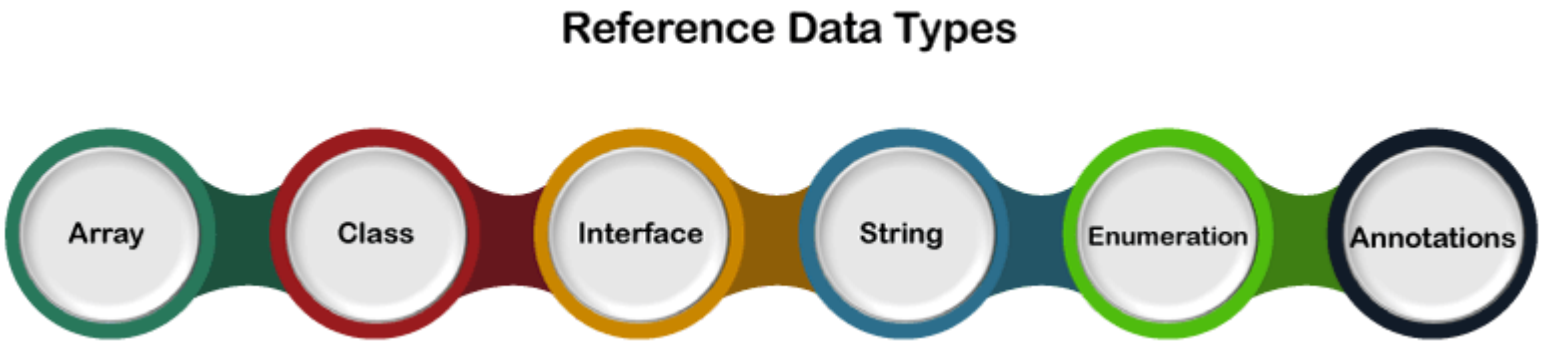
GIỚI THIỆU VỀ JAVA

REFERENCE DATA TYPES



GVHD: ThS. Hồ Khôi
Email: khoihue@gmail.com

T9

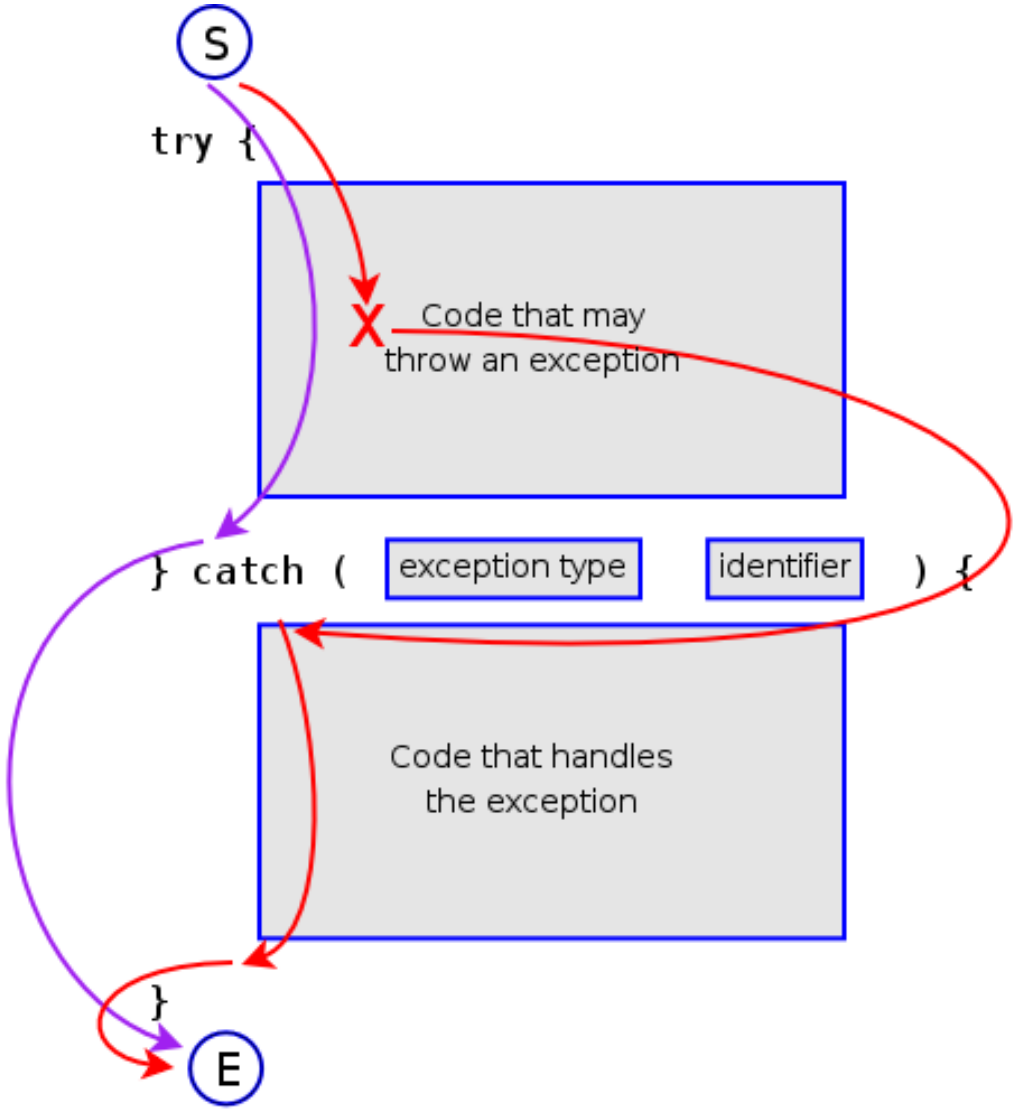
Safety code


Make it short!
WE NEED TO GET IN TOO!
I'll notify them and get back
IT WILL TAKE JUST ONE MOMENT
SYNCRONIZED
SYNCRONIZED

Tại sao phải "safety code" ?

- Mọi việc, không phải lúc nào cũng như dự tính
- Có những vấn đề bất thường xảy ra vào thời điểm thực thi chương trình "runtime-error"
- Có "*những lỗi*" có thể dự tính, nhưng lại không thể xử lý bằng các cấu trúc lệnh thông thường
- Cần phải có một cấu trúc để hỗ trợ cho các tình huống riêng biệt

Cấu trúc try - catch

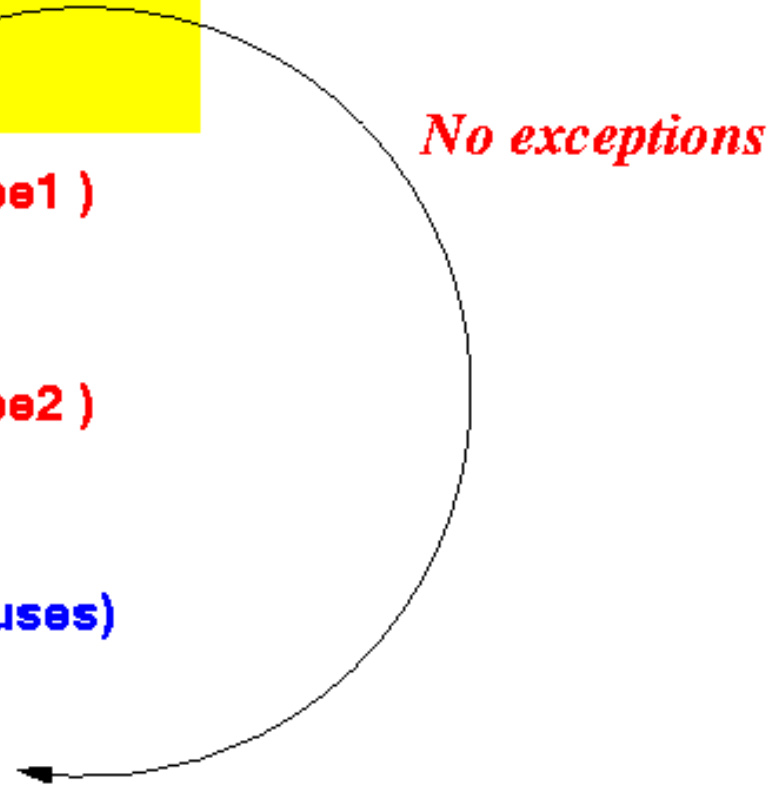


❖Những phần việc dùng để xử lý khi mọi việc hoàn toàn theo xu hướng phù hợp với dự tính của người lập trình **try{…}**

❖Những tình huống đặc biệt, bất thường có thể xảy ra trong quá trình chương trình thi hành **catch () { … }**

Cấu trúc try - catch

```
try
{
   statement1;
   statement2;
    . . .
}
catch ( exceptionType1 )
{
    statements...
}
catch ( exceptionType2 )
{
  statements...
}
 . . . (more catch-clauses)
```

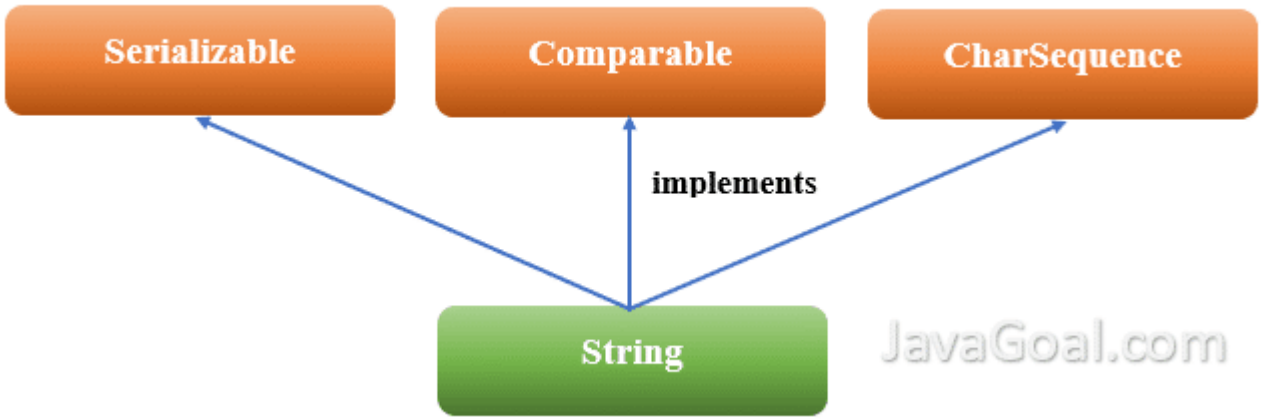


- Một try có thể có nhiều catch để xử lý nhiều loại Exception khác nhau.

  VD:

  - ✓ Lỗi liên quan đế đường truyền mạng,
  - ✓ lỗi liên quan đến xử lý dữ liệu
  - ✓ Lỗi liên quan đến thiết bị lưu trữ
  - ✓ Lỗi bộ nhớ
  - ✓ …

String class



Dữ liệu chuỗi – String

Chuỗi là tập hợp các ký tự được cấp phát tuần tự, liên tục trong bộ nhớ. Đây là loại dữ liệu sử dụng phổ biến trong chương trình. Trong Java, chuỗi được xem là một loại "***đối tượng***", do đó, java cung cấp một class cùng với các phương thức phục vụ cho việc xử lý đối với dữ liệu thuộc loại này với tên "**String**" class thuộc namespace: **java.lang**

Có thể khai báo để sử dụng chuỗi theo một trong các cách sau:

+ Khai báo trực tiếp

```
String tenTruong = "Nguyen Tat Thanh";
```

+ Khai báo để chuyển 1 chuỗi từ mảng ký tự

```
char data[] = {'N', 'g', 'u', 'y', 'e', 'n', ' ', 'T', 'a', 't', ' ', 'T', 'h', 'a', 'n', 'h'};
String str = new String(data);
```

N	g	u	y	e	n		T	a	t		T	h	a	n	h

Length = ? ; vị trí đầu tiên ? ; vị trí thứ 7 ?; vị trí thứ 12 ? ;

Thư viện

- Nhập - Xuất chuỗi

  Có thể sử dụng phương thức **nextLine** của đối tượng scanner như sau

  VD: *Khai báo biến hoTen chứa chuỗi đồng thời tiến hành nhập dữ liệu cho biến này ta làm như sau*

```
Scanner nhap = new Scanner(System.in);
System.out.print("Ho va ten:\t");
String hoTen = nhap.nextLine();
```

- Sử dụng phương thức print hoặc println để in nội dung của 1 đối tượng String ra màn hình giống như các loại dữ liệu khác (*Có thể sử dụng kết hợp với* **escape character**)

  VD: *Xuất nội dung của biến hoTen lên màn hình, kết hợp với* **place holder** *cùng* **escape character** *ta làm như sau*

```
System.out.print("Sinh viên: %s \t chuyên ngành CNTT \n Lớp: 18DTH2D ", hoTen);
```

Hàm liên quan đến chuỗi

- **So sánh chuỗi**

  public boolean equals(Object anObject)

  public boolean equalsIgnoreCase(String anotherString)

  public boolean startsWith(String prefix)

  public boolean endsWith(String suffix)

- **Chuyển đổi**

  public String toUpperCase()

  public String toLowerCase()

  public String length()

- **Cắt chuỗi**

  public String trim()

  public String substring(int beginIndex, int endIndex)

  public String[] **split**(String regex, int limit)

Ví dụ minh hoạ

- Cho một số ví dụ liên quan đến việc so sánh:
  - ❖ **equals**
  - ❖ **startsWith**
  - ❖ **endsWith**
- Ví dụ liên quan đến việc ứng dụng **split** 1 chuỗi
- Ví dụ liên quan đến: **LowerCase**, **UpperCase**, **Trim** và **substring**

Wrapper class & chuyển đổi chuỗi

- Chuyển đổi "***chuỗi số***" thành "***số nguyên, số thực***"

  Trong những trường hợp nhất định, khi dữ liệu nhận được là một chuỗi số. Cần phải chuyển "***chuỗi số***" này thành "***số***",

  VD: Số thực: ***float***, ***double***, số nguyên: ***byte***, ***int***, ***short***, ***long***. Java cung cấp các Wrapper class có tên trùng với tên của primitive data type tương ứng, với phương thức chuyển đổi có tiền tố parse để cho phép bạn thực hiện yêu cầu trên.

- Các Wrapper class tương ứng với các kiểu dữ liệu cơ sở của Java

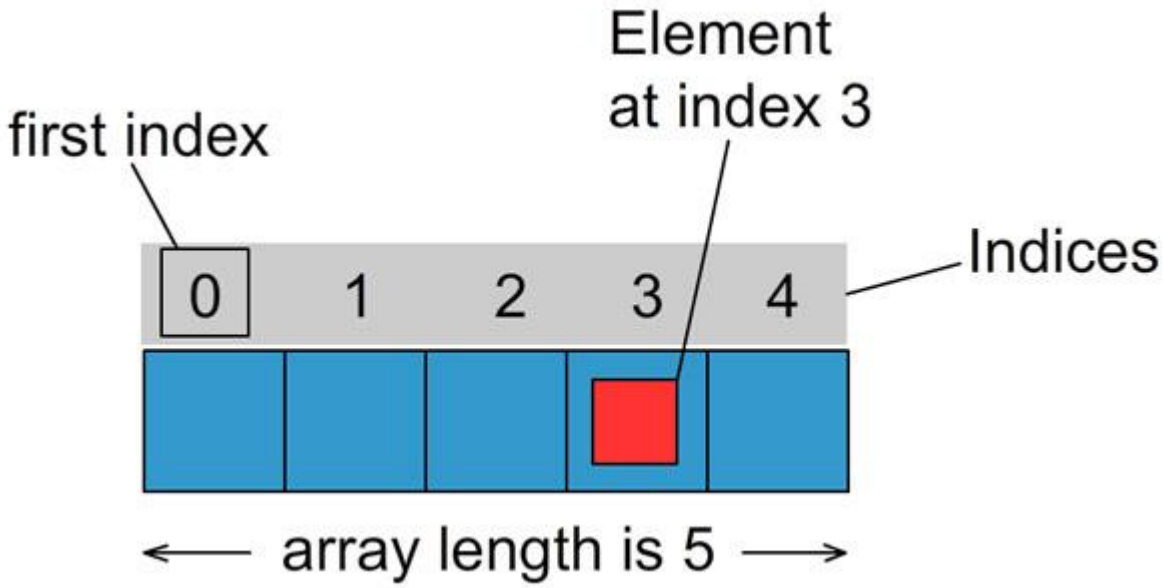
  - **Integer** ⇔ **int** ⇔ ***.parseInt***("chuỗi số")
  - **Long** ⇔ **long** ⇔ ***.parseLong***("chuỗi số")
  - **Byte** ⇔ **byte** ⇔ ***.parseByte***("chuỗi số")
  - **Short** ⇔ **short** ⇔ ***.parseShort***("chuỗi số")
  - **Float** ⇔ **float** ⇔ ***.parseFloat***("chuỗi số")
  - **Double** ⇔ **double** ⇔ ***.parseDouble***("chuỗi số")

Wrapper class & chuyển đổi chuỗi

- Yêu cầu sinh viên thực hiện các minh họa khi thực hiện quá trình chuyển đổi chuỗi số thành số ở các dạng (*Chỉ cần viết một số lệnh*)
  - **Số nguyên : byte, short, int, long**
  - **Số thực : float, double**

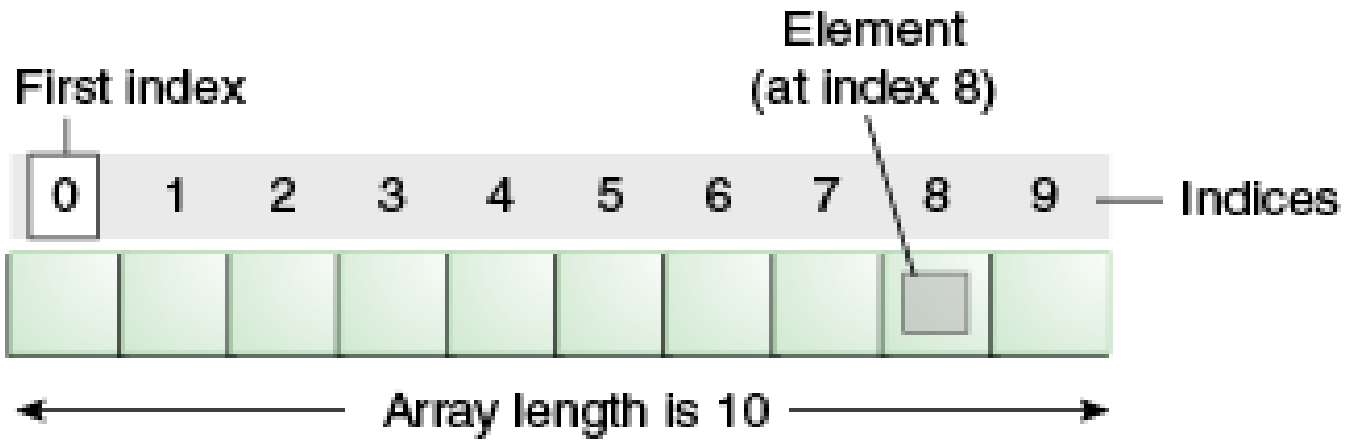
- Cho một ví dụ minh họa dựa trên việc nhập chuỗi, sau đó thực hiện chuyển đổi và tính toán, sau đó xuất dữ liệu ra màn hình. (*Phải viết 1 chương trình hoàn chỉnh*)

Array


Element
at index 3
first index
Indices
0
1
2
3
4
array length is 5

Giới thiệu

- Mảng là một cấu trúc dữ liệu đơn giản trong ngôn ngữ lập trình, dung để lưu trữ những đôi tượng dữ liệu cùng kiểu, cùng loại và cùng mục đích. Khi được khai báo, mảng được cấp phát một cách tuần tự, liên tục trong bộ nhớ
- Đặc điểm đầu tiên cần ghi nhớ là: màng lưu trữ một tập các giá trị cùng kiểu
  - ✓ Cùng kiểu "*nguyên thủy*" (int, double, etc.) or
  - ✓ Cùng loại đối tượng (***SinhVien***, ***HangHoa***, ***NhanVien***, ***PhuongTienGiaoThong***, ...)



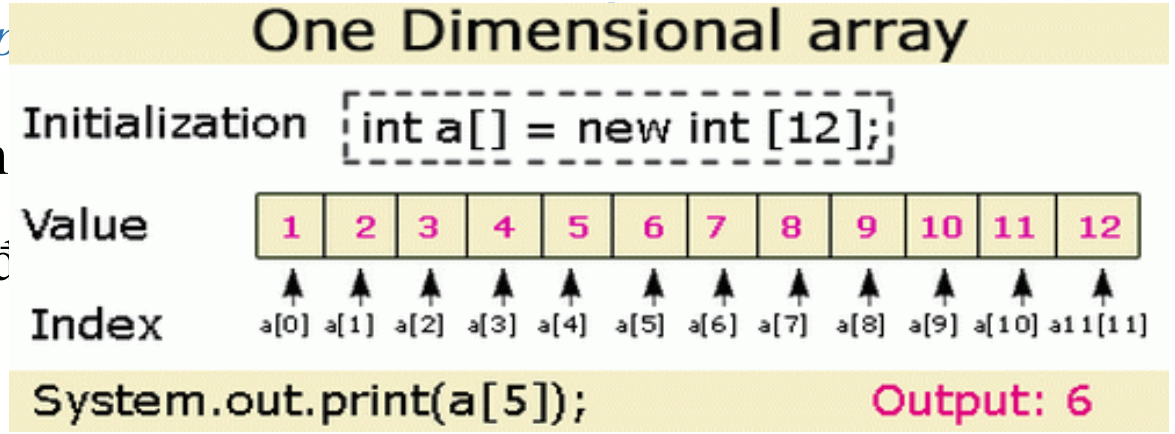
Array

- Trong ngôn ngữ lập trình Java, mảng là một phần quan trọng trong lưu trữ dữ liệu (*trên bộ nhớ*) & quá trình xử lý thuật toán. Một số điểm cần lưu ý khi làm việc với mảng trong Java
  - Mảng là **objects**, không phải "*dữ liệu nguyên thủy - primitives data*" giống như **int** hay **double**. Do đó, khi cần cấp phát 1 mảng trong chương trình, bạn có thể sử dụng cú pháp để tạo object. VD

    int a[]= new **int**[12]; *//--- Cấp*

  - Ở ví dụ trên, a là một object chứa các phần "*chiều dài – length*" , do đó, ta có thể xác đ length như sau:

    int chieuDaiMangA = a.***length***; *//---- chieuDaiMangA = 12*



Array

- Mỗi phần tử của mảng có thể truy xuất dựa trên chỉ số của chúng. VD, để gán giá trị cho phần tử đầu tiên & phần tử thứ 2 của mảng a, ta có thể thực hiện

  a[0]= 4; a[1]= 77;

- Chỉ số của phần tử đầu tiên của mảng luôn bắt đầu là 0, (*Không phải 1*). Như vậy, chỉ số của các phần tử thuộc mảng được mô tả như sau
  - Với 1 mảng có N phần tử thì "*chỉ số* - *index*" của chúng được xác định từ 0 đến N-1
  - Theo cách khác, ta nói: Mảng a có các phần tử a[0], a[1], … a[11] (*Ở VD trên, a có 12 phần tử*)

Array

- Chiều dài, hay kích thước của 1 mảng không thể bị thay đổi sau khi đã khai báo. VD: mảng a đã khai báo có 12 phần tử, thì không thể thay đổi kích thước, cho đến khi bị loại khỏi bộ nhớ.
- Có thể khai báo đồng thời khởi gán giá trị cho các phần tử của mảng theo cú pháp sau

```
int intArray [] = {5, 77, 4, 9, 28, 0, -9};
```

*//--- Trong trường hợp này, không cần sử dụng từ khóa **new***

- Trong trường hợp cấp phát mảng mà không chỉ ra giá trị khởi tạo (*giống như ví dụ trước*) thì các phần tử của mảng mặc nhiên sẽ được gán =0

```
int a [] = new int[12];
```

//--- Cấp phát 1 mảng a có 12 phần tử, giá trị của mỗi phần tử =0

Copying Arrays

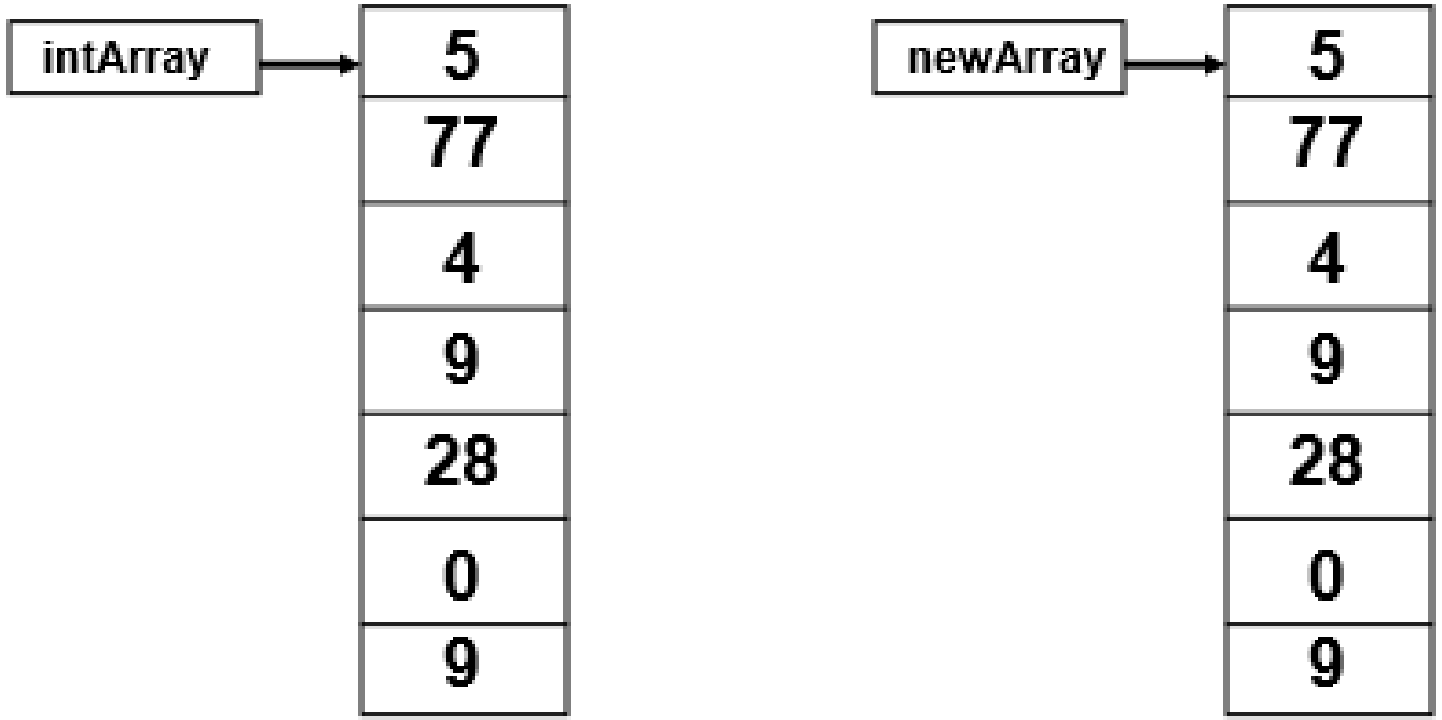
- Để sao chép 1 mảng, ta sử dụng phương thức ***arraycopy***() của System class, theo cú pháp như sau:

```
public static void arraycopy(Object src, int srcPos,
                             Object dest, int destPos,
                             int length)
```

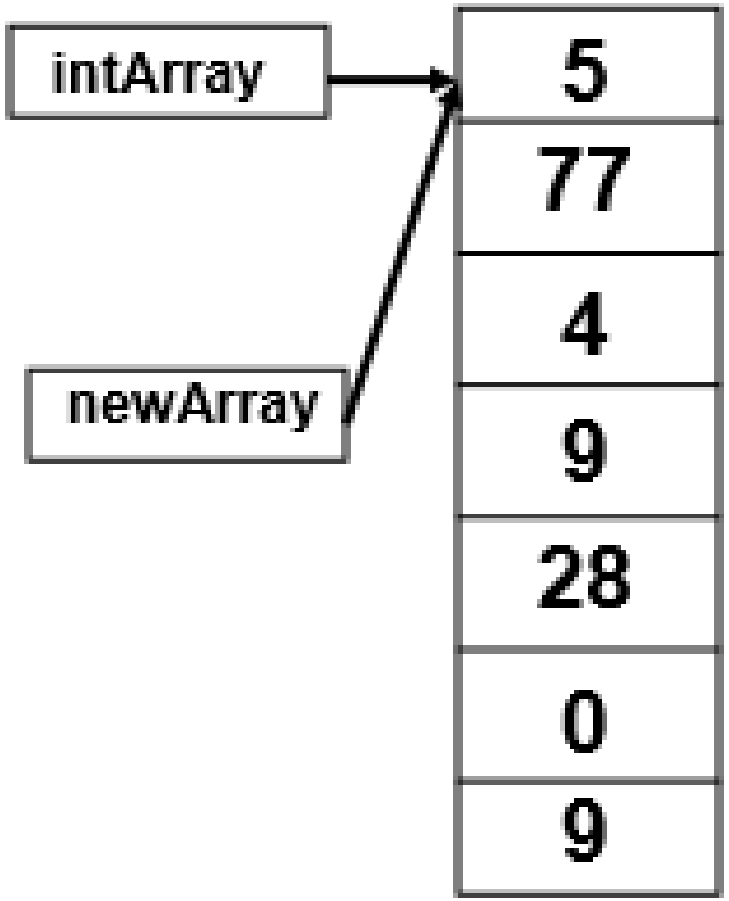
  - **src** – Mảng được xem là ***nguồn*** cho quá trình sao chép.
  - **srcPos** – Vị trí của phần tử bắt đầu sao chép (Index).
  - **dest** – Mảng được xem là ***đích*** của quá trình sao chép (*Mảng sẽ được tạo thành*).
  - **destPos** – Vị trí của phần tử bắt đầu nhận dữ liệu tại "*mảng đích*" (Index).
  - **length** – Số lượng phần tử sẽ sao chép.

Copy entire array

```
int[ ] intArray= {5, 77, 4, 9, 28, 0, 9};
int[ ] newArray = new int[intArray.length];
System.arraycopy(intArray,0,newArray,0,intArray.length)
```



Copy reference




```
int[ ] intArray= {5, 77, 4, 9, 28, 0, 9};
int[ ] newArray = intArray;
```

Cấu trúc For-Each của java

- Để duyệt trên một tập các phần tử dữ liệu. Java cung cấp cấu trúc For-Each để sử dụng theo cú pháp như sau:

for (<kiểu_dữ_liệu> <biến> : <tập_dữ_liệu>)

<lệnh>;

- **VD**: Để in ra tất cả các phần tử có trong mảng số nguyên ta có thể dùng for-each như sau:

```
int[] mangSN = { 1, 2, 3, 4, 25, 79, 113, 115, 116};


System.out.println("Mang cac phan tu cua mangSN: ");
for (int x  : mangSN) {
        System.out.print(x + " ");
}
```

Ví dụ minh họa

- Cho ví dụ minh họa đến việc nhập, xử lý, xuất dữ liệu trên các mảng:
  - ✓ **byte**
  - ✓ **int**
  - ✓ **float**
- **Lưu ý:**

  Xử lý chỉ nên yêu cầu thực hiện ở mức đơn giản.

  VD: Đếm số phần tử; Tính tổng; Tính tích; Trung bình cộng.

Tổng kết

- Tại sao phải cần đến cấu trúc **try - catch** trong quá trình xây dựng mã lệnh cho chương trình?
- Chuỗi & các phương thức liên quan đến xử lý dữ liệu chuỗi?
- Mảng dùng cho mục đích gì. Đọc, ghi dữ liệu mảng thế nào?
- Cần lưu ý gì khi thao tác với đối tượng mảng trong chương trình?

Tài liệu tham khảo

- **Class String**

  (https://docs.oracle.com/javase/9/docs/api/java/lang/String.html)

- **Arrays, The Java™ Tutorials**

  (https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html)

- **The For-Each Loop**

  (https://docs.oracle.com/javase/8/docs/technotes/guides/language/foreach.html)

- **Java - Strings Class**

  (https://www.tutorialspoint.com/java/java_strings.htm)

THANK YOU VERY MUCH

Q/A: khoihue@gmail.com