

HỌC PHẦN: CÔNG NGHỆ JAVA

MẢNG VÀ CHUỖI

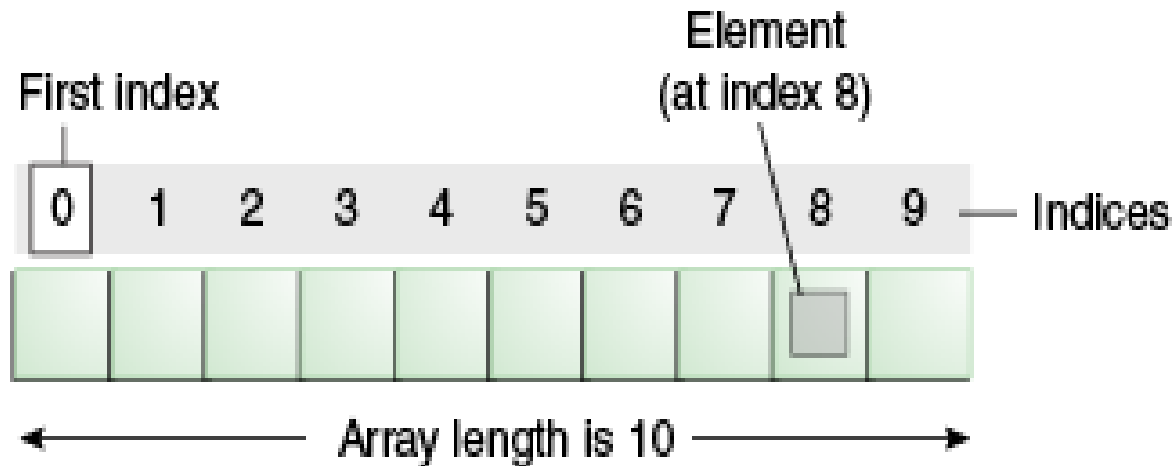


NỘI DUNG

- ⊙ Hiểu được cấu trúc của mảng
- ⊙ Phân biệt được mảng 1 chiều và mảng nhiều chiều
- ⊙ Một số thao tác cơ bản với mảng
- ⊙ Sử dụng lớp tiện ích Arrays
- ⊙ Hiểu về chuỗi và các xử lý chuỗi

GIỚI THIỆU MẢNG

- ❑ Là kiểu dữ liệu có cấu trúc gồm một tập hợp cố định các phần tử có cùng kiểu dữ liệu
- ❑ Các thông tin liên quan đến mảng: tên mảng, số phần tử của mảng, kiểu dữ liệu của mảng
- ❑ Truy xuất các phần tử mảng bằng chỉ số, bắt đầu là 0
- ❑ Sử dụng thuộc tính `length` để lấy số phần tử của mảng



❑ Ưu điểm của mảng

- ❖ Sử dụng mảng để lưu trữ được nhiều giá trị thay vì phải khai báo nhiều biến.
- ❖ Truy xuất nhanh
- ❖ Dễ dàng đọc, sắp xếp và tìm kiếm dữ liệu từ các phần tử

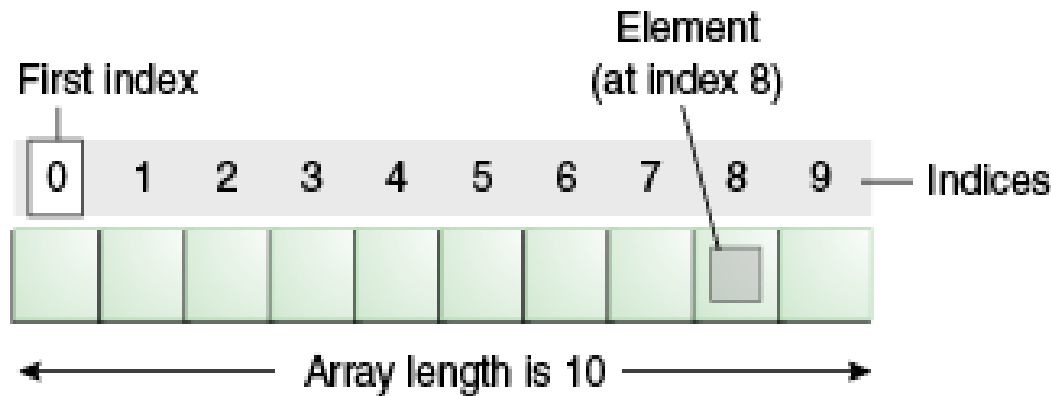
❑ Nhược điểm của mảng

- ❖ Số phần tử cố định nên không thể bổ sung hoặc bớt
- ❖ Cấp phát liên tục nên cần phải có vùng bộ nhớ liên tục đủ lớn để cấp phát.

PHÂN LOẠI MẢNG

- ❑ Mảng một chiều
- ❑ Mảng nhiều chiều (mảng của các mảng)

1 chiều



2 chiều

	0	1	2	3	
a	a[0][0]	a[0][1]	a[0][2]	a[0][3]	0
	a[1][0]	a[1][1]	a[1][2]	a[0][3]	1
	a[2][0]	a[2][1]	a[2][2]	a[2][3]	2

KHAI BÁO MẢNG 1 CHIỀU

❑ Khai báo mảng

❖ `datatype[] arr;` \rightarrow `int[] A;`

❖ `datatype arr[];` \rightarrow `int A[];`

❑ Khởi tạo kích thước mảng

`arr = new datatype[size];` \rightarrow `A = new int[10];`

❑ Khai báo và khởi tạo luôn kích thước

`datatype[] arr = new datatype[size];` \rightarrow `int[] A = new int[10];`

❑ Khai báo và khởi tạo các phần tử

`datatype[] arr = {elem1, elem2,...};` \rightarrow `int[] A = {2,4,6,7,8,9};`

❑ Chú ý:

❖ `datatype[] a, b;` // khai báo 2 mảng cùng kiểu

❖ `datatype a[], b;` // khai báo 1 mảng và 1 biến

VÍ DỤ

```
int a[] = {4, 3, 5, 7}; // khai báo và khởi tạo  
a[2] = a[1] * 4; // 3*4=12  
System.out.println(a.length); // xuất số phần tử của mảng
```

Sau phép gán `a[2]` mảng là {4, 3, 12, 7};

DUYỆT MẢNG 1 CHIỀU

- ❑ Có thể sử dụng bất kỳ vòng lặp nào để duyệt mảng. Tuy nhiên 2 vòng lặp thường được sử dụng để duyệt mảng là for và for-each.

```
int[] a = {4, 3, 5, 9};  
for(int i=0; i<a.length; i++){  
    System.out.println(a[i]);  
}
```

← **for(;;)**

for-each →

```
int[] a = {4, 3, 5, 9};  
for (int x : a){  
    System.out.println(x);  
}
```


VÍ DỤ TÌM SỐ NHỎ NHẤT

```
int arr[] = { 33, 3, 4, 5 };  
int min = arr[0];  
for(int i = 1; i < arr.length; i++){  
    if (min > arr[i]){  
        min = arr[i];  
    }  
}  
System.out.println(min);
```

- ❑ Cho mảng $A = \{3, 7, 23, -45, 33, 73, -56\}$
- ❑ Hãy viết chương trình thực hiện các công việc sau đây
 - ❖ Tìm số lớn nhất, nhỏ nhất trong mảng
 - ❖ Tính trung bình cộng các số chẵn trong mảng

```
int A[] = {3, 7, 23, -45, 33, 73, -56};
int sum=0, count=0;      float aveg=0;
for(int i=0;i<A.length;i++){
    if(A[i]%2==0) {
        count++;
        sum = sum + A[i]; } }
tb=(float) sum/count;
System.out.printf("Trung binh cong cac so chan: %0.2f", aveg);
```

- ❑ Có thể sắp xếp tăng dần hoặc giảm dần các phần tử trong mảng

```
int a[] = {8,2,6,2,9,1,5};  
for(int i=0; i<a.length-1; i++){  
    for(int j=i+1; j<a.length; j++){  
        if(a[i] > a[j]){  
            int temp = a[i];  
            a[i] = a[j];  
            a[j] = temp;  
        }  
    }  
}
```

Nếu thay đổi toán tử so sánh thành $<$ thì thuật toán trở thành sắp xếp giảm dần.

❑ **System.arraycopy**(src, srcPos, dest, destPos, length)
được sử dụng để copy mảng

❖ Trong đó:

- src là mảng nguồn
- dest là mảng đích
- srcPos là vị trí bắt đầu copy
- destPost là vị trí bắt đầu của mảng đích
- length là số phần tử cần copy

❑ Ví dụ tạo mảng b từ mảng a bỏ số phần tử có giá trị 4

```
int a[] = {1, 2, 3, 4, 5, 6};  
int b[] = new int[5];  
System.arraycopy(a, 0, b, 0, 3);  
System.arraycopy(a, 4, b, 3, 2);
```



b={1,2,3,5,6}

LỚP TIỆN ÍCH ARRAYS


```
Int[] a = {1, 9, 2, 8, 3, 7, 4, 6, 5};
```

Phương thức	Mô tả/ví dụ
<code><T> List<T> asList(T... a)</code>	Chuyển một mảng sang List với kiểu tương ứng. Ví dụ: List<Integer> b = Arrays.asList(a);
<code>int binarySearch(Object[] a, Object key)</code>	Tìm vị trí xuất hiện đầu tiên của một phần tử trong mảng. Ví dụ: int i = Arrays.binarySearch(a, 8);
<code>void sort(Object[] a)</code>	Sắp xếp các phần tử theo thứ tự tăng dần. Ví dụ: Arrays.sort(a);
<code>String toString(Object[] a)</code>	Chuyển mảng thành chuỗi được bọc giữ cặp dấu [] và các phần tử mảng cách nhau dấu phẩy. Ví dụ: String s = Arrays.toString(a);
<code>void fill(Object[] a, Object val)</code>	Gán 1 giá trị cho tất cả các phần tử mảng. Ví dụ: Arrays.fill(a, 9);

*Chú ý: **binarySearch()** chỉ làm việc với mảng đã được sắp xếp tăng dần*

LỚP TIỆN ÍCH ARRAYS

```
int[] a = {9, 3, 8, 7, 3, 9, 4, 2};  
  
System.out.println("Mảng gốc: " + Arrays.toString(a));  
  
Arrays.sort(a);  
System.out.println("Sau sort: " + Arrays.toString(a));  
  
int i = Arrays.binarySearch(a, 8);  
System.out.println("Vị trí của 8 là " + i);  
  
Arrays.fill(a, 0);  
System.out.println("Sau fill: " + Arrays.toString(a));
```



Mảng gốc: [9, 3, 8, 7, 3, 9, 4, 2]
Sau sort: [2, 3, 3, 4, 7, 8, 9, 9]
Vị trí của 8 là 5
Sau fill: [0, 0, 0, 0, 0, 0, 0, 0]

MẢNG NHIỀU CHIỀU

	0	1	2	3	
a	a[0][0]	a[0][1]	a[0][2]	a[0][3]	0
	a[1][0]	a[1][1]	a[1][2]	a[0][3]	1
	a[2][0]	a[2][1]	a[2][2]	a[2][3]	2

Mảng 2 chiều

Mảng 3 chiều

Index	0	1	2	3	4	Index	0	1	2	3	4
0	65,340	12,483	138,189	902,960	633,877	0	65,340	12,483	138,189	902,960	633,877
1	5,246	424,642	650,380	821,254	866,122	1	5,246	424,642	650,380	821,254	866,122
2	89,678	236,781	601,691	329,274	913,534	2	89,678	236,781	601,691	329,274	913,534
3	103,902	4,567	733,611	263,010	85,550	3	103,902	4,567	733,611	263,010	85,550
4	2,778	658,305	128,788	978,155	620,702	4	2,778	658,305	128,788	978,155	620,702
5	45,024	55,058	705,586	89,672	384,605	5	45,024	55,058	705,586	89,672	384,605
6	780	47,538	523,784	556,801	617,107	6	780	47,538	523,784	556,801	617,107
7	32,667	350,890	834,753	638,108	85,188	7	32,667	350,890	834,753	638,108	85,188
8	56,083	145,582	775,040	548,322	756,587	8	56,083	145,582	775,040	548,322	756,587
9	41,123	543,542	537,738	513,048	418,482	9	41,123	543,542	537,738	513,048	418,482

MẢNG NHIỀU CHIỀU

- ❑ Mảng nhiều chiều hay còn gọi là mảng của các mảng
- ❑ Mỗi phần tử của mảng là một mảng khác
- ❑ Khai báo 2 chiều
 - ❖ `datatype[][] arr;`
 - ❖ `datatype arr[][];`
 - ❖ `datatype [] arr[];`
- ❑ Khai báo có khởi tạo
 - ❖ `datatype[][] arr=new datatype[size1][size2];`
 - ❖ `datatype[][] arr = {{elem1, elem2}, {elem1, elem2}}`

VÍ DỤ MẢNG 2 CHIỀU

```
// Khai báo và khởi tạo
int arr[][] = { { 1, 2, 3 }, { 2, 4, 5 }, { 4, 4, 5 } };

// Xuất mảng 2 chiều
for(int i = 0; i < 3; i++) {
    for(int j = 0; j < 3; j++) {
        System.out.print(arr[i][j] + " ");
    }
    System.out.println();
}
```

CỘNG 2 MA TRẬN

// Tạo 2 mảng

```
int a[][] = { { 1, 3, 4 }, { 3, 4, 5 } };
```

```
int b[][] = { { 1, 3, 4 }, { 3, 4, 5 } };
```

1	3	4
3	4	5

 +

1	3	4
3	4	5

 =

2	6	8
6	8	10

// Tạo mảng lưu kết quả

```
int c[][] = new int[2][3];
```

// Cộng các phần tử và xuất ra màn hình

```
for(int i = 0; i < 2; i++) {  
    for(int j = 0; j < 3; j++) {  
        c[i][j] = a[i][j] + b[i][j];  
        System.out.print(c[i][j] + " ");  
    }  
    System.out.println();  
}
```

CHUỖI (STRING)

Khai báo chuỗi

❑ String là xâu các ký tự.

- Khai báo

```
String <ten_bien> [= tri_khoi tao];
```

- Ví dụ

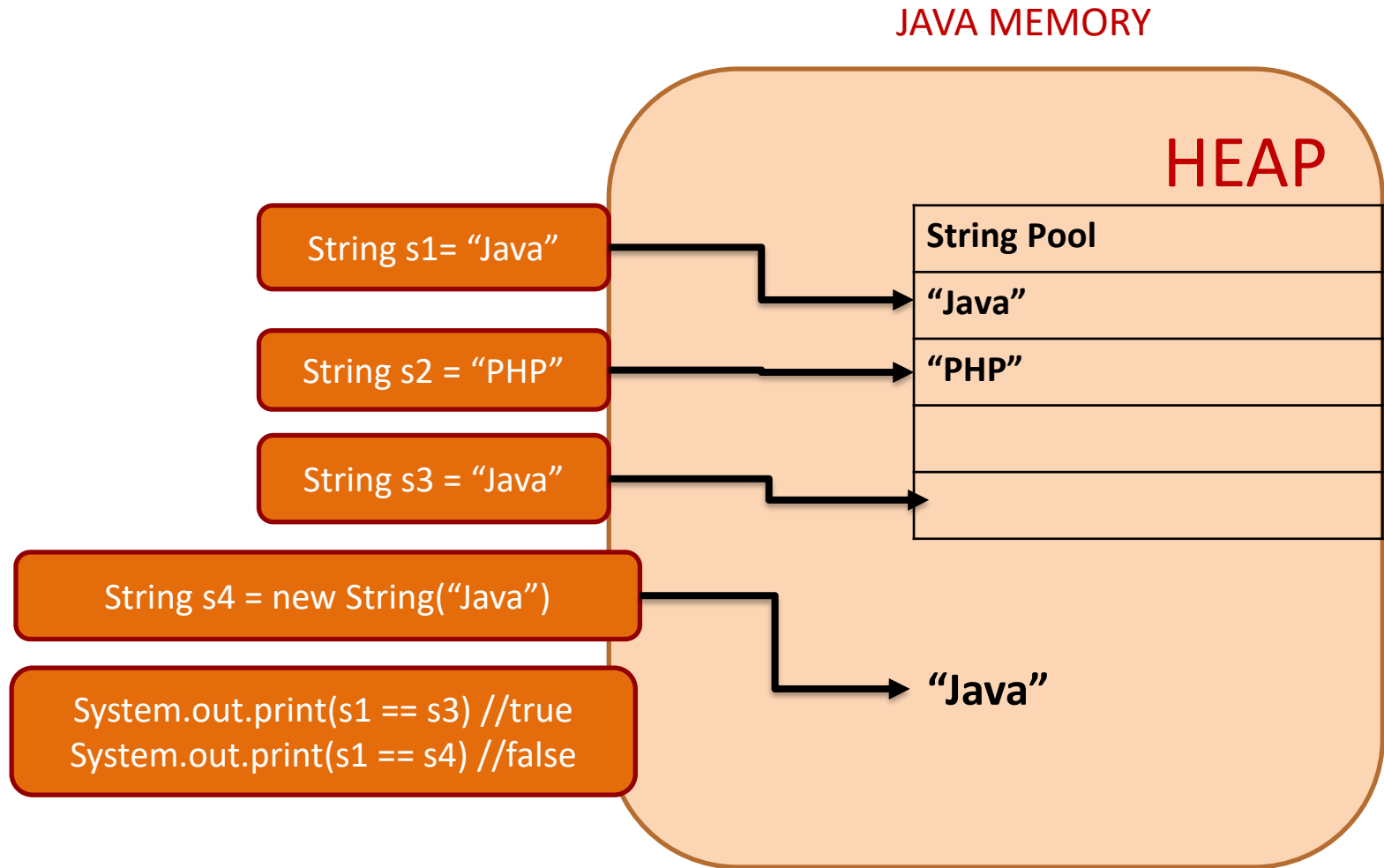
```
String s = "abc";
```

- Tránh sử dụng

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

- Chuỗi sử dụng phép gán trực tiếp được lưu giữ (interned) trong StringConstantPool và tái sử dụng khi giá trị giống nhau

Khai báo chuỗi



Một số lưu ý

- so sánh chuỗi sử dụng `.equals`

```
String s1 = "abc";  
System.out.println(s1.equals("abc"));  
//hoặc  
System.out.println("abc".equals(s1));
```

- Chuỗi là kiểu immutable (không thay đổi được). Các hàm biến đổi chuỗi đều sinh ra chuỗi mới và cần gán lại vào biến

```
String s = new String("  abc  ");  
//s không thay đổi  
s.trim();  
System.out.print(s); //"  abc  "  
//cần gán lại  
S = s.trim();  
System.out.print(s); //"abc"
```

Một số lưu ý

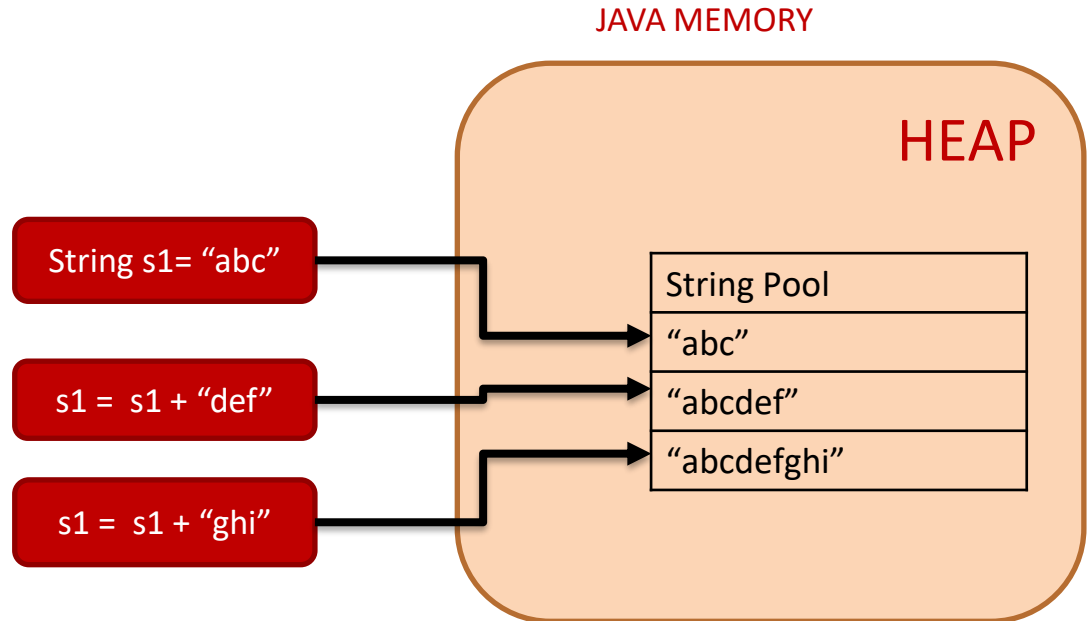
```
2 public class StringDemo3 {  
3  
4     public static void main(String[] args) {  
5         String s = "hello n02-k59";  
6         s = s.toUpperCase();  
7         System.out.println(s);  
8     }  
9 }  
10
```

String Pool
"hello n02-k59"
"HELLO N02-K59"

Nối chuỗi

- Sử dụng dấu +

```
String s1 = "An";  
System.out.println("My  
name is " + s1);
```



- Việc nối chuỗi sẽ sinh ra chuỗi mới

```
String s1 = "abc";  
s1 = s1 + "def";  
s1 = s1 + "ghi";  
System.out.println(s1);
```


Nối chuỗi – StringBuilder

- Nếu cần nối chuỗi dài nên sử dụng StringBuilder

```
StringBuilder sb = new  
StringBuilder("abc");  
sb.append("def");  
sb.append("ghi");  
  
System.out.println(sb.toString());  
;
```

```
StringBuilder sb = new  
StringBuilder("abc");  
sb.append("def");  
sb.append("ghi");
```

JAVA MEMORY

HEAP

String Pool

"abc"

"def"

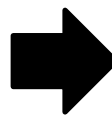
"ghi"

- Dùng StringBuilder khó đọc hơn cộng chuỗi thông thường nên cân nhắc dùng khi chuỗi dài

KÝ TỰ ĐẶC BIỆT

Ký tự	Hiển thị
\t	Ký tự tab
\r	Về đầu dòng
\n	Xuống dòng
\\	\
\"	"

```
System.out.print("\t+ Họ và tên: An Binh \r\n\t+ Tuổi: 20");
```



+ Họ và tên: An Binh
+ Tuổi: 20

LỚP STRING

- ❑ String là một class được xây dựng sẵn trong Java.
- ❑ String có rất nhiều phương thức giúp xử lý chuỗi một cách thuận tiện và hiệu quả.
- ❑ String là kiểu dữ liệu được sử dụng nhiều nhất trong lập trình

Các phương thức của STRING

Phương thức	Mô tả
s.toLowerCase ()	Đổi chuỗi s thành chữ in thường
s.toUpperCase ()	Đổi chuỗi s thành chữ in hoa
s.trim()	Cắt các ký tự trắng 2 đầu chuỗi s
s.length()	Lấy độ dài chuỗi s
s.substring(n,m)	Lấy chuỗi con của s từ vị trí a đến b
s.charAt (index)	Lấy ký tự tại vị trí index
s.replaceAll(find, replace)	Tìm kiếm và thay thế tất cả
s.split(separator)	Tách chuỗi thành mảng String s = "1,2,3,4,5,6,7"; String arr[]=s.split(",");

Các phương thức của STRING

Phương thức	Mô tả
<code>equals()</code>	So sánh bằng có phân biệt hoa/thường
<code>equalsIgnoreCase()</code>	So sánh bằng không phân biệt hoa/thường
<code>contains(String str)</code>	Kiểm tra có chứa chuỗi str hay không
<code>startsWith(String str)</code>	Kiểm tra có bắt đầu bởi chuỗi str hay không
<code>endsWith (String str)</code>	Kiểm tra có kết thúc bởi chuỗi str hay không
<code>matches ()</code>	So khớp
<code>indexOf(String str)</code>	Tìm vị trí xuất hiện đầu tiên của chuỗi con <code>s.indexOf(",")</code> //vị trí dau "," đầu tiên của chuỗi s
<code>lastIndexOf(String str)</code>	Tìm vị trí xuất hiện cuối cùng của chuỗi con
<code>String.valueOf(int a)</code>	Chuyển số thành chuỗi <code>int a = 10; String s = String.valueOf(a);</code>

- ❑ So sánh
- ❑ Tìm vị trí của chuỗi con
- ❑ Lấy chuỗi con
- ❑ Tách và hợp chuỗi
- ❑ Chuyển đổi hoa thường
- ❑ Lấy độ dài...

```
String fullname = "Nguyễn Văn Tý";  
String first = fullname.substring(0, 6);
```

↓
Nguyễn

Một số ví dụ

- Cắt chuỗi thành mảng các chuỗi con: `.split(String regex, int limit)`

```
//cắt chuỗi bởi dấu cách và tối đa 2 phần tử
String s = "myemail@gmail.com"
String[] sArr = s.split("@", 2);
System.out.print(sArr[0]); //myemail
System.out.print(sArr[1]); //gmail.com
```

- Nối chuỗi từ mảng: `String.join(CharSequence delimiter, CharSequence... elements)`

```
String s = String.join(" ", sArr);
```

- Loại bỏ khoảng trắng trước và sau chuỗi: `.trim()`

```
String s = "  abc  ";
String s = s.trim();
System.out.print(s); //"abc"
```

- Vị trí đầu tiên của chuỗi khác (bắt đầu từ 0): `.indexOf(String s)`

```
String s = "Hello world";
System.out.print(s.indexOf("world")); //6
```

Bài tập 1

- ❑ Đăng nhập hợp lệ khi mã tài khoản là "hello" và mật khẩu trên 8 ký tự
- ❑ Thực hiện:
 - ❖ Nhập username và password từ bàn phím
 - ❖ Sử dụng `equalsIgnoreCase()` để so sánh `username` và `length()` để lấy độ dài mật khẩu

```
if(username.equalsIgnoreCase("hello") && password.length() > 6){  
    ...  
}  
else{  
    ...  
}
```


❑ Quản lý sinh viên

- ❖ Nhập mảng họ tên sinh viên
- ❖ Xuất mảng họ và tên (IN HOA)
- ❖ Xuất danh sách sinh viên có tên được nhập từ bàn phím
- ❖ Xuất danh sách sinh viên có đệm hoặc tên là Thanh

❑ Gợi ý

- ❖ `fullname.toUpperCase()`: đổi IN HOA
- ❖ `fullname.startsWith("Nguyễn ")`: họ Nguyễn
- ❖ `fullname.endsWith(" Tuấn")`: tên Tuấn
- ❖ `fullname.contains(" Mỹ ")`: lót Mỹ
- ❖ `fullname.lastIndexOf(" ")`: Lấy vị trí trắng cuối cùng
- ❖ `fullname.substring(lastIndex + 1)`: Lấy tên

- ❑ Tìm kiếm và thay thế chuỗi

- ❑ Thực hiện theo hướng dẫn sau

- ❖ Nhập chuỗi nội dung, tìm kiếm và thay thế từ bàn phím

- String content = scanner.nextLine()

- String find = scanner.nextLine()

- String replace = scanner.nextLine()

- ❖ Thực hiện tìm và thay

- String result = content.**replaceAll**(find, replace)

VÍ DỤ 4

- ❑ Nhập chuỗi chứa dãy số phân cách bởi dấu phẩy và xuất các số chẵn ra màn hình
- ❑ Thực hiện
 - ❖ Sử dụng `split()` để tách chuỗi thành mảng bởi ký tự phân cách là dấu phẩy
 - ❖ Duyệt mảng, đổi sang số nguyên và kiểm tra số chẵn

```
String chuoi = "2,5,6,,8,9,12,13";  
String[] daySo = chuoi.split(",")  
for(String so : daySo){  
    int x = Integer.parseInt(so); if(x % 2 == 0){  
        System.out.println(x);  
    }  
}
```

- ❑ Tìm kiếm và thay thế chuỗi

- ❑ Thực hiện theo hướng dẫn sau

- ❖ Nhập chuỗi nội dung, tìm kiếm và thay thế từ bàn phím

- String content = scanner.nextLine()

- String find = scanner.nextLine()

- String replace = scanner.nextLine()

- ❖ Thực hiện tìm và thay

- String result = content.**replaceAll**(find, replace)

VÍ DỤ 4

- ❑ Nhập chuỗi chứa dãy số phân cách bởi dấu phẩy và xuất các số chẵn
- ❑ Thực hiện
 - ❖ Sử dụng `split()` để tách chuỗi thành mảng bởi ký tự phân cách là dấu phẩy
 - ❖ Duyệt mảng, đổi sang số nguyên và kiểm tra số chẵn

```
String[] daySo = chuoi.split("")
for(String so : daySo){
    int x = Integer.parseInt(so);
    if(x % 2 == 0){
        Số chẵn
    }
}
```

BIỂU THỨC CHÍNH QUI

❑ Bạn có biết các chuỗi sau đây biểu diễn những gì hay không?

❖ thuy@gmail.com

❖ 30A-771.61

❖ 0913745789

❖ 192.168.11.200

1. Bạn có biết tại sao bạn nhận ra chúng không?
2. Làm thế nào để máy tính cũng có thể nhận ra như bạn?

BIỂU THỨC CHÍNH QUI

- ❑ Máy tính có thể nhận dạng như chúng ta nếu chúng ta cung cấp qui luật nhận dạng cho chúng. Biểu thức chính qui cung cấp qui luật nhận dạng chuỗi cho máy tính.
- ❑ Biểu thức chính qui là một chuỗi mẫu được sử dụng để qui định dạng thức của các chuỗi. Nếu một chuỗi nào đó phù hợp với mẫu dạng thức thì chuỗi đó được gọi là so khớp (hay đối sánh).
- ❑ Ví dụ: **[0-9]{3,7}**: Biểu thức chính qui này so khớp các chuỗi từ 3 đến 7 ký tự số.
 - ❖ [0-9]: đại diện cho 1 ký tự số
 - ❖ {3,7}: đại diện cho số lần xuất hiện (ít nhất 3 nhiều nhất 7)

VÍ DỤ: BIỂU THỨC CHÍNH QUI

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

```
System.out.print("Số mobile: ");  
String mobile = scanner.nextLine();
```

```
String pattern = "0[0-9]{9,10}";
```

```
if(mobile.matches(pattern)){  
    System.out.println("Bạn đã nhập đúng số mobile");  
}  
else{  
    System.out.println("Bạn đã nhập không đúng số mobile");  
}
```

Biểu thức
chính qui

Kiểm tra mobile có so
khớp với pattern không?

s.matches(regex)

XÂY DỰNG BIỂU THỨC CHÍNH QUI

Regular Expression

Ký tự đại diện

[xyz]	đại diện một ký tự x, y hay z
[ad-f]	đại diện một ký tự a, d, e hay f
[^xyz]	đại diện ký tự không thuộc [xyz]
\d	tương đương [0-9]
\w	tương đương [0-9a-zA-Z_]
\D	tương đương [^\d]
\W	tương đương [^\w]
\s	đại diện ký tự trắng (\r\n\t\f)
.	đại diện ký tự bất kỳ
^	chỉ ra mẫu bắt đầu
\$	chỉ ra mẫu kết thúc
\\, \., \\$, \^	đại diện '\', '.', '\$' hay '^'

Số lần xuất hiện

{M,N}	Ít nhất M, nhiều nhất N lần
{N}	Đúng N lần
?	0-1
*	0-N
+	1-N
Không	1

[0-9]{3, 7}

BIỂU THỨC CHÍNH QUI THƯỜNG DÙNG

❑ Số CMND

❖ $[0-9]\{9\}$

❑ Số điện thoại di động việt nam

❖ $0\d{9}$

❑ Số xe máy sài gòn

❖ $5\d-[A-Z]\d-(\d{4})|(\d{3}\.\d{2}))$

❑ Địa chỉ email

❖ $\w+@\w+(\.\w){1,2}$

VÍ DỤ VỀ BIỂU THỨC CHÍNH QUI

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

```
System.out.print("Email: ");  
String email = in.nextLine();
```

```
System.out.print("Số điện thoại Huế: ");  
String phone = in.nextLine();
```

Email đơn giản

```
String reEmail = "\\w+@\\w+\\.\\w+";  
if(!email.matches(reEmail)) {  
    System.out.println("Không đúng dạng email !");  
}
```

Số điện thoại để bàn ở Huế

```
String rePhone = "0543\\d{6}";  
if(!phone.matches(rePhone)) {  
    System.out.println("Không phải số điện thoại ở Huế !");  
}
```

VÍ DỤ RÀNG BUỘC HỢP LỆ

- ❑ Nhập thông tin nhân viên từ bàn phím. Thông tin của mỗi nhân viên phải tuân theo các ràng buộc sau. Xuất thông báo lỗi và yêu cầu nhập lại

Thông tin	Kiểm soát	RegEx
Mã sinh viên	5 ký tự hoa	[A-Z]{5}
Mật khẩu	Ít nhất 6 ký tự	.{6,}
Họ và tên	Chỉ dùng alphabet và ký tự trắng	[a-zA-Z]+
Email	Đúng dạng email	\w+@\w+(\. \w+){1,2}
Điện thoại	Điện thoại Sài Gòn	083\d{7}
Số xe máy	Số xe máy Sài Gòn	5\d-[A-Z]-((\d{4}) (\d{3}\. {2}))
Số CMND	10 chữ số	\d{10}
Website	Địa chỉ website	http://www\.\w+\.\w{2,4}