hay sử dụng mảng 2 chiều.

Đối với mảng 2 chiều, dữ liệu được lưu trữ theo hai chiều.

Chiều thứ nhất gọi là hàng và chiều thứ hai gọi là cột.

Trong nội dung này chúng ta chủ yếu tìm hiểu về mảng 2 chiều



# Mảng đa chiều

	Column 1	Column 2	Column 3	Column 4
Row1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Cột

Hàng



# Mảng đa chiều

Khai báo mảng:

```
<Kiểu dữ liệu> <Tên mảng>[ ][ ];
```

Hoặc:

```
<Kiểu dữ liệu>[ ][ ] <Tên mảng>;
```

Cú pháp cấp phát bộ nhớ cho mảng:

```
<Tên mảng> = new <kiểu dữ liệu>[<Số dòng>][<Số cột>];
```

```
int[][] a= {{1,2,3},{4,5,6},{7,8,9}};  
for (int i=0; i<3; i++){  
    for(int j=0; j<3; j++){  
        System.out.print(a[i][j]+" ");  
    }  
    System.out.println();  
}
```

Sử dụng hai vòng lặp để  
in thông tin mảng đa chiều

```
1 2 3  
4 5 6  
7 8 9
```



## For each

Vòng lặp `for each` chủ yếu được sử dụng để duyệt mảng hoặc các phần tử của `collection`.

Với `for each`, thay vì khai báo hay khởi tạo biến lặp vị trí, chúng ta sẽ khai báo một biến chung kiểu dữ liệu của mảng, sử dụng biến đó để duyệt các phần tử của mảng mà không cần lấy vị trí (`index`) của mỗi phần tử



# For each

Cú pháp:

```
for(<Kiểu dữ liệu> <Tên biến chạy>: <Tên mảng>){  
    <Khối lệnh lặp lại>;  
}
```



```
int[] a;  
a = new int[3];  
a[0] = 5;  
a[1] = 2;  
a[2] = 1; Tên biến chạy  
  
System.out.println("Mang a: ");  
for (int i: a) {  
    System.out.println(i);  
}
```

```
5  
2  
1
```

# Phương thức (Method)



# Định nghĩa

Phương thức là một đoạn code chỉ được chạy khi nó được gọi.

Lợi ích:

- + Cấu trúc/tổ chức code tốt hơn
- + Tái sử dụng lại code

Cú pháp khai báo:

```
<Modifier> <Static/non-static> <Kiểu dữ liệu trả về> <Tên Phương thức>(<Kiểu dữ liệu> <tên tham số>) {  
  
    // Khối lệnh  
  
}
```

Không trả về gì

```
public class Main4 {  
    static void myMethod() {  
        System.out.println("Hello java method");  
    }  
  
    public static void main(String[] args) {  
        System.out.println("Hello world");  
        // gọi method  
        myMethod(); // "Hello java method"  
        myMethod(); // "Hello java method"  
    }  
}
```

Tên phương thức

phương thức tĩnh

Gọi / sử dụng phương thức

```
Hello world  
Hello java method  
Hello java method
```

Trả về kiểu số nguyên

2 tham số, kiểu số nguyên

```
static int calculateSum(int x, int y) {  
    System.out.println("tính tổng số nguyên");  
    return x + y;  
}
```

Trả về kiểu số nguyên

```
public static void main(String[] args) {  
    System.out.println("Hello world");  
  
    int sum = calculateSum(3, 6);  
    System.out.println(sum);  
}
```

Gọi / sử dụng phương thức

```
Hello world  
tính tổng số nguyên  
9
```

```
static int calculateSum(int x, int y) {  
    System.out.println("tính tổng số nguyên");  
    return x + y;  
}  
  
static double calculateSum(double x, double y) {  
    System.out.println("tính tổng số thập phân");  
    return x + y;  
}  
  
public static void main(String[] args) {  
    System.out.println("Hello world");  
    System.out.println(calculateSum(1, 2));  
    System.out.println(calculateSum(1.0, 2.0));  
}
```

Hai phương thức, nhưng  
tham số khác nhau

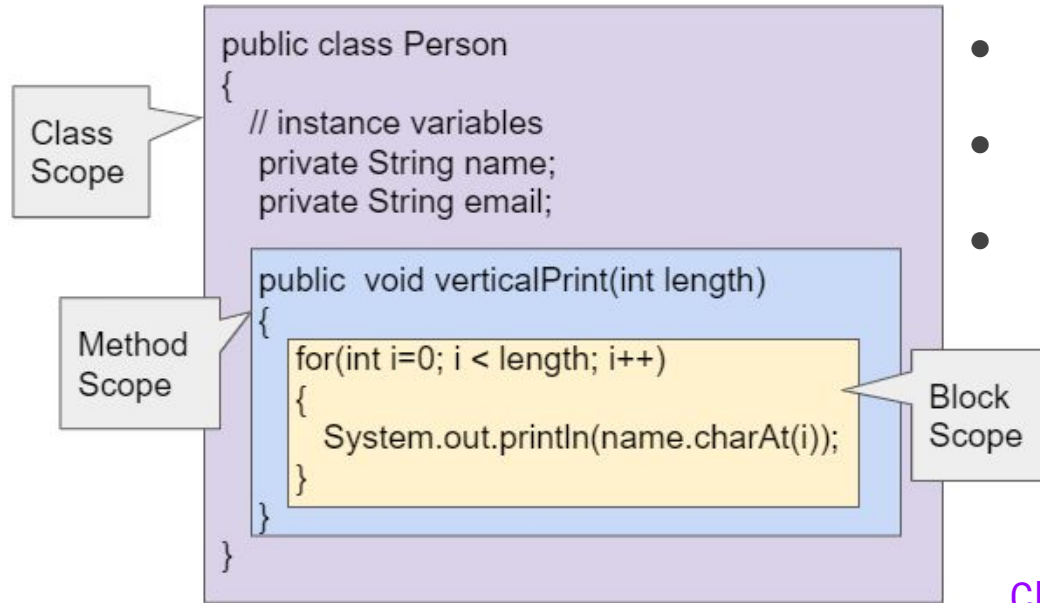
Gọi phương thức 2 lần với  
2 tham số khác nhau

//tính tổng số nguyên 3  
//tính tổng số thập phân 3.0

```
...  
3  
3.0
```

# Phạm vi của biển

# Các loại scope



- Biến khởi tạo trong scope nào thì được dùng trong scope đó
- Chỉ sử dụng được biến sau khi đã khai báo
- Biến trong scope lớn hơn có thể có thể được dùng trong scope nhỏ hơn
- Sau khi chạy hết khối lệnh trong method và block scope thì biến sẽ bị huỷ

Chú ý: Biến khai báo trong Class Scope có thể được truy xuất ở ngoài class thông qua access modifiers (Học trong bài OPP)



# Block scope

```
public static void main(String[] args) {  
    for (int i = 0; i < 5; i++) {  
        //không thể sử dụng var ở đây  
        //System.out.print(var + i); //-> Lỗi  
  
        int var = 3;  
        // bắt đầu sử dụng được biến var  
        System.out.println(var + i);  
    }  
    // ra khỏi khối {} nên không sử dụng được var nữa  
    // System.out.println(var); //-> Lỗi  
}
```

# Method scope

```
public static void main(String[] args) {  
    int methodVar = 10;  
  
    for (int i = 0; i < 5; i++) {  
        //không thể sử dụng var ở đây  
        //System.out.print(var + i); //-> Lỗi  
  
        int var = 3;  
        // bắt đầu sử dụng được biến var  
        System.out.println(var + i);  
        System.out.println(methodVar + i); // sử dụng được biến methodVar  
    }  
    // ra khỏi khối {} nên không sử dụng được var nữa  
    // System.out.println(var); //-> Lỗi  
  
    // sử dụng được biến methodVar  
    System.out.println(methodVar);  
}
```



# Clean code



# Khái niệm Clean code

Clean code là thuật ngữ chỉ đến những mã nguồn tốt, nó có các đặc điểm:



## Nguyên tắc của hướng đạo sinh



*Leave the campground cleaner than you found it.*

*Sau khi cắm trại, phải sạch hơn lúc bạn mới đến*

Liên tục cải thiện code, làm cho code của dự án tốt dần theo thời gian chính là một phần quan trọng của sự chuyên nghiệp



# SMELL CODE



# Tại sao phải clean code?

Teamwork dễ dàng hơn

Debug dễ hơn

Ít rủi ro hơn

Năng suất hơn



**Tại sao phải clean code?**

**Hạnh phúc  
hơn**





## Làm gì để clean code?

- Đặt tên thế nào cho tốt?
- Viết hàm thế nào để tốt, ngắn gọn?
- Cách ghi chú thế nào?
- Xử lý lỗi, bắt lỗi thế nào cho hợp lệ?
- Format ra sao?
- Thiết kế liên quan tới đối tượng, lớp?



# Đặt tên có ý nghĩa và đọc được

- Không tốt:

```
String ddmmyyyy = fDateFormat.format(date);
```

- Tốt:

```
String currentDate = fDateFormat.format(date);
```



# Đặt tên đồng nhất khi có cùng ý nghĩa

- Không tốt

```
getUserInfo();  
getClientData();  
getCustomerRecord();
```

- Tốt

```
getUser();
```



# Tên dễ tìm kiếm. Tránh magic number

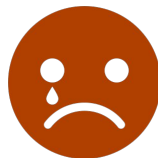
- Không tốt

```
case 2:  
    System.out.printf("Thang %d co 28 hoac 29 ngay", month);  
    break;
```

- Tốt

```
private static final int FEBRUARY = 2;  
case FEBRUARY:  
    System.out.printf("Thang %d co 28 hoac 29 ngay", month);  
    break;
```

```
final int maxcount = 1;  
boolean change = true;  
interface Repository;  
public class personaddress;  
void getallorders();
```



```
final int MAXCOUNT = 1; //constant, viết hoa tất.  
boolean isChanged = true ; //tiền tố is đây là boolean  
interface Irepository //tiền tố I đây là Interface  
public class PersonAddress //Camel case phân biệt từ dễ học  
void getAllOrders() //Camel case nhưng chữ cái đầu viết thường
```



# Không thêm các ngữ cảnh thừa

- Không tốt

```
public class Person {  
  
    public String personName;  
    public int personAge;  
    public String personAddress;  
}
```

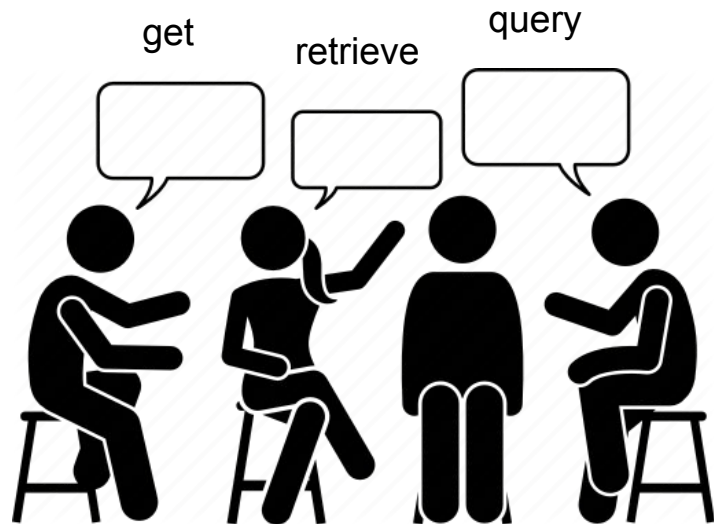
- Tốt

```
public class Person {  
  
    public String name;  
    public int age;  
    public String address;  
}
```



# Thống nhất tên gọi

```
List<Student> getAllStudents();  
List<Student> retrieveAllStudents();  
List<Student> findAllStudents();  
List<Student> queryAllStudents();  
List<Student> getListOfStudents();
```



- Chọn từ nào bình dân, dễ nhớ, khó viết sai chính tả !
- Dùng nó ở mọi nơi.
- Nếu bạn thấy đồng nghiệp sử dụng một từ đồng nghĩa, hay thảo luận để refactor để cùng thống nhất dùng một từ

```
public class Person
```

```
{  
    String addressCity;  
    String addressHomeNumber;  
    String addressPostCode;  
}
```



Vi phạm tính đóng gói và khó bảo trì

```
public class Address {
```

```
    String addressCity;  
    String addressHomeNumber;  
    String addressPostCode;  
}
```

```
public class Person
```

```
{  
    List<Address> addresses;  
}
```



Đóng gói tốt, đúng phạm vi (context)

Có thể tùy biến 1 address hoặc nhiều address !





## Exercise

1 Viết chương trình nhập vào 2 ma trận vuông A và B , in mảng đó ra màn hình.

- Thực hiện cộng 2 ma trận
- Tìm ma trận chuyển vị của 2 ma trận đó

2 Viết chương trình nhập vào 1 mảng đa chiều, in mảng đó ra màn hình.  
Tính tổng các phần tử chia hết cho 5 trong mảng đó



# Exercise

Viết chương trình tạo mảng số nguyên gồm  $n$  phần tử ( $n$  nhập vào từ bàn phím), thực hiện:

- Nhập phần tử cho mảng và in mảng ra màn hình
- Hiển thị phần tử tại vị trí  $\text{index} = 2$
- Tính tổng các phần tử trong mảng
- In ra màn hình các số chẵn và tổng các số đó
- Sắp xếp mảng theo thứ tự tăng dần
- Sắp xếp mảng theo thứ tự giảm dần



## Exercise

Tạo mảng chuỗi gồm  $n$  phần tử ( $n$  nhập vào từ bàn phím)

- Nhập thông tin cho các phần tử
- In thông tin ra màn hình
- Đếm số lần “Java” xuất hiện trong mảng
- Nhập vào từ bàn phím chuỗi bất kỳ, kiểm tra chuỗi đó ở vị trí nào của mảng