

JAVATM

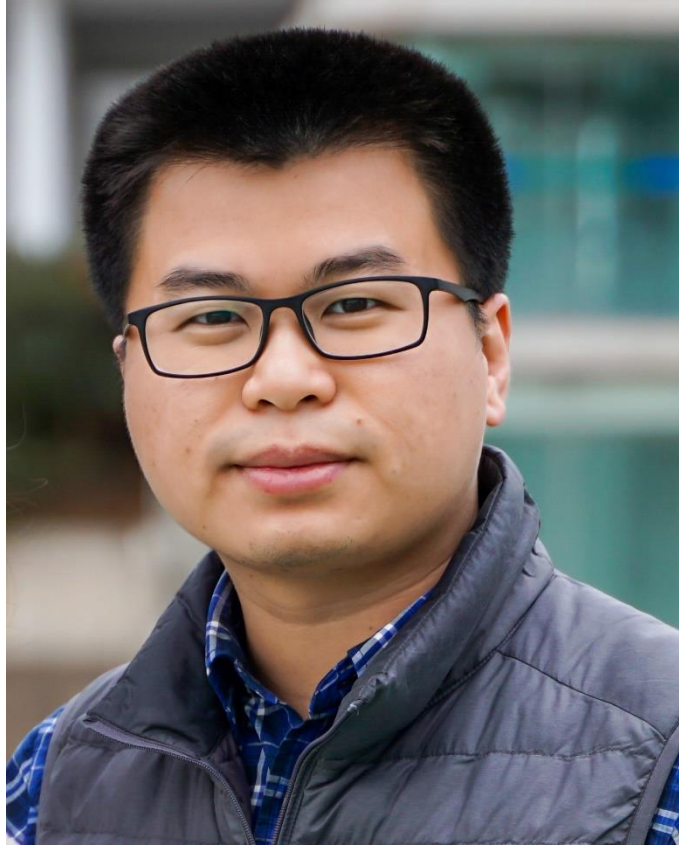


JAVA CƠ BẢN – CẤU TRÚC LẬP – KIỂU KÝ TỰ - CHUỖI – ĐỊNH DẠNG HIỂN THỊ

GIẢNG VIÊN

TS. Hà Ngọc Long

THÔNG TIN GIẢNG VIÊN



TS. Hà Ngọc Long

Email: hnlng@hueuni.edu.vn

- Thông tin Giảng viên
 - Tiến sĩ ngành Hệ thống Thông tin và Truyền thông, Hàn Quốc.
 - Thạc sĩ ngành Kỹ thuật Quản trị Công nghiệp, Hàn Quốc.
 - Cử nhân ngành Hệ thống Thông tin Quản lý, Đại học Kinh tế Huế
- Hướng nghiên cứu:
 - Quản trị quy trình nghiệp vụ (Business Process Management).
 - Chuyển đổi số.
 - FinTech, RegTech
 - Tiền điện tử, hợp đồng thông minh, và Blockchain.
 - Khai phá quy trình nghiệp vụ (Process Mining)

MỤC TIÊU CỦA HỌC PHẦN

“Giới thiệu về ngôn ngữ lập trình hướng đối tượng Java. Tập trung giới thiệu cho sinh viên kỹ năng, kỹ thuật lập trình theo phương pháp hướng đối tượng”

TIÊU CHUẨN ĐÁNH GIÁ SINH VIÊN

Hình thức Đánh giá	% Điểm
Chuyên cần	5%
Phát Biểu & Thảo Luận	5%
Bài Tập Về Nhà	5%
Thực Hành & Thực Tế	15%
Kiểm tra giữa kỳ	15%
Bài tập cá nhân	10%
<i>Thi kết thúc học phần</i>	<i>45%</i>
Tổng cộng:	100%

NỘI DUNG MÔN HỌC

1

- Java cơ bản

2

- Nhập môn lập trình HĐT

3

- Lập trình HĐT với Java

4

- Lập trình đồ họa và sự kiện trong Java

5

- Cấu trúc dữ liệu nâng cao trong Java

TÀI LIỆU HỌC TẬP

- *Y. Daniel Liang* (2019) Introduction to Java Programming and Data Structures - Comprehensive Version (12th edition), *Pearson*
- *C Thomas Wu.* (2004). An introduction to object oriented programming with Java. *McGraw-Hill – Boston.*

Tài liệu hướng dẫn và phần mềm thực hành:

- Apache Netbeans IDE 12 (or higher)
- Java SE 17 (or higher)
- Videos bài giảng và hướng dẫn thực hành



Chapter 4-5 Mathematical Functions, Characters, and Strings. Loops

Cấu trúc lặp - Loops

Motivations

- Giả sử chúng ta muốn in ra màn hình một chuỗi String (vd: “Welcom to Java!”) 100 lần.
- Liệu có khôn ngoan nếu chúng ta viết câu lệnh sau 100 lần?

```
System.out.println("Welcome to Java!");
```

- Vậy, làm sao chúng ta giải quyết vấn đề này?

Opening Problem

- Vấn đề được đặt ra:

100 lần

```
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");
```

...

...

...

```
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");
```

Introducing while Loops

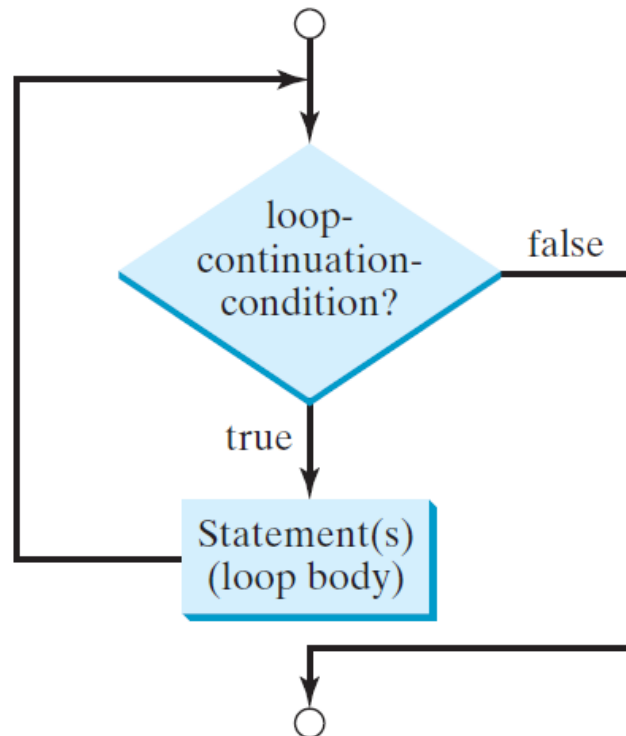
```
int count = 0;
while (count < 100) {
    System.out.println("Welcome to Java");
    count++;
}
```

Objectives

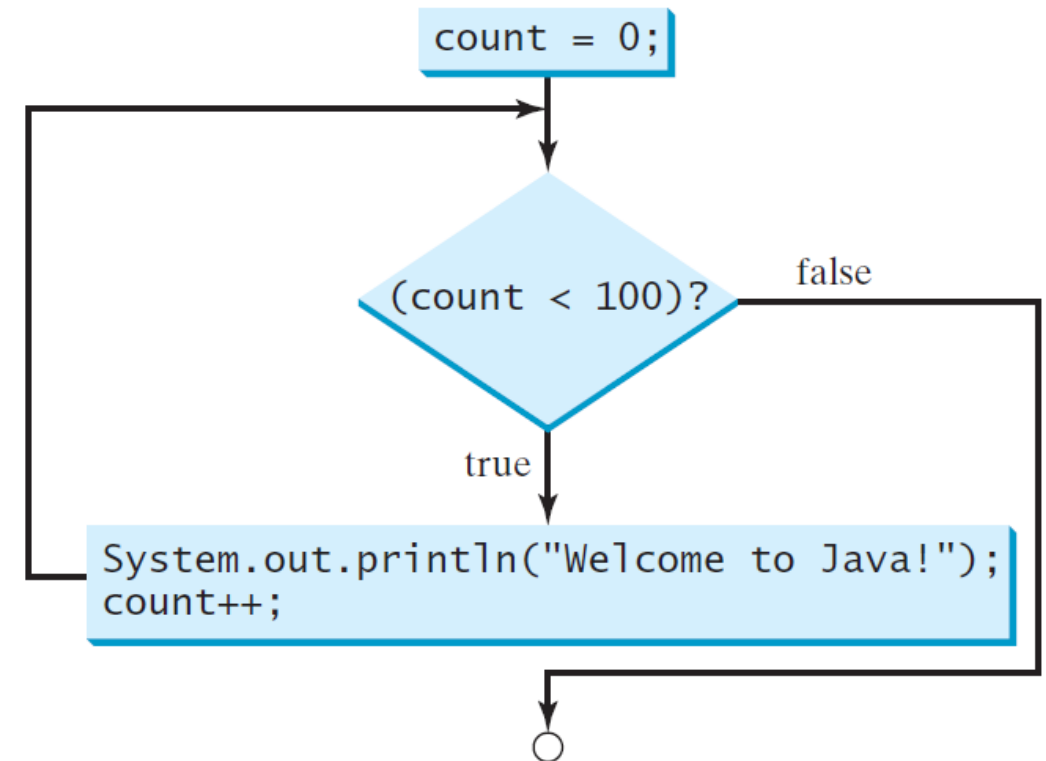
- Viết chương trình để thực thi câu lệnh một cách liên tiếp sử dụng lệnh lặp **while** (§5.2).
- Sử dụng phương pháp thiết kế vòng lặp (§§5.2.1–5.2.3).
- Kiểm soát vòng lặp với biến điều khiển (§5.2.4).
- Nhập giá trị đầu vào từ file thay vì dùng bàn phím (§5.2.5).
- To write loops using **do-while** statements (§5.3).
- Sử dụng vòng lặp **for** (§5.4).
- Phân biệt sự giống và khác nhau giữa các loại vòng lặp (§5.5).
- Viết chương trình với các vòng lặp lồng nhau (§5.6).
- Nắm bắt những kỹ thuật để giảm thiểu những lỗi số học (§5.7).
- Viết chương trình sử dụng lệnh **break** và **continue** (§5.9).

while Loop Flow Chart

```
while (điều-kiện-tiếp-tục-lặp) {  
    // loop-body;  
    Statement(s);  
}
```



```
int count = 0;  
while (count < 100) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```



Trace while Loop

Khởi tạo biến đếm

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

Trace while Loop

(count < 2) bằng true

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

Trace while Loop

In ra màn hình: Welcome to Java

```
int count = 0;  
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```


Trace while Loop

Tăng biến đếm count lên 1 đơn vị

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```

Trace while Loop

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

(count < 2) vẫn bằng true vì count = 1

Trace while Loop

In ra màn hình Welcome to Java

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```

Trace while Loop

Tăng biến đếm count lên 1 đơn vị

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```

Trace while Loop

(count < 2) bằng false vì count = 2

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

Trace while Loop

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```

Thoát khỏi vòng lặp. Thực thi các câu lệnh tiếp theo, sau câu lệnh lặp

Exercise 1: Guessing Numbers

- ❑ Viết một chương trình tạo một số ngẫu nhiên trong khoảng 0 đến 100.
 - Chương trình viết ra sẽ yêu cầu người dùng nhập vào liên tiếp các con số sao cho số nhập vào bằng với số ngẫu nhiên đã tạo ra trước đó.
 - Đối với mỗi lần input của người dùng, chương trình sẽ báo cho người dùng biết số nhập vào (bị sai) là bé hơn hay lớn hơn số ngẫu nhiên cho trước.
- Sinh viên viết chương trình bằng Java. Chụp ảnh màn hình (1) Code; (2) output của chương trình. Lưu lại trong file Word.

Ending a Loop with a Sentinel Value

- Thông thường, số lần lặp của một vòng lặp là không được xác định trước. Chúng ta có thể sử dụng một giá trị input để biểu thị sự kết thúc của vòng lặp.
- Giá trị như trên được gọi là một Sentinel Value (đóng vai trò như lính canh để giúp vòng lặp hữu hạn).
- Biến count được gọi là Sentinel

```
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

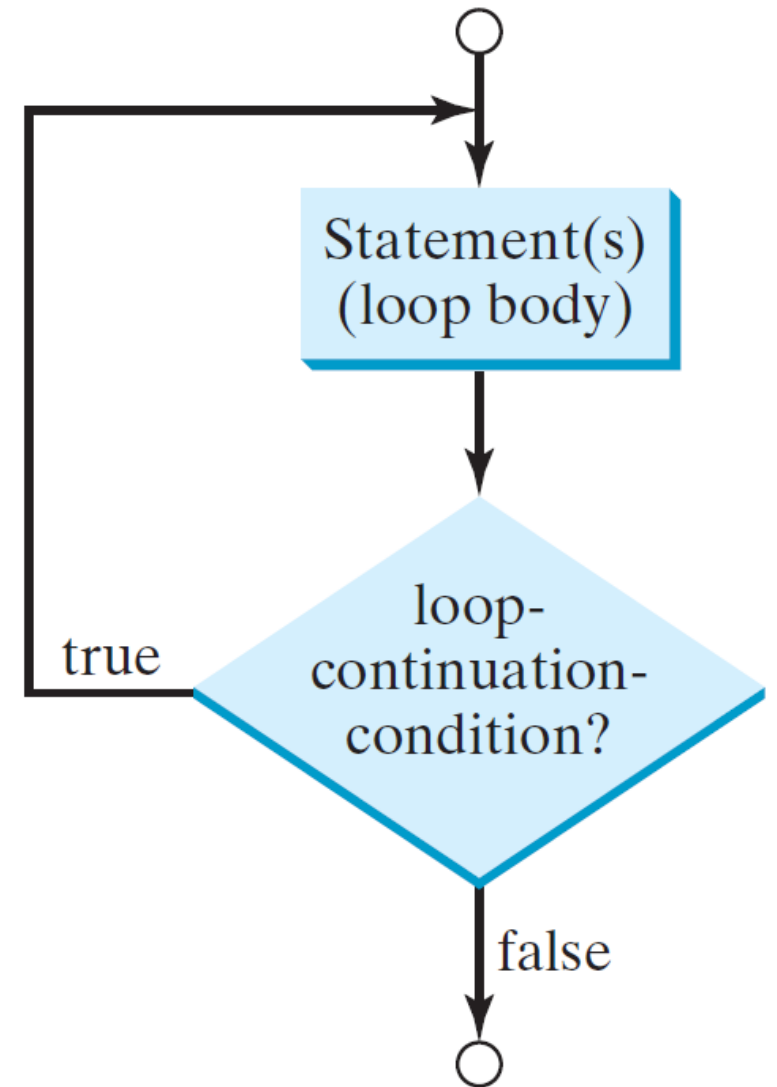

Caution

- Không sử dụng các giá trị có dấu phẩy động để kiểm tra trong việc quản lý vòng lặp.
- Vì giá trị có dấu phẩy động là giá trị mang tính chất tương đối với một số giá trị.

```
double item = 1; double sum = 0;
while (item != 0) { //Không đảm bảo item sẽ bằng 0
    sum += item;
    item -= 0.1; //Giảm giá trị item 0.1 đơn vị
}
System.out.println(sum);
```

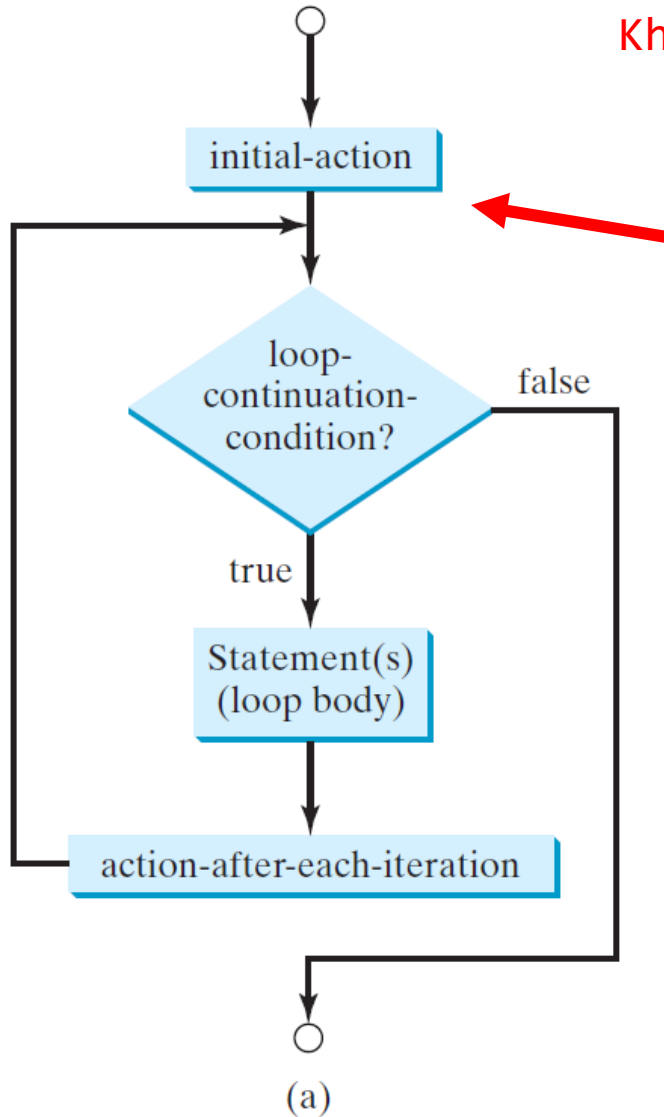
do-while Loop

```
do {  
    // Loop body  
    Statement(s);  
} while (điều-kiện-tiếp-tục-lặp);
```



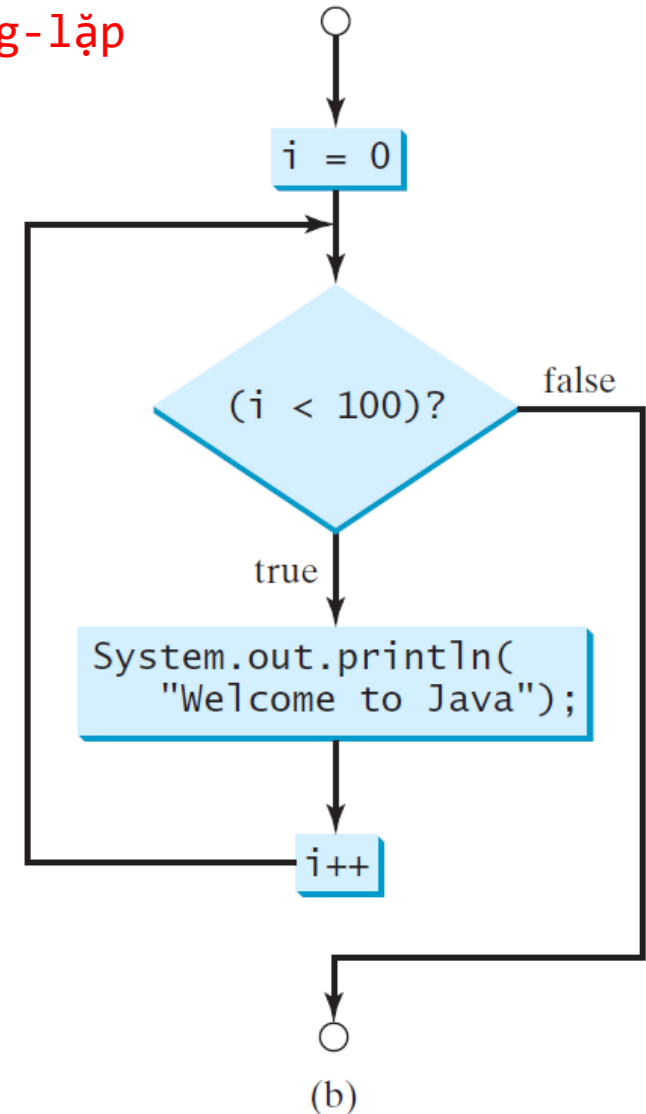
for Loops

Khởi-tạo; điều-kiện-lặp; hành-động-sau-mỗi-vòng-lặp



```
for (initial-action; loop-continuation-condition; action-after-each-iteration) {  
    // loop body;  
    Statement(s);  
}
```

```
int i;  
for (i = 0; i < 100; i++) {  
    System.out.println(  
        "Welcome to Java!");  
}
```



Note (1/2)

- Lưu ý: initial-action trong vòng lặp for có thể là một danh sách của không hay nhiều biểu thức ngăn cách nhau bởi dấu phẩy.
- action-after-each-iteration trong vòng lặp for có thể là một danh sách của không hay nhiều câu lệnh ngăn cách bởi dấu phẩy.

```
for (int i = 1; i < 100; System.out.println(i++));
```

```
for (int i = 0, j = 0; (i + j < 10); i++, j++) {  
    // Do something  
}
```

Note (2/2)

- Nếu loop-continuation-condition trong vòng lặp for bị lược bỏ, nó sẽ được hàm ý là luôn luôn đúng (true).
- Do đó câu lệnh ở (a), lặp vô hạn, là đúng cú pháp.
- Ngược lại, sẽ tốt hơn nếu chúng ta sử dụng một lệnh lặp tương đương ở (b) để tránh những sự nhầm lẫn.

```
for ( ; ; ) {  
    // Do something  
}
```

(a)

Equivalent

```
while (true) {  
    // Do something  
}
```

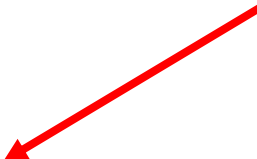
(b)

Caution (1/2)

- Thêm dấu chấm phẩy (;) ở cuối mệnh đề for và trước thân của vòng lặp là một lỗi hay gặp:

```
for (int i = 0; i < 10; i++);  
{  
    System.out.println("i is " + i);  
}
```

Logic Error



Caution (2/2)

- Tương tự như vậy, những vòng lặp sau là sai:

```
int i=0;
while (i < 10);
{
    System.out.println("i is " + i);
    i++;
}
```

Logic
Error

- Tuy nhiên, trong trường hợp lệnh lặp do-while, dấu chấm phẩy ở cuối vòng lặp là cần thiết.

```
int i=0;
do {
    System.out.println("i is " + i);
    i++;
} while (i<10);
```

Đúng

Which Loop to Use?

- Có 3 loại lệnh lặp, while, do-while, và for, được xem là tương đương với nhau. Ví dụ, lệnh lặp while có thể được chuyển sang for:

```
while (loop-continuation-condition) {  
    // Loop body  
}
```

(a)

Equivalent

```
for ( ; loop-continuation-condition; )  
    // Loop body  
}
```

(b)

- Tương tự, lệnh for cũng có thể được chuyển đổi sang lệnh while:

```
for (initial-action;  
     loop-continuation-condition;  
     action-after-each-iteration) {  
    // Loop body;  
}
```

(a)

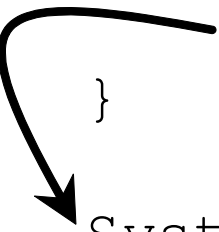
Equivalent

```
initial-action;  
while (loop-continuation-condition) {  
    // Loop body;  
    action-after-each-iteration;  
}
```

(b)

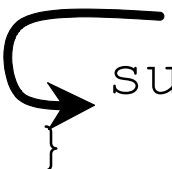
Using break and continue

```
public class TestBreak {  
    public static void main(String[] args) {  
        int sum = 0;  
        int number = 0;  
  
        while (number < 20) {  
            number++;  
            sum += number;  
            if (sum >= 100)  
                break;  
        }  
        System.out.println("The number is " + number);  
        System.out.println("The sum is " + sum);  
    }  
}
```



continue

```
public class TestContinue {  
    public static void main(String[] args) {  
        int sum = 0;  
        int number = 0;  
  
        while (number < 20) {  
            number++;  
            if (number == 10 || number == 11)  
                continue;  
            sum += number;  
        }  
  
        System.out.println("The sum is " + sum);  
    }  
}
```



Exercise 2: Displaying Prime Numbers

Problem: Write a program that displays the first 50 prime numbers in five lines, each of which contains 10 numbers. An integer greater than 1 is *prime* if its only positive divisor is 1 or itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not.

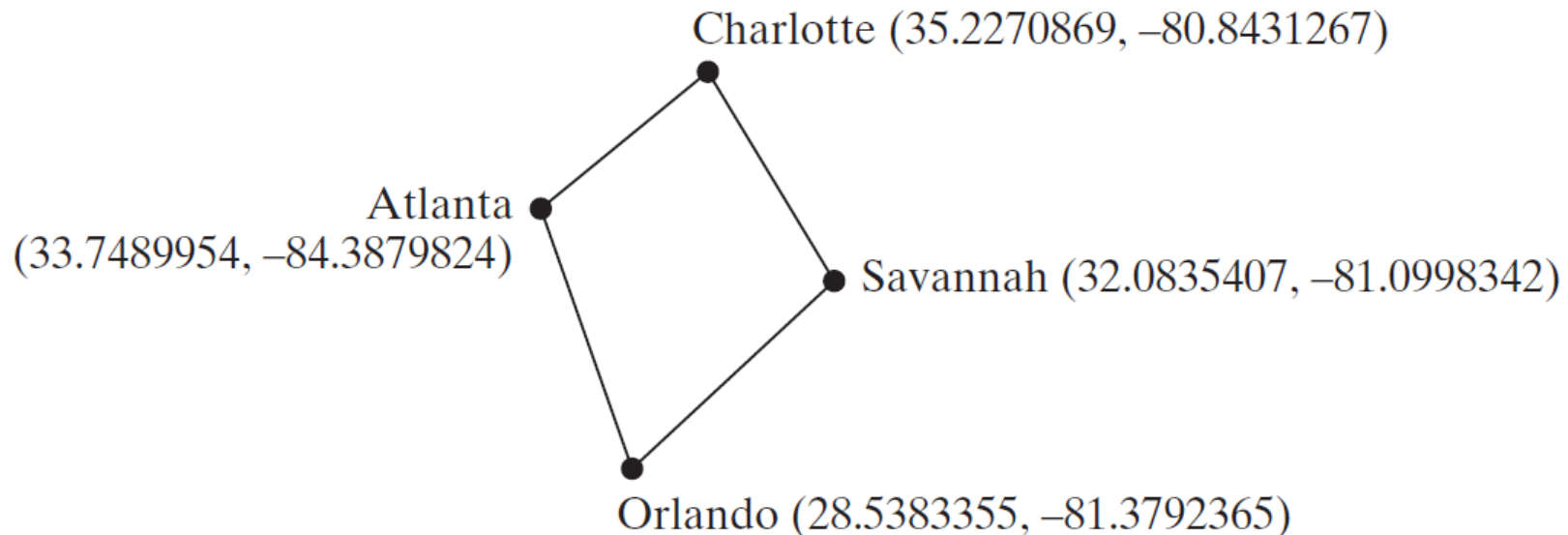
Solution: The problem can be broken into the following tasks:

- For number = 2, 3, 4, 5, 6, ..., test whether the number is prime.
- Determine whether a given number is prime.
- Count the prime numbers.
- Print each prime number, and print 10 numbers per line.
- Sinh viên viết chương trình bằng Java. Chụp ảnh màn hình (1) Code; (2) output của chương trình. Lưu lại trong file Word.

Hàm Toán Học – Kiểu Ký Tự - Kiểu Chuỗi

Motivations

- Giả sử chúng ta cần ước tính diện tích của một vùng bao quanh bởi 4 thành phố, tọa độ GPS của các thành phố này được cho bởi vĩ độ và kinh độ.
- Làm sao chúng ta viết được chương trình?



Objectives

- Giải các bài toán sử dụng phương thức trong lớp **Math** (§4.2).
- Biểu diễn các ký tự sử dụng kiểu **char** (§4.3).
- Mã hóa các ký tự bằng cách sử dụng bảng mã ASCII và Unicode (§4.3.1).
- Chuyển đổi một giá trị số thành một ký tự và chuyển đổi một ký tự thành một số nguyên (§4.3.3).
- So sánh và kiểm tra các ký tự bằng cách sử dụng các phương thức tĩnh trong lớp **Character** (§4.3.4).
- Giới thiệu các đối tượng và các bản sao (**instances**) phương thức (§4.4).
- Biểu diễn chuỗi ký tự sử dụng đối tượng **String** (§4.4).
- Đọc một ký tự hoặc chuỗi ký tự từ console (§4.4.4 - §4.4.5).
- So sánh các chuỗi ký tự với phương thức **equals** và phương thức **compareTo** (§4.4.6).
- Tìm một ký tự hoặc chuỗi con sử dụng phương thức **indexOf** (§4.4.8).
- Định dạng hiển thị sử dụng phương thức **System.out.printf** (§4.6).

Mathematical Functions

- Java cung cấp nhiều phương thức (methods) trong lớp Math thực thi các hàm toán học thông thường.
- Lớp Math được sử dụng với với các hằng số và phương thức sau:

Class constants:

☞ **PI**

☞ **E**

Class methods:

☞ **Lượng giác** (Trigonometric)

☞ **Số mũ** (Exponent)

☞ **Làm tròn** (Round)

☞ **min, max, abs, và random**

Trigonometric Methods

- `sin(double a)`
- `cos(double a)`
- `tan(double a)`
- `acos(double a)`
- `asin(double a)`
- `atan(double a)`

Ví dụ:

```
Math.sin(0) //trả về 0.0
```

```
Math.sin(Math.PI / 6) //trả về 0.5
```

```
Math.sin(Math.PI / 2) //trả về 1.0
```

```
Math.cos(0) //trả về 1.0
```

```
Math.cos(Math.PI / 6) //trả về 0.866
```

```
Math.cos(Math.PI / 2) //trả về 0
```


Exponent Methods

- **exp(double a)**

Trả về e mũ a .

- **log(double a)**

Trả về logarit số tự nhiên của a .

- **log10(double a)**

Trả về logarit thập phân của a .

- **pow(double a, double b)**

Trả về a mũ b .

- **sqrt(double a)**

Trả về căn bậc hai của a .

Ví dụ:

```
Math.exp(1) // trả về 2.71
```

```
Math.log(2.71) // trả về 1.0
```

```
Math.pow(2, 3) // trả về 8.0
```

```
Math.pow(3, 2) // trả về 9.0
```

```
Math.pow(3.5, 2.5) // trả về 22.91765
```

```
Math.sqrt(4) // trả về 2.0
```

```
Math.sqrt(10.5) // trả về 3.24
```

Rounding Methods

- **double ceil(double x)**

Làm tròn x lên đến số nguyên gần nhất. Số trả về là một số dạng double.

- **double floor(double x)**

Làm tròn x xuống số nguyên gần nhất. Số trả về là một số dạng double.

- **double rint(double x)**

Làm tròn x tới số nguyên gần nhất. Ví dụ: `rint(100.500) = 100.0`; `rint(100.2) = 100.0`

- **int round(float x)**

Trả về giá trị làm tròn, tương đương với `(int)Math.floor(x+0.5)`.

- **long round(double x)**

Trả về giá trị làm tròn, tương đương với `(long)Math.floor(x+0.5)`.

Rounding Methods Examples

```
Math.ceil(2.1) //trả về 3.0
Math.ceil(2.0) //trả về 2.0
Math.ceil(-2.0) //trả về -2.0
Math.ceil(-2.1) //trả về -2.0
Math.floor(2.1) //trả về 2.0
Math.floor(2.0) //trả về 2.0
Math.floor(-2.0) //trả về -2.0
Math.floor(-2.1) //trả về -3.0
```

```
Math rint(2.1) //trả về 2.0
Math rint(2.0) //trả về 2.0
Math rint(-2.0) //trả về -2.0
Math rint(-2.1) //trả về -2.0
Math rint(2.5) //trả về 2.0
Math rint(-2.5) //trả về -2.0
Math.round(2.6f) //trả về 3
Math.round(2.0) //trả về 2
Math.round(-2.0f) //trả về -2
Math.round(-2.6) //trả về -3
```

min, max, and abs

- **max(a, b)** and **min(a, b)**

Trả về giá trị lớn/nhỏ nhất trong hai giá trị đầu vào a, b.

- **abs(a)**

Trả về giá trị tuyệt đối của a.

- **random()**

Trả về số ngẫu nhiên double trong khoảng [0.0, 1.0).

Ví dụ:

`Math.max(2, 3)` returns 3

`Math.max(2.5, 3)` returns 3.0

`Math.min(2.5, 3.6)` returns 2.5

`Math.abs(-2)` returns 2

`Math.abs(-2.1)` returns 2.1

The random Method

- Tạo một giá trị ngẫu nhiên dạng số thực (double) lớn hơn hoặc bằng 0.0 và bé hơn 1.0 ($0 \leq \text{Math.random()} < 1.0$).

Ví dụ:

`(int) (Math.random() * 10)` \longrightarrow Returns a random integer between 0 and 9.

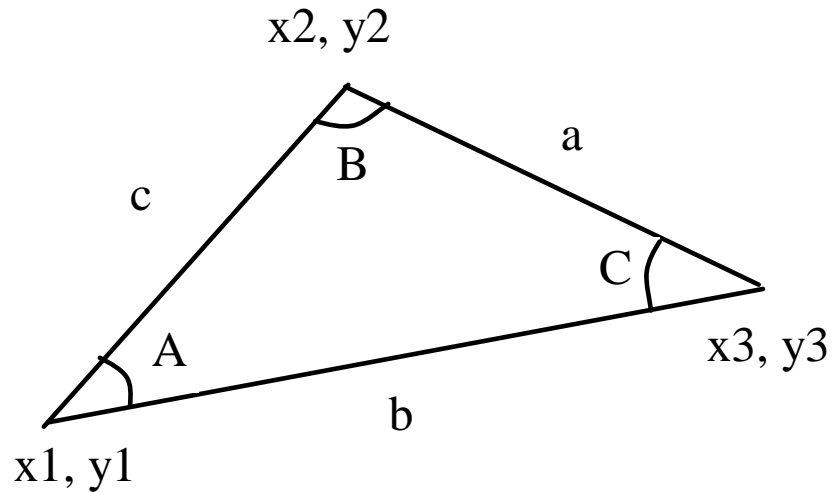
`50 + (int) (Math.random() * 50)` \longrightarrow Returns a random integer between 50 and 99.

Một cách tổng quát:

`a + Math.random() * b` \longrightarrow Returns a random number between a and a + b, excluding a + b.

Exerciser 1: Computing Angles of a Triangle

- Viết chương trình để người dùng nhập vào tọa độ x và y trên trục tọa độ của 3 đỉnh tam giác. Hiển thị góc của các đỉnh này.




$$A = \arccos \left(\frac{a^2 - b^2 - c^2}{-2bc} \right)$$
$$B = \arccos \left(\frac{b^2 - a^2 - c^2}{-2ac} \right)$$
$$C = \arccos \left(\frac{c^2 - b^2 - a^2}{-2ab} \right)$$

➤ Sinh viên viết đoạn chương trình. Nộp lại (1) ảnh đoạn code (2) output.

Character Data Type

```
char letter = 'A'; // ASCII      Bốn chữ số thập lục phân.  
char numChar = '4'; // ASCII  
char letter = '\u0041'; // Unicode  
char numChar = '\u0034'; // Unicode
```



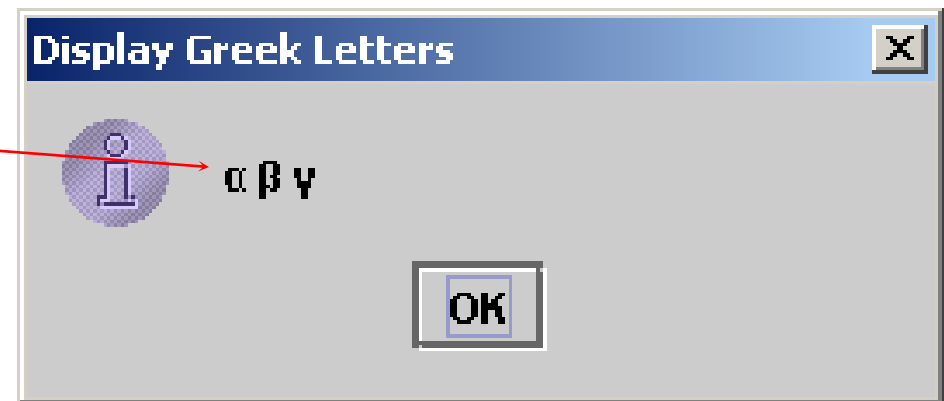
Lưu ý: Toán tử tăng và giảm cũng có thể được sử dụng cho các biến kiểu ký tự (char) để. Sử dụng các toán tử này sẽ trả về các ký tự liền sau hoặc liền trước trong bảng mã Unicode. Ví dụ, câu lệnh sau cho ra kết quả là b.

```
char ch = 'a';  
System.out.println(++ch);
```

Unicode Format

- Ký tự trong Java sử dụng bảng mã Unicode (bảng mã 16-bit).
- Ký tự Unicode cần 2 byte để lưu trữ.
- Ký tự Unicode được khai báo bắt đầu bởi \u kết hợp với 4 số thập lục phân từ ‘\u0000’ đến ‘\uFFFF’.
- Với cấu trúc trên, Unicode có thể biểu diễn được $65535 + 1$ ký tự.

Unicode \u03b1 \u03b2 \u03b3
là ba ký tự Hy Lạp



ASCII Code for Commonly Used Characters

Characters	Code Value in Decimal	Unicode Value
'0' to '9'	48 to 57	\u0030 to \u0039
'A' to 'Z'	65 to 90	\u0041 to \u005A
'a' to 'z'	97 to 122	\u0061 to \u007A

Escape Sequences for Special Characters

Escape sequence (Ký tự điều khiển): Escape sequence là chuỗi ký tự nó thường được sử dụng với một số các nghiệp vụ cụ thể ... Theo cách nói khác thì **Escape sequence** chính là ký tự xử lý dòng hay ký tự điều khiển trong Java.

<i>Escape Sequence</i>	<i>Name</i>	<i>Unicode Code</i>	<i>Decimal Value</i>
<code>\b</code>	Ký tự Backspace	<code>\u0008</code>	8
<code>\t</code>	Ký tự Tab	<code>\u0009</code>	9
<code>\n</code>	Ký tự xuống dòng	<code>\u000A</code>	10
<code>\f</code>	Cuộn trang giấy vào máy tin	<code>\u000C</code>	12
<code>\r</code>	Xuống hàng	<code>\u000D</code>	13
<code>\\</code>	Dấu gạch chéo	<code>\u005C</code>	92
<code>\"</code>	Dấu nháy kép	<code>\u0022</code>	34

ASCII Character Set (1/2)

- Bảng mã ASCII là một tập con của bảng mã Unicode từ \u0000 đến \u007f

TABLE B.1 ASCII Character Set in the Decimal Index

	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dcl	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	\$	%	&	'
4	()	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[\]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{		}	~	del		

ASCII Character Set (2/2)

TABLE B.2 ASCII Character Set in the Hexadecimal Index

	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	nl	vt	ff	cr	so	si
1	dle	dcl	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
2	sp	!	“	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	del

Casting between char and Numeric Types

❑ Chuyển đổi giữa các kiểu ký tự và kiểu số

```
int i = 'a'; // Same as int i = (int)'a';
```

```
char c = 97; // Same as char c = (char)97;
```

Comparing and Testing Characters

```
if (ch >= 'A' && ch <= 'Z')
```

```
    System.out.println(ch + " la mot ky tu viet hoa");
```

```
else if (ch >= 'a' && ch <= 'z')
```

```
    System.out.println(ch + " la mot ky tu viet thuong");
```

```
else if (ch >= '0' && ch <= '9')
```

```
    System.out.println(ch + " la mot ky tu dang so hoc");
```

Methods in the Character Class

Method	Description
<code>isDigit(ch)</code>	Trả về giá trị TRUE nếu ch là một số
<code>isLetter(ch)</code>	Trả về giá trị TRUE nếu ch là một ký tự
<code>isLetterOfDigit(ch)</code>	Trả về giá trị TRUE nếu ch là một số hay ký tự
<code>isLowerCase(ch)</code>	Trả về giá trị TRUE nếu ch là một ký tự viết thường
<code>isUpperCase(ch)</code>	Trả về giá trị TRUE nếu ch là một ký tự viết hoa
<code>toLowerCase(ch)</code>	Trả về ký tự viết thường của ch
<code>toUpperCase(ch)</code>	Trả về ký tự viết hoa của ch

The String Type

- Kiểu char chỉ biểu diễn một ký tự đơn nhất. Để biểu diễn một chuỗi các ký tự, chúng ta phải sử dụng một kiểu dữ liệu String. Ví dụ:

```
String message = "Welcome to Java";
```

- String là một lớp được định nghĩa sẵn trong thư viện Java.
- Kiểu String không phải là một kiểu dữ liệu nguyên thủy (primitive).

Simple Methods for String Objects (1/2)

Method	Description
<code>length()</code>	Trả về số lượng ký tự của chuỗi hiện tại
<code>charAt(index)</code>	Trả về ký tự ở vị trí index trong chuỗi hiện tại
<code>concat(s1)</code>	Trả về một chuỗi mới mà nối với chuỗi s1
<code>toUpperCase()</code>	Trả về một chuỗi mới với tất cả ký tự được viết hoa
<code>toLowerCase()</code>	Trả về một chuỗi mới với tất cả ký tự được viết thường
<code>trim()</code>	Trả về một chuỗi với các khoảng trắng bị cắt bớt ở hai đầu

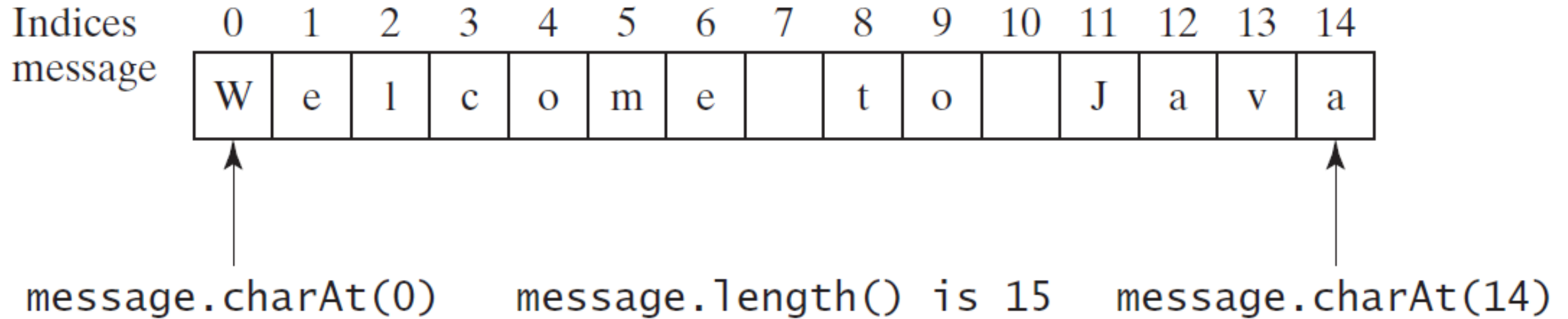
Simple Methods for String Objects (2/2)

- Các chuỗi ký tự là đối tượng trong Java.
- Những phương thức ở bảng trong slide trước chỉ có thể được gọi từ một instance kiểu string.
- Vì lý do này, những phương thức này được gọi là *instance* (trường hợp cụ thể) *method*.
- Một phương thức *non-instance* được gọi là phương thức tĩnh (*static method*).
- Tất cả các phương thức trong lớp **Math** là *static method*.

Getting String Length

```
String message = "Welcome to Java";  
System.out.println("Do dai cua chuoï " + message + " la "  
    + message.length());
```

Getting Characters from a String



```
String message = "Welcome to Java";
```

```
System.out.println("Ky tu dau tien trong chuoai message la "  
    + message.charAt(0));
```

Converting Strings

// Trả về một chuỗi mới, welcome.

"Welcome".toLowerCase()

// Trả về một chuỗi mới, WELCOME.

"Welcome".toUpperCase()

// Trả về một chuỗi mới, Welcome.

" Welcome ".trim()

String Concatenation

```
String s3 = s1.concat(s2); ⇔ String s3 = s1 + s2;
```

```
// Ba chuỗi được nối lại (concatenated)
```

```
String message = "Welcome " + "to " + "Java";
```

```
// Chuỗi Chapter được nối với số 2
```

```
String s = "Chapter" + 2; //s trở thành Chapter2
```

```
// String Supplement is concatenated with character B
```

```
String s1 = "Supplement" + 'B'; //s1 trở thành SupplementB
```

Reading a String from the Console

```
Scanner input = new Scanner(System.in);  
System.out.print("Nhập vào 3 chuỗi ngăn cách bởi khoảng  
trắng: ");  
String s1 = input.next();  
String s2 = input.next();  
String s3 = input.next();  
System.out.println("s1 is " + s1);  
System.out.println("s2 is " + s2);  
System.out.println("s3 is " + s3);
```

Reading a Character from the Console

```
Scanner input = new Scanner(System.in);  
System.out.print("Nhập vào 1 ký tự: ");  
String s = input.nextLine();  
char ch = s.charAt(0);  
System.out.println("Ký tự nhập vào là " + ch);
```


Comparing Strings

Method	Description
<code>equals(s1)</code>	Trả về TRUE nếu chuỗi hiện tại bằng s1
<code>equalsIgnoreCase(s1)</code>	Trả về TRUE nếu chuỗi hiện tại bằng s1, bất kể chữ in hoa hay thường
<code>compareTo(s1)</code>	Trả về một số nguyên lớn hơn, bằng, hoặc bé hơn 0 biểu thị liệu chuỗi hiện tại là lớn hơn, bằng hay bé hơn s1.
<code>compareToIgnoreCase(s1)</code>	Tương tự như <i>compareTo</i> ngoại trừ nó so sánh bất kể chữ in hoa hay thường
<code>startsWith(prefix)</code>	Trả về TRUE nếu một chuỗi bắt đầu với một tiền tố (prefix)
<code>endsWith(suffix)</code>	Trả về TRUE nếu một chuỗi bắt đầu với một hậu tố (suffix)

Obtaining Substrings

Method

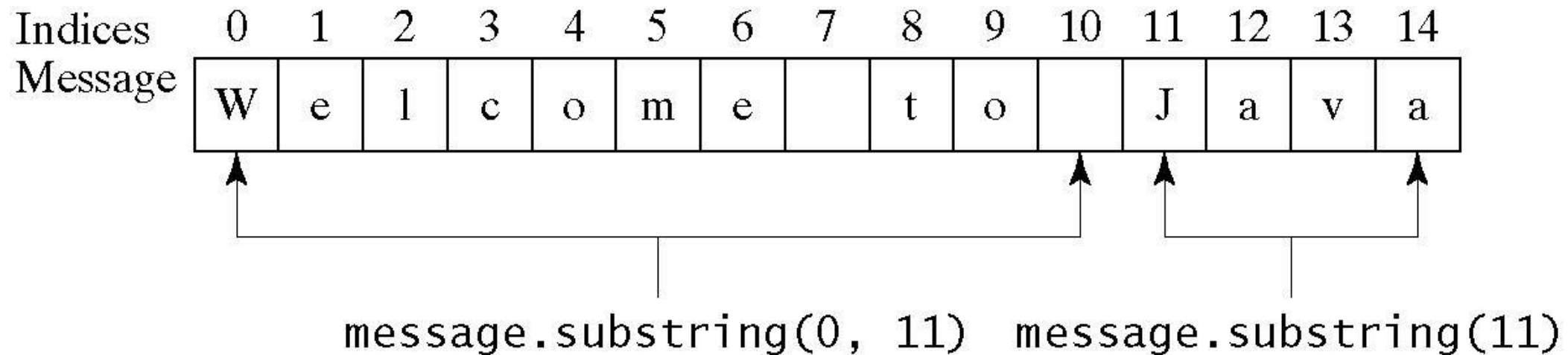
Description

`substring(beginIndex)`

Trả về một chuỗi con của chuỗi hiện tại, bắt đầu từ ký tự với chỉ số (**beginIndex**) đến cuối của chuỗi hiện tại. Xem hình dưới.

`substring(beginIndex, endIndex)`

Trả về một chuỗi con của chuỗi hiện tại, bắt đầu từ ký tự với chỉ số (**beginIndex**) đến ký tự với chỉ số (**endIndex - 1**). Xem hình dưới. Lưu ý, ký tự ở `endIndex` không nằm trong chuỗi con



Finding a Character or a Substring in a String (1/2)

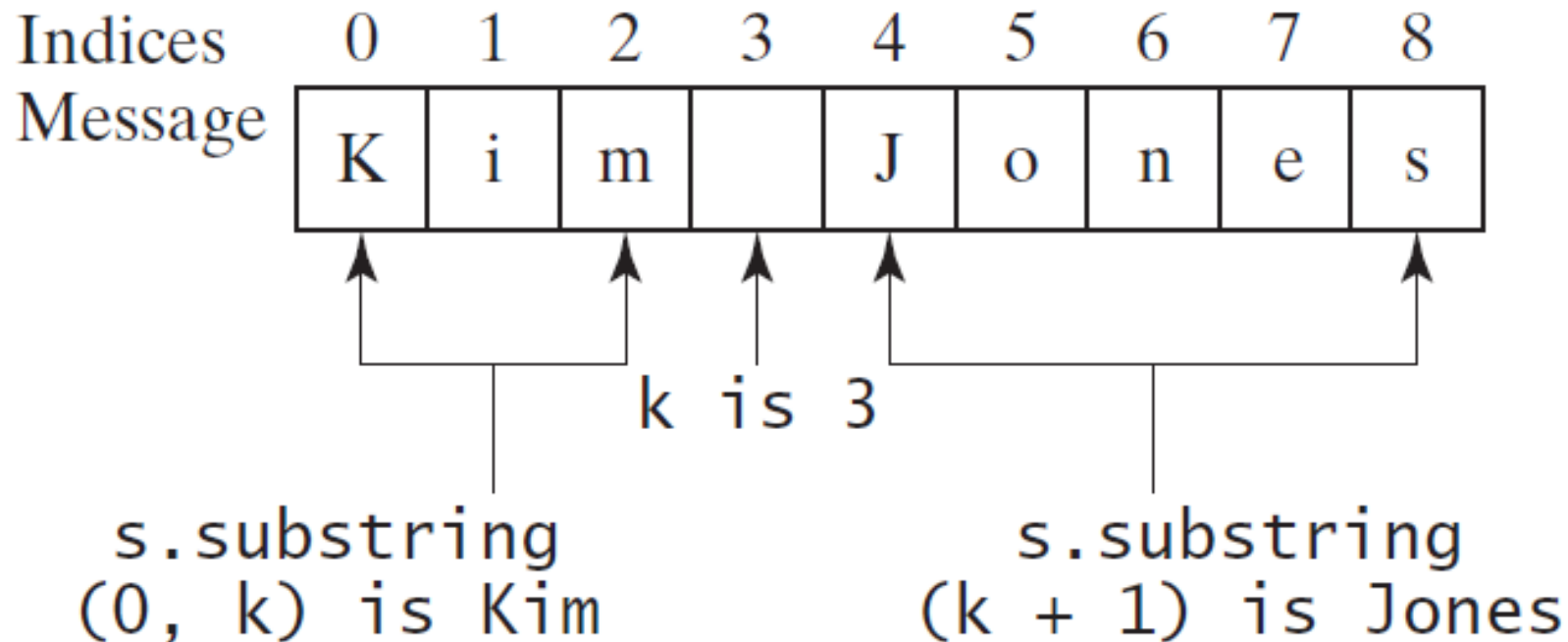
Method	Description
<code>indexOf(ch)</code>	Trả về vị trí (chỉ số) lần xuất hiện đầu tiên của ch trong chuỗi hiện tại. Trả về giá trị -1 nếu không tìm thấy
<code>indexOf(ch, fromIndex)</code>	Trả về vị trí lần xuất hiện đầu tiên của ch trong chuỗi hiện tại sau vị trí fromIndex . Trả về giá trị -1 nếu không tìm thấy
<code>indexOf(s)</code>	Trả về vị trí (chỉ số) lần xuất hiện đầu tiên của chuỗi s trong chuỗi hiện tại. Trả về giá trị -1 nếu không tìm thấy
<code>indexOf(s, fromIndex)</code>	Trả về vị trí (chỉ số) lần xuất hiện đầu tiên của chuỗi s trong chuỗi hiện tại sau vị trí fromIndex . Trả về giá trị -1 nếu không tìm thấy
<code>lastIndexOf(ch)</code>	Trả về vị trí của vị trí xuất hiện cuối cùng của ch trong chuỗi hiện tại. Trả về giá trị -1 nếu không tìm thấy.
<code>lastIndexOf(ch, fromIndex)</code>	Trả về vị trí của vị trí xuất hiện cuối cùng của ch trong chuỗi hiện tại sau vị trí fromIndex . Trả về giá trị -1 nếu không tìm thấy.
<code>lastIndexOf(s)</code>	Trả về vị trí của vị trí xuất hiện cuối cùng của chuỗi s trong chuỗi hiện tại. Trả về giá trị -1 nếu không tìm thấy.
<code>lastIndexOf(s, fromIndex)</code>	Trả về vị trí của vị trí xuất hiện cuối cùng của chuỗi s trong chuỗi hiện tại sau vị trí fromIndex . Trả về giá trị -1 nếu không tìm thấy.

Finding a Character or a Substring in a String (2/2)

```
int k = s.indexOf(' ');
```

```
String firstName = s.substring(0, k);
```

```
String lastName = s.substring(k + 1);
```



Conversion between Strings and Numbers

```
int intValue = Integer.parseInt(intString);
```

```
double doubleValue = Double.parseDouble(doubleString);
```

```
String s = number + "";
```

Formatting Output

- Sử dụng câu lệnh **printf**.

System.out.printf(format, items);

- Trong đó **format** là một chuỗi có thể bao gồm các chuỗi con và mã định dạng. Trình xác định định dạng chỉ định cách hiển thị một mục.
- Mỗi **item** có thể là một giá trị số, ký tự, giá trị **boolean** hoặc **String**.
- Mỗi **item** bắt đầu bằng dấu phần trăm.

Frequently-Used Specifiers

Specifier Output

%b	một giá trị boolean
%c	một ký tự
%d	một số nguyên thập phân
%f	một số có dấu phẩy động
%e	một số với ký hiệu khoa học tiêu chuẩn
%s	một chuỗi

Example

true or false
'a'
200
45.460000
4.556000e+01
"Java is cool"

```
int count = 5;  
double amount = 45.56;  
System.out.printf("count is %d and amount is %f", count, amount);
```



display count is 5 and amount is 45.560000

Q&A

