



STRING, ARRAY & DATE

ThS. Nguyễn Nghiệm
0913.745.789 - NghiemN@fpt.edu.vn



NỘI DUNG

- String
- Array
- Date





- String
 - ✱ Quản lý chuỗi
- StringBuffer
 - ✱ Chuỗi thao tác được
- Regular Expression
 - ✱ Biểu thức chính qui





CHUỖI (STRING)

- String là chuỗi ký tự.
- Có nhiều cách tạo chuỗi:
 - ✱ `String s = "hello world";`
 - ✱ `String s = new String("hello world");`
 - ✱ `String s = new String(byte[] data)`
 - ✱ `String s = new String(char[] data)`
 - ✱ `String s = "" + 123;`
 - ✱ `String s = String.valueOf(123);`
- String là kiểu dữ liệu được sử dụng nhiều nhất trong lập trình



KÝ TỰ ĐẶC BIỆT

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

+ Họ và tên: Nguyễn Nghiêm
+ Tuổi: 90



```
String name = "Nguyễn Nghiêm";  
int age = 90;  
System.out.printf("\t+ Họ và tên%s\r\n\t+ Tuổi: %d", name, age);
```



THAO TÁC CHUỖI

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

```
String fullname = “Nguyễn Văn Tèo”;  
String first = fullname.substring(0, 6);
```



STRING API

Phương thức	Mô tả
toLowerCase ()	Đổi in thường
toUpperCase ()	Đổi in hoa
trim()	Cắt các ký tự trắng 2 đầu chuỗi
length()	Lấy độ dài chuỗi
substring()	Lấy chuỗi con
charAt ()	Lấy ký tự tại vị trí
replaceAll()	Tìm kiếm và thay thế tất cả
replaceFirst()	Tìm và thay 1 lần
split()	Tách chuỗi thành mảng



STRING API

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()</code>	Kiểm tra có chứa hay không
<code>startsWith()</code>	Kiểm tra có bắt đầu bởi hay không
<code>endsWith ()</code>	Kiểm tra có kết thúc bởi hay không
<code>matches ()</code>	So khớp với hay không?
<code>indexOf()</code>	Tìm vị trí xuất hiện đầu tiên của chuỗi con
<code>lastIndexOf()</code>	Tìm vị trí xuất hiện cuối cùng của chuỗi con
<code>getBytes ()</code>	Đổi sang mảng <code>byte[]</code>



THỰC HÀNH CHUỖI 1

- Đăng nhập hợp lệ khi mã tài khoản là “hello” và mật khẩu trên 6 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{  
    ...  
}
```



THỰC HÀNH CHUỖI 2

● Quản lý sinh viên

- ✱ Nhập mảng họ tên sinh viên
- ✱ Xuất họ và tên (IN HOA) những sinh viên tên Tuấn hoặc họ Nguyễn
- ✱ Xuất tên những sinh viên có tên lót là Mỹ

● Thực hiện

- ✱ 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



THỰC HÀNH CHUỖI 3

- 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)



THỰC HÀNH CHUỖI 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  
    }  
}
```



HÀM TIỆN ÍCH CHUỖI

- **String.format(format, args...)**
 - ✱ Định dạng chuỗi. Các ký tự định dạng gồm %s, %d, %f tương ứng là chuỗi, số nguyên và số thực
- **String.join(array, seperator)**
 - ✱ Kết nối các phần tử của mảng thành chuỗi
- **String.valueOf(object)**
 - ✱ Chuyển đổi giá trị của kiểu bất kỳ sang chuỗi



STRING.FORMAT()

Mã nguồn

```
String name = "Mỹ Lệ";  
int age = 40;  
double salary = 500.8;  
String format = "Lương của %s (%d tuổi) là %.3f đô la";  
String s = String.format(format, name, age, salary)
```



Kết quả

Lương của Mỹ Lệ (40 tuổi) là 500.800 đô la



STRING.JOIN() & STRING.VALUEOF()

```
String[] mang = {"Tuấn", "Cường", "Hồng"};  
String chuoi = String.join("~", mang);  
System.out.print(chuoi);
```

↳ Tuấn~Cường~Hồng

```
String s = String.valueOf(2) + String.valueOf(true);  
System.out.print(s);
```

↳ 2true



STRING & STRINGBUFFER

- String không cho phép thay đổi nội dung chuỗi. Tất cả các thao tác với String sẽ cho kết quả là một chuỗi khác (chuỗi gốc không đổi).
- StringBuffer cho phép thao tác trực tiếp lên nội dung chuỗi gốc.

```
String[] array = {"Tuấn", "Hồng", "Phương"};
```

```
String names = "";  
for(String name: array){  
    names += name;  
}
```

```
String[] array = {"Tuấn", "Hồng", "Phương"};
```

```
StringBuffer names = new StringBuffer();  
for(String name: array){  
    names.append(name);  
}
```




STRINGBUFFER API

Phương thức	Mô tả
append(String s)	Bổ sung
insert(int index, String s)	Chèn thêm
delete(int start, int end)	Xóa bớt
setCharAt(int index, char c)	Thay đổi ký tự
setLength(int length)	Thay đổi độ dài chuỗi
reverse()	Đảo các ký tự trong chuỗi



BIỂU THỨC CHÍNH QUI

● Hãy cho biết những thành phần sau đây là gì?

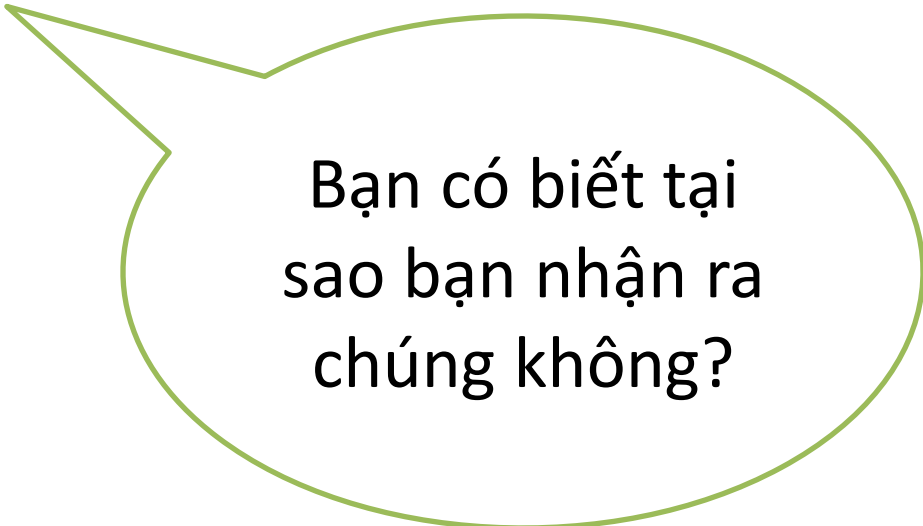
✱ user@abc.com

✱ 54-P6-6661

✱ 54-P6-666.01

✱ 0913745789

✱ 192.168.11.200



Bạn có biết tại
sao bạn nhận ra
chúng không?



BIỂU THỨC CHÍNH QUI

- Máy tính có thể nhận dạng như chúng ta nếu chúng ta cung cấp qui luật nhận dạng cho chúng. BTCQ cung cấp qui luật nhận dạng chuỗi cho máy tính.
- BTCQ 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** (đối sánh).
- Ví dụ: **[0-9]{10, 11}**: BTCQ này so khớp các chuỗi 10 hoặc 11 ký tự số



KHỞI ĐỘNG CÙNG REGEX

```
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");  
}
```

Regular
Expression

Kiểm tra mobile có so
khớp với pattern không?



BIỂU THỨC CHÍNH QUI

Biểu thức chính quy

0[0-9]

{9,10}

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



CÁC REGEX THƯỜNG DÙNG

- Số CMND

`[0-9]{9}`

- Số điện thoại di động việt nam

`0\d{9,10}`

- Số xe máy sài gòn

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

- Địa chỉ email

`\w+@\w+(\.\w{2,4}){1,2}`



BIỂU THỨC CHÍNH QUI

- Java cung cấp các phương thức làm việc với RegEx

Phương thức	Mô tả
boolean matches (String regex)	Kiểm tra một chuỗi có khớp với regex không
String replaceAll (String regex , String replace)	Tìm kiếm và thay thế các chuỗi khớp với regex bằng replace
String replaceFirst (String regex , String replace)	Tìm kiếm và thay thế các chuỗi đầu tiên khớp với regex bằng replace
String[] split (String regex)	Tách chuỗi thành mảng bởi chuỗi phân cách khớp với regex



VÍ DỤ VỀ REGEX

```
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 !");  
}
```




Kiểm soát dữ liệu với BTCQ



THỰC HÀNH - VALIDATION

- 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]+
Ngày sinh	Dạng ngày-tháng-năm	\d{2}-\d{2}-\d{4}
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}



MẢNG

- Mảng là gì
- Duyệt mảng với for-each
- Thư viện tiện ích Arrays
- Thuật toán sắp xếp đơn giản





MẢNG LÀ GÌ

- Mảng là một cấu trúc lưu trữ các thành phần có cùng kiểu

0	1	2	3	4	5	6	7	8	← Indices
5	7	9	1	45	1	9	9	2	← Elements

- Sử dụng một biến mảng để quản lý nhiều giá trị
- Tổ chức dữ liệu giúp cho chúng ta thao tác dễ dàng hơn
- Thao tác và kiểm duyệt các phần tử rất nhanh
- Thư viện Arrays cung cấp các phương thức tiện ích xử lý mảng một cách thuận lợi



KHAI BÁO MẢNG

● Khai báo không khởi tạo

✱ `int[] a;` // mảng số nguyên chưa biết số phần tử

✱ `int b[];` // mảng số nguyên chưa biết số phần tử

✱ `String[] c = new String[5];` // mảng chứa 5 chuỗi

● Khai báo có khởi tạo

✱ `double[] d1 = new double[]{2, 3, 4, 5, 6};` // mảng số thực, 5 phần tử, đã được khởi tạo

✱ `double[] d2 = {2, 3, 4, 5, 6};` // mảng số thực, 5 phần tử, đã được khởi tạo



TRUY XUẤT CÁC PHẦN TỬ

- Sử dụng **chỉ số (index)** để phân biệt các phần tử.
Chỉ số mảng tính từ 0.
 - ✱ `int a[] = {5,7,9,1,45,1,9,9,2};`
 - ✱ `a[2] = a[4] * 4; // 45*4=180`
 - ✱ Sau phép gán này mảng là {5,7,**180**,1,45,1,9,9,2};
- Sử dụng thuộc tính **length** để lấy số phần tử của mảng
 - ✱ `a.length` có giá trị là 9



DUYỆT MẢNG

- Vòng lặp kiểu for-each sẽ duyệt lần lượt các phần tử trong tập hợp
- Cú pháp

```
for(Type element : array){  
    Xử lý element  
}
```





DUYỆT MẢNG

- Ví dụ sau tính tổng các số chẵn của mảng.

- ✱ Lấy từng phần tử từ mảng với for-each

- ✱ Nếu là số chẵn thì cộng vào tổng

```
int[] a = {9, 3, 8, 7, 3, 9, 4, 2};  
  
double tong = 0;  
for(int x : a){  
    if(x % 2 == 0){  
        tong += x;  
    }  
}  
  
System.out.print("Tổng: " + tong);
```




Nhập mảng số nguyên và
tính trung bình cộng.



THAO TÁC MẢNG


```
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);



VÍ DỤ THAO TÁC MẢNG

```
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]
```



Nhập 5 SV và xuất theo thứ tự
alphabet



THUẬT TOÁN SẮP XẾP

- Arrays.sort(mảng) không thể thực hiện
 - ✱ Sắp xếp giảm
 - ✱ Các kiểu không so sánh được
- Giải pháp: tự xây dựng thuật toán sắp xếp

```
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.



Nhập 2 mảng họ tên và điểm. Xuất 2 mảng theo thứ tự giảm của điểm



XỬ LÝ THỜI GIAN

- **Date**

- ✱ Quản lý các thông số thời gian

- **SimpleDateFormat**

- ✱ Chuyển đổi giữa thời gian và chuỗi





THỜI GIAN

- Date là kiểu dữ liệu chứa các thông số về thời gian như: ngày, tháng, năm, giờ, phút, giây, tuần, quý, sáng chiều...
- Tạo Date
 - ✱ `Date now = new Date(time)`
 - Trong đó time được tính bằng mili giây mốc 1/1/1970
 - ✱ `Date now = new Date()`
 - Thời gian hiện tại của máy tính
- `SimpleDateFormat` là lớp được sử dụng để chuyển đổi giữa String và Date

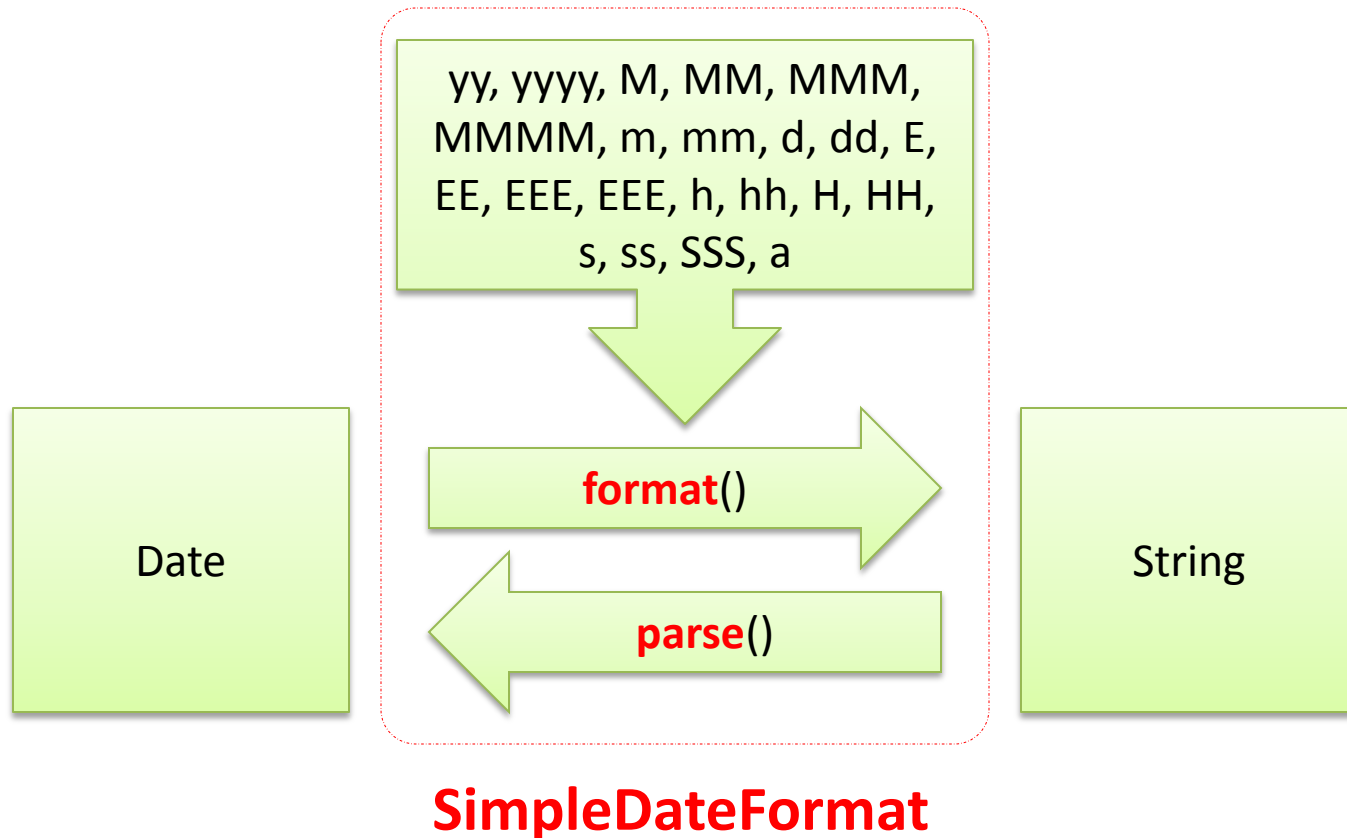


DATE API

Phương thức	Mô tả
boolean after (Date date)	So sánh lớn hơn
boolean before (Date date)	So sánh nhỏ hơn
Object clone ()	Nhân bản (sao chép)
long getTime ()	Lấy thời gian (mili giây) tính từ 1/1/1970
void setTime (long time)	Thay đổi thời gian



CHUYỂN ĐỔI DATE <=> STRING





KÝ TỰ ĐỊNH DẠNG

Ký tự	Mô tả	Ví dụ
yy, yyyy	Năm 2, 4 chữ số	2015
M, MM, MMM, MMMM	Tháng 1, 2 chữ số hoặc tên tháng viết tắt hoặc đầy đủ	July or 07
d, dd	Ngày trong tháng	10
h, hh, H, HH	Giờ 1, 2 chữ số dạng 12(h) hoặc 24(H)	12
m, mm	Phút 1,2 chữ số	30
s, ss	Giây 1, 2 chữ số	55
S	Mili giây	234
E, EE, EEE, EEEE	Ngày trong tuần	Tuesday
D	Ngày trong năm	360
w	Tuần trong năm	40
W	Tuần thứ mấy trong tháng	1
a	Buổi (sáng, chiều)	PM



VÍ DỤ

```
SimpleDateFormat df = new SimpleDateFormat();

try {
    String ngay = "20-10-2015";
    df.applyPattern("dd-MM-yyyy");
    Date d = df.parse(ngay);
}
catch(Exception ex) {
    System.out.println("Sai định dạng ngày");
}

Date now = new Date();
df.applyPattern("MMMM dd, yyyy");
String ngay = df.format(now);

System.out.print(ngay);
```



Nhập ngày và xuất tuổi



TÓM TẮT

● String

- ✱ String là chuỗi ký tự, được sử dụng nhiều nhất trong lập trình
- ✱ StringBuffer là chuỗi cho phép thay đổi nội dung chuỗi gốc
- ✱ Regular Expression là mẫu so khớp chuỗi

● Array

- ✱ Mảng là tập hợp các phần tử cùng kiểu
- ✱ Duyệt mảng với for-each
- ✱ Thư viện tiện ích Arrays
- ✱ Thuật toán sắp xếp đơn giản

● Date

- ✱ Thời gian
- ✱ Chuyển đổi thời gian và chuỗi

