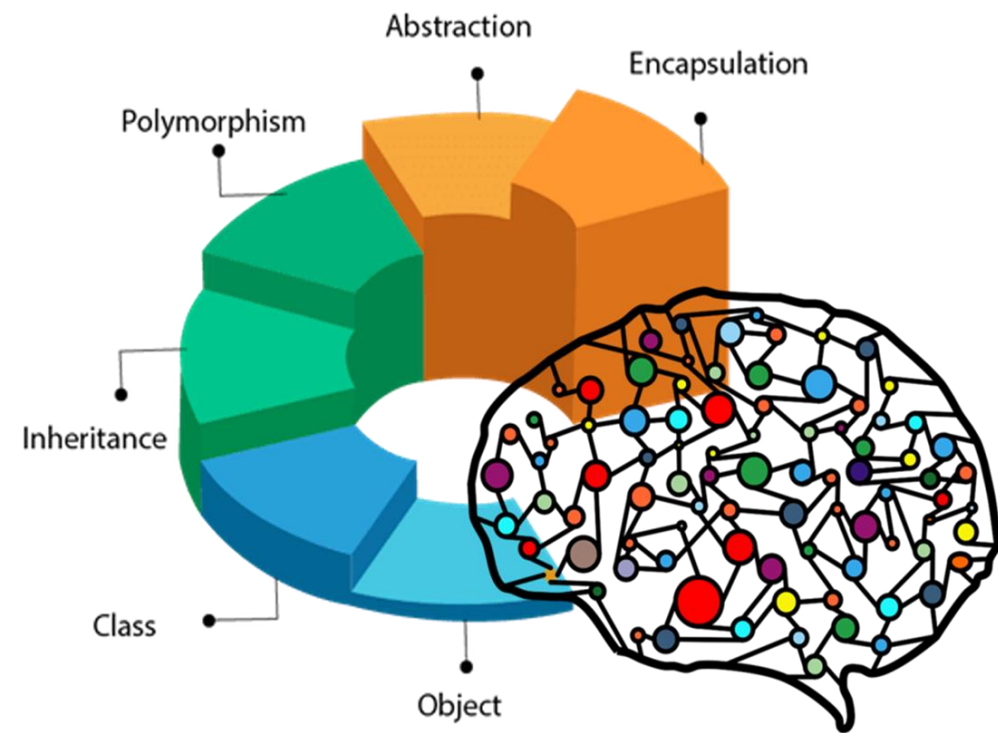


KỸ THUẬT LẬP TRÌNH

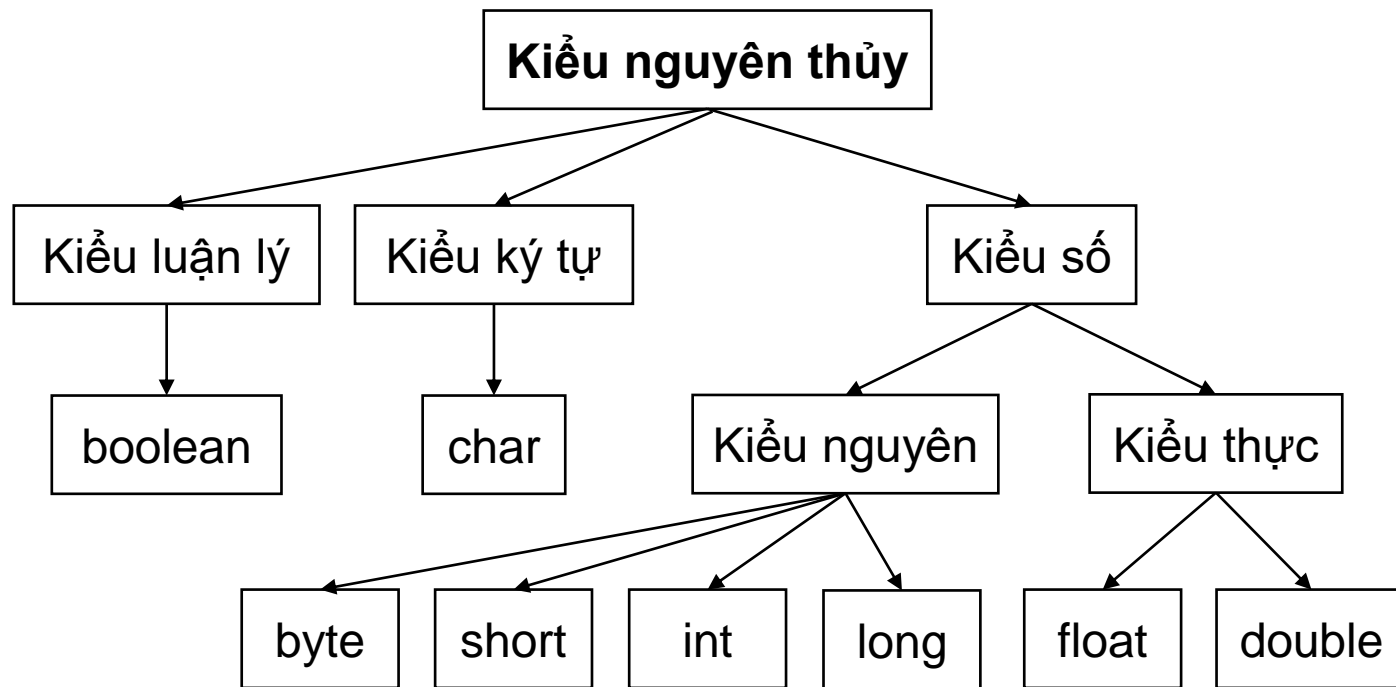
ĐẶNG VĂN NGHĨA
0975079414
nghiadv@donga.edu.vn

1. Kiểu dữ liệu
2. Biến (Variable)
3. Hằng (Constant)
4. Kiểu enum
5. Mảng (Array)
6. Hàm và chương trình
7. Biến, mảng tự động
8. Biến, mảng ngoài
9. Biến tĩnh, mảng tĩnh



1. KIỂU DỮ LIỆU

- ❖ **Kiểu dữ liệu trong Java:** chia thành 2 loại chính
 - Kiểu dữ liệu nguyên thủy (Primitive Data Types)
 - Kiểu dữ liệu tham chiếu (Reference Types)



Kiểu dữ liệu	Mô tả
Mảng (Array)	Tập hợp các dữ liệu cùng kiểu.
Lớp (Class)	Là sự cài đặt mô tả về một đối tượng trong bài toán.
Giao diện (Interface)	Là một lớp thuần trừu tượng được tạo ra cho phép cài đặt đa thừa kế trong Java.

1. KIỂU DỮ LIỆU

❖ **Kiểu dữ liệu nguyên thủy** bao gồm có các loại sau. Ứng với mỗi kiểu dữ liệu bộ nhớ sẽ cấp phát vùng nhớ tương ứng

kiểu dữ liệu	kích thước	khoảng giá trị
byte	1 byte	- 2^7 đến 2^7-1 (-128 đến 127)
short	2 bytes	- 2^{15} đến $2^{15}-1$ (-32768 đến 32767)
int	4 bytes	- 2^{31} đến $2^{31}-1$
long	8 bytes	- 2^{63} đến $2^{63}-1$
float	4 bytes	6 đến 7 thập phân 1.000,1232321
double	8 bytes	15 dấu thập phân
boolean	1 bit	chứa giá trị true hoặc false
char	2 bytes	chứa các ký tự đơn

❖ Kiểu số nguyên (Integer):

- byte có thể chứa giá trị từ -128 (-2^7) đến 127 (2^7-1).
 - ✓ byte num1 = 120;
 - ✓ System.out.println(num1);

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        byte num1 = 120;  
        System.out.println(num1);  
    }  
}
```

❖ Kiểu số nguyên (Integer):

- short có thể chứa giá trị từ -32768 (-2^{15}) đến 32767 ($2^{15}-1$)
 - ✓ short num2 = 10000;
 - ✓ System.out.println(num2);

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        short num2 = 10000;  
        System.out.println(num2);  
    }  
}
```

❖ Kiểu số nguyên (Integer):

- int chứa giá trị từ -2^{31} đến $2^{31}-1$
 - ✓ `int num3 = 150;`
 - ✓ `System.out.println(num3);`

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        int num3 = 150;  
        System.out.println(num3);  
    }  
}
```

1. KIỂU DỮ LIỆU

❖ Kiểu số nguyên (Integer):

- long chứa giá trị từ -9223372036854775808 (-2^{63}) đến 9223372036854775807 ($2^{63}-1$)
 - ✓ long num4 = 10000000;
 - ✓ System.out.println(num4);

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        long num4 = 10000000;  
        System.out.println(num4);  
    }  
}
```


1. KIỂU DỮ LIỆU

❖ Kiểu số thực:

- float chứa từ $3.4e-038$ đến $3.4e+038$
 - ✓ float num5 = 15.75f;
 - ✓ System.out.println(num5);
- double chứa giá trị từ $1.7e-308$ to $1.7e+308$
 - ✓ double num6 = 9.99d;
 - ✓ System.out.println(num6);

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        float num5 = 15.75f;  
        System.out.println(num5);  
    }  
}
```

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        double num6 = 9.99d;  
        System.out.println(num6);  
    }  
}
```

1. KIỂU DỮ LIỆU

❖ **Kiểu dữ liệu Khoa học:** có thể dùng e để mô tả bội số của 10

- float f1 = 35e3f;
- double d1 = 12E4d;
- System.out.println(f1);
- System.out.println(d1);

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        float f1 = 35e3f;  
        double d1 = 12E4d;  
        System.out.println(f1);  
        System.out.println(d1);  
    }  
}
```

1. KIỂU DỮ LIỆU

❖ **Kiểu dữ liệu Boolean:** chứa kết quả đúng hay sai

- `boolean isMale = true;`
- `boolean isFemale = false;`
- `System.out.println(isMale);`
- `System.out.println(isFemale);`

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        boolean isMale = true;  
        boolean isFemale = false;  
        System.out.println(isMale);  
        System.out.println(isFemale);  
    }  
}
```

1. KIỂU DỮ LIỆU

❖ Kiểu dữ liệu ký tự:

- Kiểu char chỉ chứa 1 ký tự duy nhất

- ✓ char myChar = 'C';
- ✓ System.out.println(myChar);

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        char myChar = 'C';  
        System.out.println(myChar);  
    }  
}
```

- Trong kiểu ký tự chúng ta có thể sử dụng bảng mã ASCII để hiển thị giá trị

- ✓ char a = 65, b = 66, c = 67;
- ✓ System.out.println(a);
- ✓ System.out.println(b);
- ✓ System.out.println(c);

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        char a = 65, b = 66, c = 67;  
        System.out.println(a);  
        System.out.println(b);  
        System.out.println(c);  
    }  
}
```

❖ Kiểu dữ liệu chuỗi:

- Kiểu String dùng để lưu dạng chuỗi các ký tự
 - ✓ String firstProgram = "Hello World";
 - ✓ System.out.println(firstProgram);

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        String firstProgram = "Hello World";  
        System.out.println(firstProgram);  
    }  
}
```

1. KIỂU DỮ LIỆU

❖ Kiểu dữ liệu đối tượng:

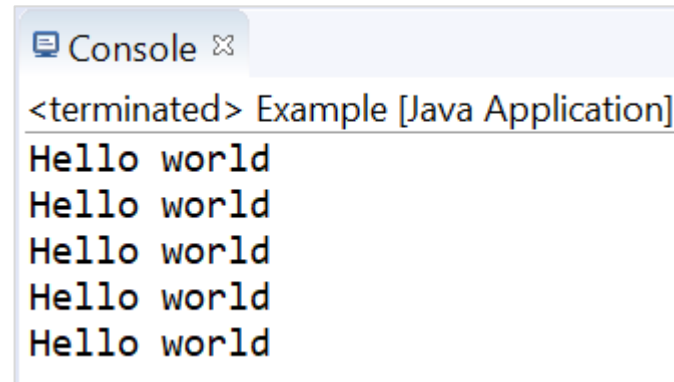
- Kiểu dữ liệu đối tượng thường tham chiếu tới 1 đối tượng
 - ✓ `Object o = new Object();`

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        Object o = new Object();  
    }  
}
```

2. BIẾN (VARIABLE)

❖ Viết chương trình in dòng chữ “Hello world” xuất hiện 5 lần trên màn hình.

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        System.out.println("Hello world");  
        System.out.println("Hello world");  
        System.out.println("Hello world");  
        System.out.println("Hello world");  
        System.out.println("Hello world");  
    }  
}
```



Console ✕

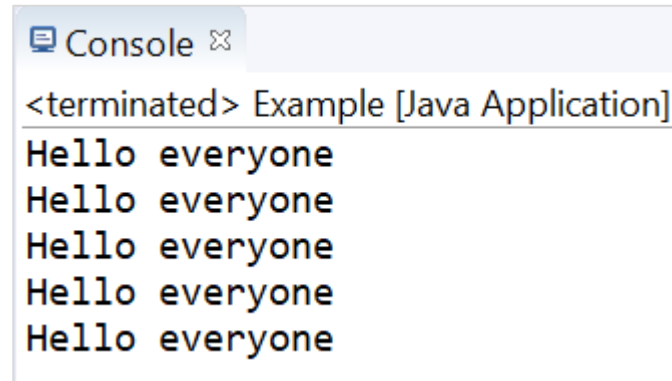
<terminated> Example [Java Application]

Hello world
Hello world
Hello world
Hello world
Hello world

2. BIẾN (VARIABLE)

- ❖ Hãy hiệu chỉnh chương trình để xuất hiện “Hello everyone” 5 lần trên màn hình.

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        System.out.println("Hello everyone");  
        System.out.println("Hello everyone");  
        System.out.println("Hello everyone");  
        System.out.println("Hello everyone");  
        System.out.println("Hello everyone");  
    }  
}
```



Console ✕
<terminated> Example [Java Application]
Hello everyone
Hello everyone
Hello everyone
Hello everyone
Hello everyone

- ❖ Thay thế Hello world bằng Hello everyone → Thực hiện được và nhanh chóng vì chương trình nhỏ gọn, nhưng chương trình lớn (số lượng dòng code nhiều) thì việc làm này không khả thi: phải hiệu chỉnh toàn bộ code/viết lại chương trình → mất thời gian hoặc nhầm lẫn → chương trình chạy cho kết quả sai.

2. BIẾN (VARIABLE)

❖ **Ý tưởng:** giả sử có hộp A bên trong chứa **Cam**. Cần dùng đến **Cam** thì truy cập hộp A. Như vậy **Cam** được hiểu là nội dung chứa bên trong hộp A. Chúng ta có thể thay đổi **Cam** bằng **Xoài**, **Quýt** hoặc bất kỳ loại trái cây nào khác thì nội dung bên trong của hộp A cũng thay đổi theo. Vì vậy khi truy cập đến hộp A sẽ nhận được loại trái cây mới thay thế. Hộp A chứa **Cam**. Truy cập hộp A 5 lần sẽ lấy được 5 quả **Cam** (1 lần truy cập lấy được 1 quả). Thay **Cam** bằng **Quýt** → hộp A chứa **Quýt**. Truy cập hộp A 5 lần sẽ lấy được 5 quả **Quýt** (1 lần truy cập lấy được 1 quả)

- **Hộp A gọi là biến: có thể**
- sử dụng nhiều lần;
 - thay đổi/cập nhật nội dung.

2. BIẾN (VARIABLE)

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        System.out.println("Lấy Cam");  
        System.out.println("Lấy Cam");  
        System.out.println("Lấy Cam");  
        System.out.println("Lấy Cam");  
        System.out.println("Lấy Cam");  
    }  
}
```

Hiệu chỉnh

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        System.out.println("Lấy Quýt");  
        System.out.println("Lấy Quýt");  
        System.out.println("Lấy Quýt");  
        System.out.println("Lấy Quýt");  
        System.out.println("Lấy Quýt");  
    }  
}
```

Hiệu chỉnh

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        Hộp A chứa Cam; //Hộp A là biến  
        System.out.println("Truy cập A");  
        System.out.println("Truy cập A");  
        System.out.println("Truy cập A");  
        System.out.println("Truy cập A");  
        System.out.println("Truy cập A");  
    }  
}
```

Hiệu chỉnh

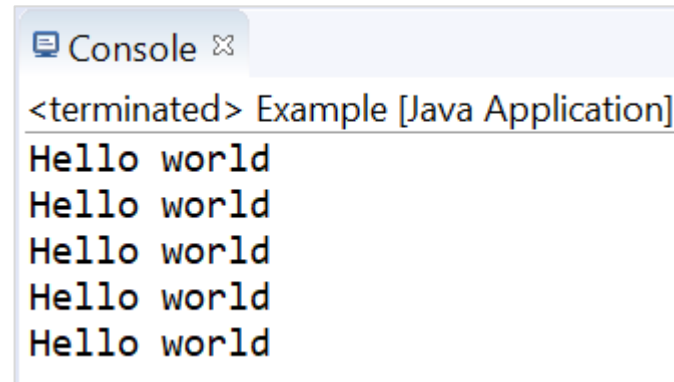
```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        Hộp A chứa Quýt; //Hộp A là biến  
        System.out.println("Truy cập A");  
        System.out.println("Truy cập A");  
        System.out.println("Truy cập A");  
        System.out.println("Truy cập A");  
        System.out.println("Truy cập A");  
    }  
}
```

Hiệu chỉnh

2. BIẾN (VARIABLE)

❖ Viết chương trình in dòng chữ “Hello world” xuất hiện 5 lần trên màn hình.

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        String str = "Hello world";//Biến str  
        System.out.println(str);  
        System.out.println(str);  
        System.out.println(str);  
        System.out.println(str);  
        System.out.println(str);  
    }  
}
```



Console ✕

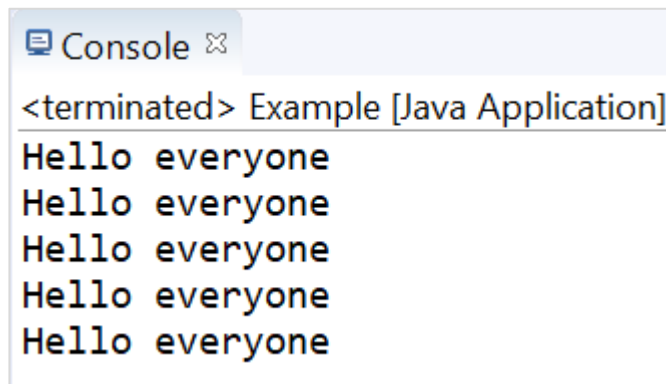
<terminated> Example [Java Application]

Hello world
Hello world
Hello world
Hello world
Hello world

2. BIẾN (VARIABLE)

❖ Hãy hiệu chỉnh chương trình để xuất hiện “Hello everyone” 5 lần trên màn hình.

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        String str = "Hello everyone";//Biến str  
        System.out.println(str);  
        System.out.println(str);  
        System.out.println(str);  
        System.out.println(str);  
        System.out.println(str);  
    }  
}
```



Console ✕

<terminated> Example [Java Application]

Hello everyone
Hello everyone
Hello everyone
Hello everyone
Hello everyone

2. BIẾN (VARIABLE)



Biến (Variable) sử dụng để lưu lại giá trị cho chương trình xử lý. Mỗi biến được cấp phát một vùng nhớ trong bộ nhớ (memory). *Có 3 kiểu biến trong java gồm biến cục bộ (local), biến toàn cục (instance) và biến tĩnh (static)*

2. BIẾN (VARIABLE)

❖ Lưu ý:

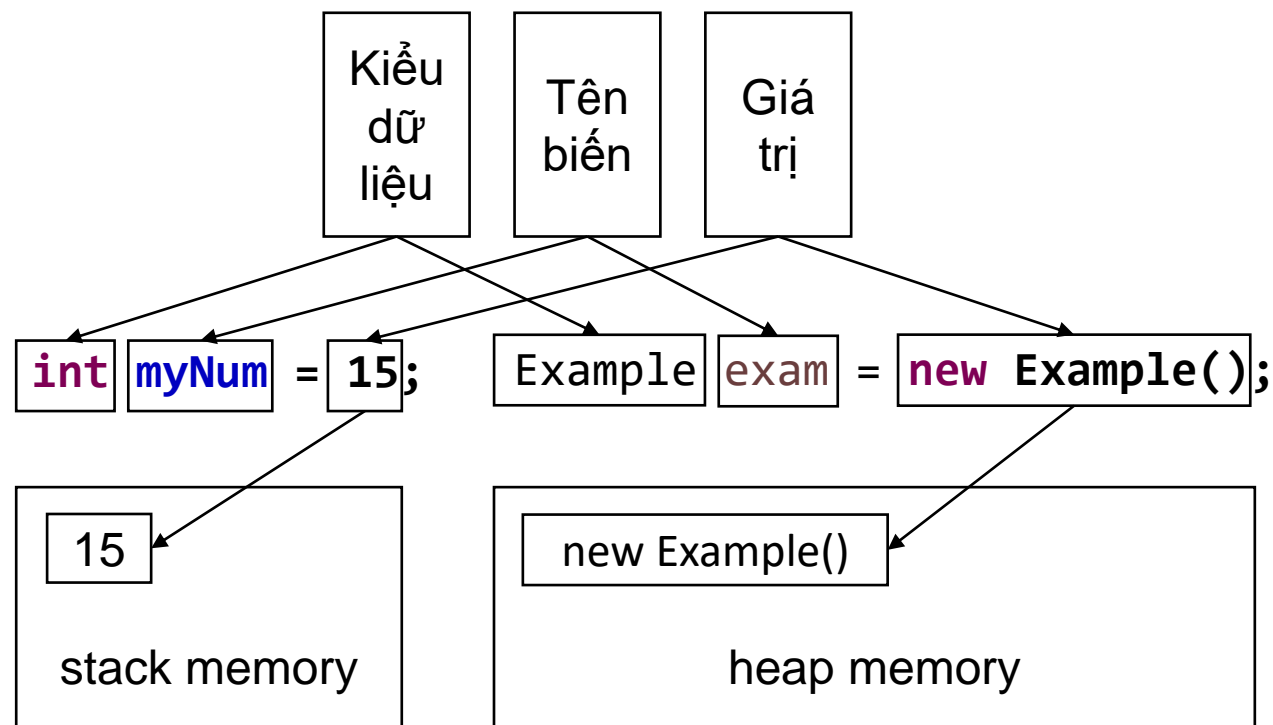
- Biến lưu giữ trực tiếp các giá trị số, ký tự hoặc chuỗi ký tự.
- Biến chỉ lưu giữ địa chỉ của mảng hoặc địa chỉ của instance trên bộ nhớ máy tính.



2. BIẾN (VARIABLE)

❖ Biến (variable):

- Giá trị trên bộ nhớ mà biến liên kết đến chính là giá trị của biến
- Kiểu dữ liệu của biến chính là kiểu dữ liệu lưu trên bộ nhớ.



2. BIẾN (VARIABLE)

❖ Khai báo biến:

- Cú pháp: **type name = value;**
- **type** là kiểu dữ liệu mà biến có thể lưu trữ (kiểu số, kiểu văn bản, ...) tương ứng với int, long, String, ...
- **name**: tên biến là một chuỗi ký tự được đặt theo quy tắc đặt tên.
- **value** là giá trị khởi tạo cho biến (có thể có hoặc không), nếu bỏ qua phần này thì giá trị ban đầu của biến được khởi tạo giá trị mặc định.
- **Ví dụ 1**: **type** là int thì **value** phải là kiểu số nguyên và nằm trong phạm vi từ -2,147,483,648 (-2^{31}) đến 2,147,483,647 ($2^{31}-1$).

2. BIẾN (VARIABLE)

❖ Đặt tên biến:

- Ký tự bắt đầu của tên biến phải là chữ cái, hoặc dấu gạch dưới (_), hoặc ký tự đô la (\$)
 - ✓ `int count;` → đúng
 - ✓ `int _count;` → đúng
 - ✓ `int $count;` → đúng
- Không chứa khoảng trắng
 - ✓ `int my Num;` → sai
 - ✓ `int myNum;` → đúng
 - ✓ `int my_num;` → sai

2. BIẾN (VARIABLE)

❖ Đặt tên biến:

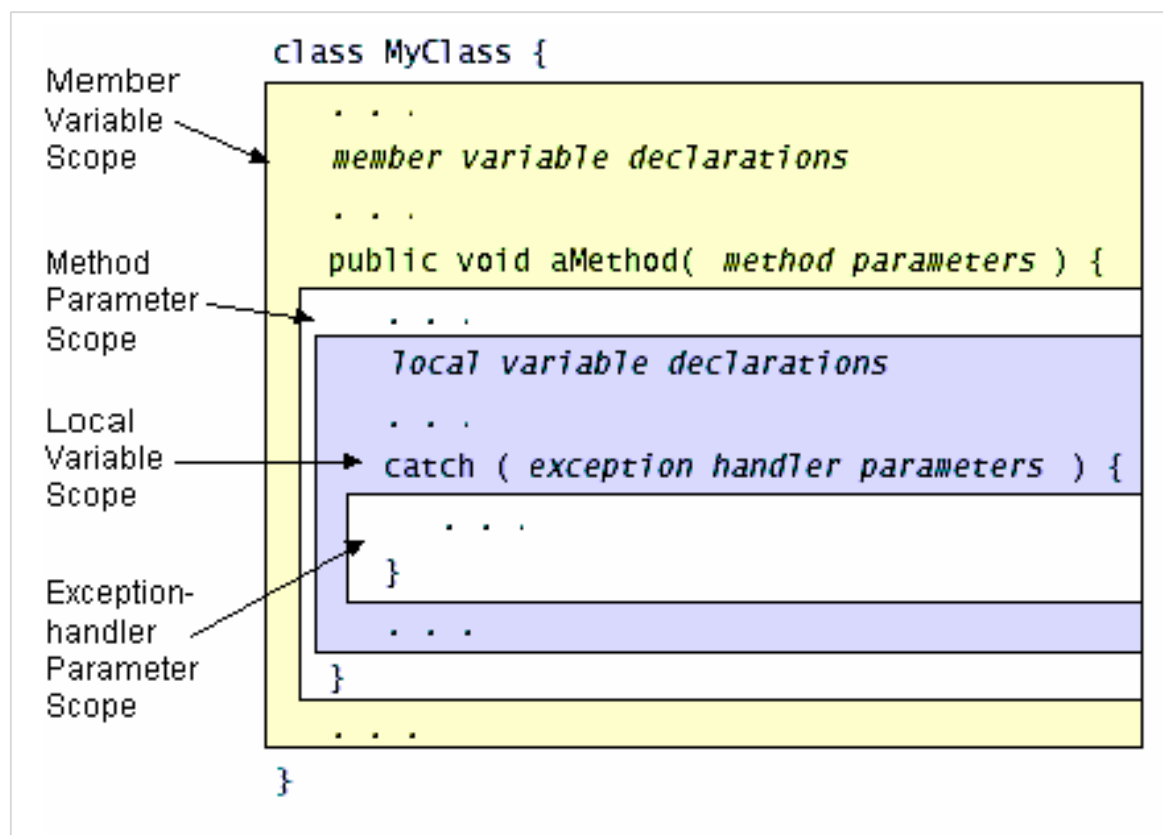
- Không chứa ký tự đặc biệt !@#%^&*
 - ✓ `int c@unt;` → sai
 - ✓ `int count#;` → sai
 - ✓ `int count*count;` → sai
- Không dùng từ khóa đặt tên biến
 - ✓ `boolean continue = true;` → sai
 - ✓ `long class;` → sai
 - ✓ `int final;` → sai
- **Lưu ý:** Đặt tên biến theo quy tắc camelCase (xem lại Chương 2)

2. BIẾN (VARIABLE)

- ❖ **Khai báo biến** là giới thiệu cho chương trình biết một biến sẽ sử dụng có tên cùng với kiểu dữ liệu.
- ❖ **Khởi tạo biến** là giới thiệu cho chương trình biết một biến sẽ sử dụng có tên cùng với kiểu dữ liệu và **gán giá trị ban đầu cho biến**.

2. BIẾN (VARIABLE)

❖ **Phạm vi biến:** Mỗi biến được khai báo sẽ có một phạm vi hoạt động nhất định, phạm vi của biến là nơi mà biến có thể được truy cập, điều này xác định cả tính thấy được và thời gian sống của biến.



2. BIẾN (VARIABLE)

❖ **Biến cục bộ:**

- Biến cục bộ được khai báo trong các phương thức, hàm constructor hoặc trong các block.
- Biến cục bộ được tạo bên trong các phương thức, constructor, block và sẽ bị phá hủy khi kết thúc các phương thức, constructor và block.
- Các biến cục bộ sẽ nằm trên vùng bộ nhớ stack của bộ nhớ.
- Cần khởi tạo giá trị mặc định cho biến cục bộ trước khi sử dụng.
- Không sử dụng access modifier khi khai báo biến cục bộ.

2. BIẾN (VARIABLE)

❖ Ví dụ 2

```
public class Example {  
    public void myValue()  
    {  
        int num = 10; //Biến cục bộ  
        System.out.println("Giá trị của num là " + num);  
    }  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        Example exam = new Example();  
        exam.myValue();  
    }  
}
```

Console

<terminated> Example [Java Application]
Giá trị của num là 10

2. BIẾN (VARIABLE)

❖ Ví dụ 3

```
public class Example {  
    public void myValue()  
    {  
        int num; //Biến cục bộ  
        System.out.println("Giá trị của num là " + num);  
    }  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        Example exam = new Example();  
        exam.myValue();  
    }  
}
```

Console

<terminated> Example [Java Application] C:\Program Files\Java\jdk-13\bin\javaw.exe (Aug 28, 2022, 9:20:11 PM)

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The local variable num may not have been initialized

at Example.myValue(Example.java:5)
at Example.main(Example.java:10)

2. BIẾN (VARIABLE)

❖ **Biến toàn cục (instance/global):**

- Khai báo trong một lớp (class), bên ngoài các phương thức, constructor và các block.
- Lưu trong bộ nhớ heap.
- Sử dụng từ khóa new để tạo và bị phá hủy khi đối tượng bị phá hủy.
- Có thể được sử dụng bởi các phương thức, constructor, block, ... Nhưng phải thông qua một đối tượng cụ thể.
- Có giá trị mặc định phụ thuộc vào kiểu dữ liệu.
- Có thể gọi trực tiếp bằng tên bên trong class.
- Được phép sử dụng access modifier khi khai báo biến toàn cục, mặc định là default.

2. BIẾN (VARIABLE)

❖ Ví dụ 4

```
public class Example {  
    int myNum = 15; // Biến toàn cục  
    public void myValue1()  
    {  
        System.out.println("Giá trị của num1 là " + myNum);  
    }  
    public void myValue2()  
    {  
        System.out.println("Giá trị của num2 là " + (myNum + 1));  
    }  
    public static void main(String[] args) {  
        // Các câu lệnh xử lý  
        Example exam = new Example();  
        exam.myValue1();  
        exam.myValue2();  
        System.out.println("Giá trị của num3 là " + (exam.myNum + 2));  
    }  
}
```

Console

<terminated> Example [Java Application]

Giá trị của num1 là 15

Giá trị của num2 là 16

Giá trị của num3 là 17

2. BIẾN (VARIABLE)

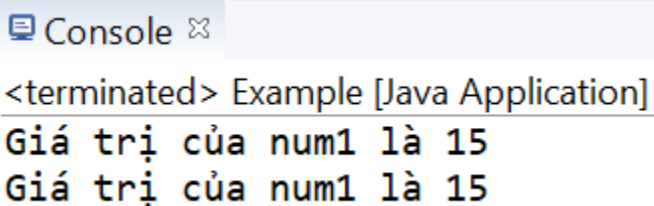
❖ Biến tĩnh:

- Khai báo trong một class với từ khóa static, phía bên ngoài các phương thức, constructor và block.
- Chỉ có duy nhất một bản sao của các biến static được tạo ra, dù bạn tạo bao nhiêu đối tượng từ lớp tương ứng.
- Lưu trữ trong bộ nhớ static riêng.
- Được tạo khi chương trình bắt đầu chạy và chỉ bị phá hủy khi chương trình dừng.
- Giá trị mặc định của biến static phụ thuộc vào kiểu dữ liệu khai báo.
- Được truy cập thông qua tên lớp (class) chứa nó, với cú pháp: `TenClass.tenBien`.
- Trong class, các phương thức sử dụng biến static bằng cách gọi tên của nó khi phương thức đó cũng được khai báo với từ khóa static.

2. BIẾN (VARIABLE)

❖ Ví dụ 5

```
public class Example {  
    public static int myNum = 15; //Biến tĩnh  
    public static void main(String[] args) {  
        //Biến tĩnh được gọi trực tiếp  
        System.out.println("Giá trị của num1 là " + myNum);  
        //Biến tĩnh được gọi thông qua tên lớp  
        System.out.println("Giá trị của num1 là " + Example.myNum);  
    }  
}
```



Console ✕
<terminated> Example [Java Application]
Giá trị của num1 là 15
Giá trị của num1 là 15

3. HẲNG (CONSTANT)

- ❖ **Hằng** là loại biến chỉ có thể gán giá trị cho nó một lần duy nhất và giá trị này được giữ không đổi trong suốt quá trình sử dụng.
- ❖ **Cú pháp:** *final type name = value;*
 - *final*: từ khóa xác định biến thuộc dạng hằng số
 - *type*: kiểu biến
 - *name*: tên biến (viết hoa, nếu nhiều từ thì phải nối nhau bằng dấu gạch dưới _)
 - *value*: giá trị gán cho biến
- ❖ **Lưu ý:** Cách đặt tên hằng (Xem lại Chương 2)

3. HẲNG (CONSTANT)

❖ Ưu điểm

- Khai báo một lần nhưng có thể sử dụng nhiều lần trong chương trình;
- Khi muốn thay đổi chỉ cần cập nhật giá trị tại một nơi duy nhất trong mã nguồn;
- Tiết kiệm thời gian;
- Thay thế nhập dữ liệu trực tiếp;
- Hạn chế sai sót;
- Tăng chất lượng mã nguồn chương trình.

3. HẲNG (CONSTANT)

❖ Ví dụ 1

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        final int MY_NUM = 10; //Khai báo hằng  
        System.out.println("Giá trị của MY_NUM là " + MY_NUM);  
    }  
}
```

Console

<terminated> Example [Java Application]
Giá trị của MY_NUM là 10

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        int myNum = 10; //Khai báo biến  
        System.out.println("Giá trị của num là " + myNum);  
        myNum = 11; //Gán lại giá trị cho biến  
        System.out.println("Giá trị của num là " + myNum);  
    }  
}
```

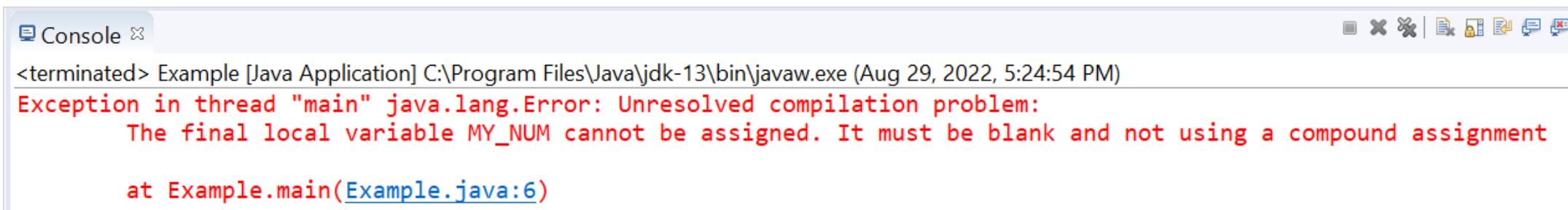
Console

<terminated> Example [Java Application]
Giá trị của num là 10
Giá trị của num là 11

3. HẲNG (CONSTANT)

❖ Ví dụ 2

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        final int MY_NUM = 10; //Khai báo hằng  
        System.out.println("Giá trị của MY_NUM là " + MY_NUM);  
        MY_NUM = 11; //Gán lại giá trị cho hằng  
        System.out.println("Giá trị của MY_NUM là " + MY_NUM);  
    }  
}
```



Console

<terminated> Example [Java Application] C:\Program Files\Java\jdk-13\bin\javaw.exe (Aug 29, 2022, 5:24:54 PM)

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The final local variable MY_NUM cannot be assigned. It must be blank and not using a compound assignment

at Example.main(Example.java:6)

Hằng chỉ gán giá trị một lần duy nhất lúc khai báo, không thể gán lại giá trị.

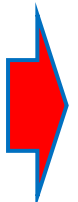
4. KIỂU ENUM

- ❖ **Enum** là một kiểu dữ liệu đặc biệt cho phép một biến có thể là tập hợp các hằng số cụ thể. Một enum có thể chứa các trường, phương thức và Constructor.
- ❖ Enum được định nghĩa: bên trong lớp, bên ngoài lớp, trong một file riêng.
- ❖ Duyệt các phần tử trong enum: sử dụng vòng lặp for
- ❖ Khởi tạo giá trị cho hằng số enum (được làm rõ ở học phần [Lập trình hướng đối tượng](#))
- ❖ Sử dụng enum trong câu lệnh **Switch** (làm rõ ở Chương 6)
- ❖ So sánh các phần tử enum sử dụng toán tử **==** hoặc phương thức **equals()**

4. KIỂU ENUM

❖ **Ví dụ 1:** định nghĩa enum bên trong lớp

```
public class Example {  
    //Định nghĩa enum bên trong lớp  
    enum Weekday{  
        MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;  
    }  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        System.out.println("Thứ 2 -> "+ Weekday.MONDAY);  
        System.out.println("Thứ 3 -> "+ Weekday.TUESDAY);  
        System.out.println("Thứ 4 -> "+ Weekday.WEDNESDAY);  
        System.out.println("Thứ 5 -> "+ Weekday.THURSDAY);  
        System.out.println("Thứ 6 -> "+ Weekday.FRIDAY);  
        System.out.println("Thứ 7 -> "+ Weekday.SATURDAY);  
        System.out.println("Chủ nhật -> "+ Weekday.SUNDAY);  
    }  
}
```



Console

<terminated> Example [Java]

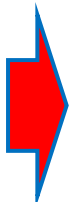
Thứ 2 ->	MONDAY
Thứ 3 ->	TUESDAY
Thứ 4 ->	WEDNESDAY
Thứ 5 ->	THURSDAY
Thứ 6 ->	FRIDAY
Thứ 7 ->	SATURDAY
Chủ nhật ->	SUNDAY

4. KIỂU ENUM

❖ Ví dụ 2: định nghĩa enum bên ngoài lớp

```
//Định nghĩa enum bên ngoài lớp
enum Weekday{
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;
}

public class Example {
    public static void main(String[] args) {
        //Các câu lệnh xử lý
        System.out.println("Thứ 2 -> "+ Weekday.MONDAY);
        System.out.println("Thứ 3 -> "+ Weekday.TUESDAY);
        System.out.println("Thứ 4 -> "+ Weekday.WEDNESDAY);
        System.out.println("Thứ 5 -> "+ Weekday.THURSDAY);
        System.out.println("Thứ 6 -> "+ Weekday.FRIDAY);
        System.out.println("Thứ 7 -> "+ Weekday.SATURDAY);
        System.out.println("Chủ nhật -> "+ Weekday.SUNDAY);
    }
}
```



Console

<terminated> Example [Java]


Thứ 2 ->	MONDAY
Thứ 3 ->	TUESDAY
Thứ 4 ->	WEDNESDAY
Thứ 5 ->	THURSDAY
Thứ 6 ->	FRIDAY
Thứ 7 ->	SATURDAY
Chủ nhật ->	SUNDAY

4. KIỂU ENUM

❖ Ví dụ 3: định nghĩa enum trong một file riêng

```
package weekday;  
public enum Weekday {  
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;  
}
```

```
//import package chứa enum định nghĩa trong file Weekday.java  
import weekday.Weekday;  
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        System.out.println("Thứ 2 -> "+ Weekday.MONDAY);  
        System.out.println("Thứ 3 -> "+ Weekday.TUESDAY);  
        System.out.println("Thứ 4 -> "+ Weekday.WEDNESDAY);  
        System.out.println("Thứ 5 -> "+ Weekday.THURSDAY);  
        System.out.println("Thứ 6 -> "+ Weekday.FRIDAY);  
        System.out.println("Thứ 7 -> "+ Weekday.SATURDAY);  
        System.out.println("Chủ nhật -> "+ Weekday.SUNDAY);  
    }  
}
```



Console

<terminated> Example [Java]

Thứ 2 ->	MONDAY
Thứ 3 ->	TUESDAY
Thứ 4 ->	WEDNESDAY
Thứ 5 ->	THURSDAY
Thứ 6 ->	FRIDAY
Thứ 7 ->	SATURDAY
Chủ nhật ->	SUNDAY

4. KIỂU ENUM

❖ **Ví dụ 4:** khởi tạo giá trị cho hằng số enum (tham khảo)

```
public class Example {  
    enum Weekday {  
        MONDAY(2), TUESDAY(3), WEDNESDAY(4), THURSDAY(5), FRIDAY(6), SATURDAY(7), SUNDAY(1);  
        private int value;  
        private Weekday(int value) {  
            this.value = value;  
        }  
    }  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        System.out.println("Thứ 2 -> " + Weekday.MONDAY + ": " + Weekday.MONDAY.value);  
        System.out.println("Thứ 3 -> " + Weekday.TUESDAY + ": " + Weekday.TUESDAY.value);  
        System.out.println("Thứ 4 -> " + Weekday.WEDNESDAY + ": " + Weekday.WEDNESDAY.value);  
        System.out.println("Thứ 5 -> " + Weekday.THURSDAY + ": " + Weekday.THURSDAY.value);  
        System.out.println("Thứ 6 -> " + Weekday.FRIDAY + ": " + Weekday.FRIDAY.value);  
        System.out.println("Thứ 7 -> " + Weekday.SATURDAY + ": " + Weekday.SATURDAY.value);  
        System.out.println("Chủ nhật -> " + Weekday.SUNDAY + ": " + Weekday.SUNDAY.value);  
    }  
}
```



Console

<terminated> Example [Java App]

Thứ 2 ->	MONDAY: 2
Thứ 3 ->	TUESDAY: 3
Thứ 4 ->	WEDNESDAY: 4
Thứ 5 ->	THURSDAY: 5
Thứ 6 ->	FRIDAY: 6
Thứ 7 ->	SATURDAY: 7
Chủ nhật ->	SUNDAY: 1

4. KIỂU ENUM

❖ Ví dụ 5: khởi tạo nhiều giá trị cho hằng số enum (tham khảo)

```
package season;

public enum Season {
    SPRING("Spring months: Jan-Mar", "Cool", "The 1st quarter"),
    SUMMER("Summer months: Apr-Jun", "Hot", "The 2nd quarter"),
    AUTUMN("Autumn months: Jul-Sep", "Chilly", "The 3rd quarter"),
    WINTER("Winter months: Oct-Dec", "Cold", "The 4th quarter");

    private String month;
    private String climate;
    private String quarter;

    private Season(String month, String climate, String quarter) {
        this.month = month;
        this.climate = climate;
        this.quarter = quarter;
    }
    //get & set
    public String getMonth() {
        return month;
    }
    public void setMonth(String month) {
        this.month = month;
    }
    public String getClimate() {
        return climate;
    }
    public void setClimate(String climate) {
        this.climate = climate;
    }
    public String getQuarter() {
        return quarter;
    }
    public void setQuarter(String quarter) {
        this.quarter = quarter;
    }
}
```

```
import season.Season;
public class Example {
    public static void main(String[] args) {
        //Các câu lệnh xử lý
        System.out.println(Season.SPRING.getMonth() + " | " + Season.SPRING.getClimate() + " | " +
            Season.SPRING.getQuarter());
        System.out.println(Season.SUMMER.getMonth() + " | " + Season.SUMMER.getClimate() + " | " +
            Season.SUMMER.getQuarter());
        System.out.println(Season.AUTUMN.getMonth() + " | " + Season.AUTUMN.getClimate() + " | " +
            Season.AUTUMN.getQuarter());
        System.out.println(Season.WINTER.getMonth() + " | " + Season.WINTER.getClimate() + " | " +
            Season.WINTER.getQuarter());
    }
}
```



Console

<terminated> Example [Java Application] C:\Program Files\Java\jdk-13\

```
Spring months: Jan-Mar | Cool | The 1st quarter
Summer months: Apr-Jun | Hot | The 2nd quarter
Autumn months: Jul-Sep | Chilly | The 3rd quarter
Winter months: Oct-Dec | Cold | The 4th quarter
```

4. KIỂU ENUM

❖ **Ví dụ 6:** so sánh các phần tử trong enum

```
public class Example {  
    enum Weekday {  
        MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;  
    }  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        Weekday wee1 = Weekday.SUNDAY;  
        Weekday wee2 = Weekday.MONDAY;  
        //Sử dụng toán tử ==  
        if(wee1 == Weekday.SUNDAY)  
            System.out.println("Sunday");  
        //Sử dụng phương thức equals()  
        if(wee2.equals(Weekday.MONDAY))  
            System.out.println("Monday");  
    }  
}
```



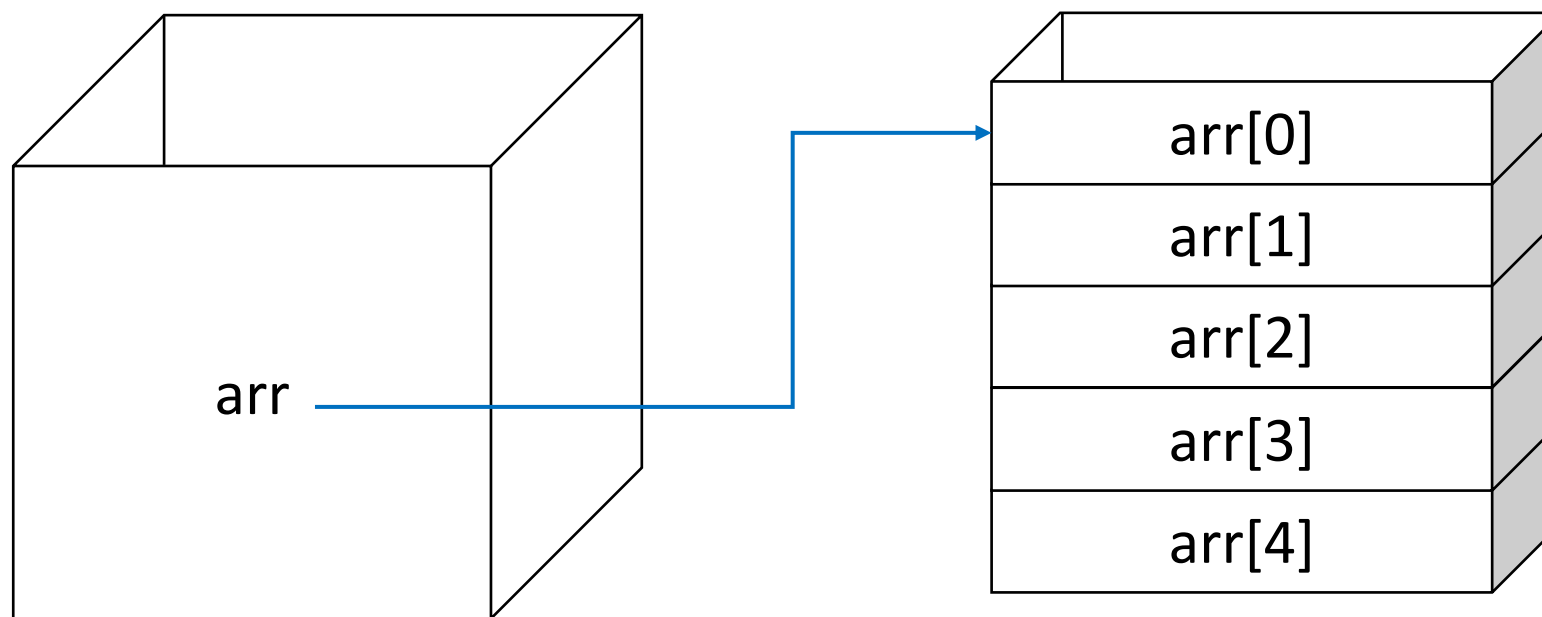
Console ✕
<terminated> Example
Sunday
Monday

❖ Câu hỏi

- Phân biệt các kiểu dữ liệu?
- Phân biệt giữa các loại biến?
- Phân biệt giữa biến với hằng?
- Trình bày kiểu dữ liệu enum?
- Trình bày qui tắc đặt tên biến, hằng, enum?
- Phân biệt giữa bộ nhớ stack với heap?

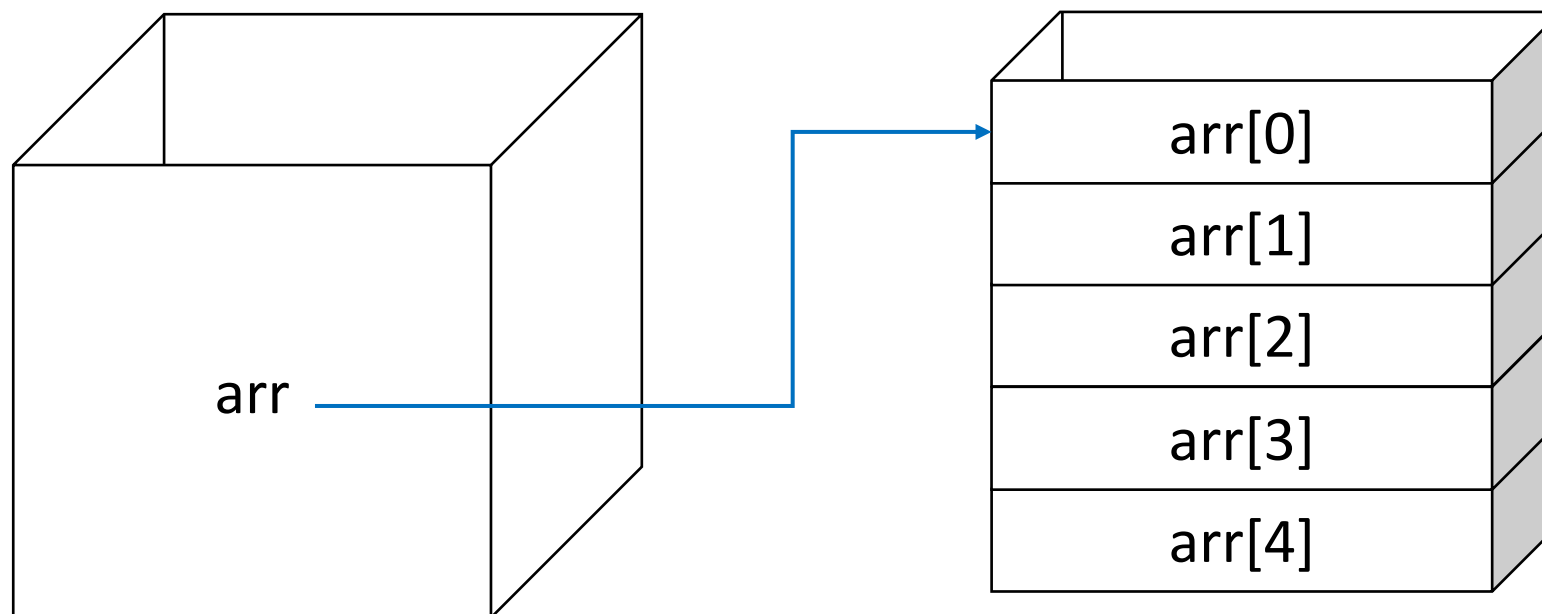
5. MẢNG (ARRAY)

- ❖ **Mảng** là một tập hợp gồm nhiều phần tử có cùng kiểu dữ liệu và có chung một tên.
Mỗi phần tử của mảng đóng vai trò như một biến và lưu giữ được một giá trị.
- ❖ **Biến mảng** sẽ lưu giữ địa chỉ của phần tử đầu tiên của nó trong bộ nhớ máy tính.



5. MẢNG (ARRAY)

- ❖ Số lượng kiểu biến cũng chính là số lượng kiểu mảng.
- ❖ Mảng cần khai báo để xác định rõ:
 - Kiểu mảng (int, float, double, ...);
 - Tên mảng;
 - Số chiều và kích thước mỗi chiều.



5. MẢNG (ARRAY)

❖ Ví dụ 1:

- `int arr[10], arr1[4][2];`
- `float arr2[6], arr3[3][3];`

❖ Trong ví dụ 1 xác định có 4 mảng: `arr`, `arr1`, `arr2`, `arr3`. Có ý nghĩa như sau:

- **Mảng thứ nhất** kiểu là `int`, tên là `arr`, số chiều là 1, kích thước bằng 10. Mảng `arr` có 10 phần tử được đánh số như sau: `arr[0]`, `arr[1]`, ..., `arr[9]`. Ứng với mỗi phần tử `arr[i]` chứa được 1 giá trị kiểu `int`. Như vậy mảng `arr` có thể biểu diễn được 1 dãy 10 số nguyên.

5. MẢNG (ARRAY)

❖ Ví dụ 1:

- `int arr[10], arr1[4][2];`
- `float arr2[6], arr3[3][3];`

❖ Trong ví dụ 1 xác định có 4 mảng: `arr`, `arr1`, `arr2`, `arr3`. Có ý nghĩa như sau:

- **Mảng thứ hai** kiểu `int`, tên là `arr1`, số chiều là 2, kích thước của các chiều là 4 và 2. Mảng `arr1` có 8 phần tử được đánh số như sau:
 - ✓ `arr1[0][0]` `arr1[0][1]`
 - ✓ `arr1[1][0]` `arr1[1][1]`
 - ✓ `arr1[2][0]` `arr1[2][1]`
 - ✓ `arr1[3][0]` `arr1[3][1]`
- Mỗi phần tử `arr1[i][j]` chứa 1 giá trị kiểu `int` và mảng `arr1` có thể biểu diễn được 1 bảng số nguyên 4 dòng, 2 cột.

5. MẢNG (ARRAY)

❖ Ví dụ 1:

- `int arr[10], arr1[4][2];`
- `float arr2[6], arr3[3][3];`

❖ Trong ví dụ 1 xác định có 4 mảng: `arr`, `arr1`, `arr2`, `arr3`. Có ý nghĩa như sau:

- **Mảng thứ ba** kiểu `float`, tên là `arr2`, số chiều là 1, kích thước bằng 6. Mảng `arr2` có 6 phần tử được đánh số như sau: `arr2[0]`, `arr2[1]`, `arr2[2]`, `arr2[3]`, `arr2[4]`, `arr2[5]`. Mỗi phần tử `arr2[i]` chứa 1 giá trị kiểu `float` và mảng `arr2` có thể biểu diễn được một dãy 6 số thực.

5. MẢNG (ARRAY)

❖ Ví dụ 1:

- `int arr[10], arr1[4][2];`
- `float arr2[6], arr3[3][3];`

❖ Trong ví dụ 1 xác định có 4 mảng: `arr`, `arr1`, `arr2`, `arr3`. Có ý nghĩa như sau:

- **Mảng thứ tư** kiểu `float`, tên là `arr3`, số chiều là 2, kích thước của các chiều là 3.

Mảng `arr3` có 9 phần tử được đánh số như sau:

- ✓ `arr3[0][0]` `arr3[0][1]` `arr3[0][2]`
- ✓ `arr3[1][0]` `arr3[1][1]` `arr3[1][2]`
- ✓ `arr3[2][0]` `arr3[2][1]` `arr3[2][2]`

- Mỗi phần tử `arr3[i][j]` chứa 1 giá trị kiểu `float` và mảng `arr3` có thể biểu diễn được 1 bảng số thực 3 dòng, 3 cột.

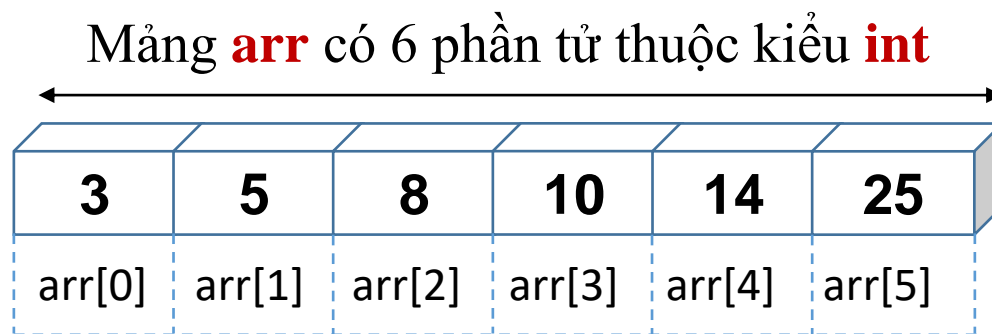
5. MẢNG (ARRAY)

❖ Lưu ý:

- Các phần tử của mảng được cấp phát các vùng nhớ liên tiếp nhau trong bộ nhớ. Các phần tử mảng có địa chỉ liên tiếp nhau.
- Trong bộ nhớ, các phần tử của mảng 2 chiều được sắp xếp theo hàng.

5. MẢNG (ARRAY)

❖ **Mảng một chiều** là tập hợp các phần tử được sắp xếp liên tục và có thứ tự trên bộ nhớ máy tính. Các phần tử trong mảng được đánh số thứ tự hay còn gọi là chỉ số. Truy cập đến phần tử của mảng được thực hiện thông qua chỉ số. Chỉ số phần tử đầu tiên bằng 0, chỉ số phần tử cuối cùng bằng $n-1$ đối với mảng có kích thước n (có n phần tử).



int arr[] = {3, 5, 8, 10, 14, 25};

❖ **Lưu ý:** Sử dụng phương thức **toString()** để chuyển đổi mảng thành chuỗi và in ra màn hình. Ví dụ: in mảng **arr** ra màn hình → **Arrays.toString(arr);**

5. MẢNG (ARRAY)

❖ Cú pháp khai báo mảng

- *type* *name*[];
- Hoặc *type*[] *name*;
- *type*: kiểu dữ liệu; *name*: tên mảng.
- **Ví dụ 2**: `int arr[];` // Khai báo một mảng có tên arr kiểu nguyên (int)

❖ Cú pháp khai báo và tạo mảng

- *type* *name*[] = *new type[size]*;
- *type*: kiểu dữ liệu; *name*: tên mảng; *size*: kích thước của mảng/số phần tử của mảng.

- ❖ **Lưu ý**: Để xác định kích thước của mảng sử dụng thuộc tính *length*. Cú pháp: *name.length* (name: tên mảng).

5. MẢNG (ARRAY)

❖ Cú pháp khai báo và tạo mảng

- *type* *name*[] = new *type*[*size*];
- *type*: kiểu dữ liệu; *name*: tên mảng; *size*: kích thước của mảng/số phần tử của mảng.
- **Ví dụ 3:**
 - ✓ `int arr[] = new int[5];` //Khai báo một mảng có tên arr kiểu nguyên (int) gồm 5 phần tử. Giá trị các phần tử mặc định bằng 0.
 - ✓ `String str[] = new String[5];` //Khai báo một mảng có tên str kiểu chuỗi (String) gồm 5 phần tử. Giá trị các phần tử mặc định bằng null.
 - ✓ Viết chương trình kiểm chứng hai trường hợp cho ở ví dụ 3 (arr và str).

5. MẢNG (ARRAY)

❖ Khai báo và khởi tạo giá trị cho mảng

- **Ví dụ 4:** `int arr[] = {3, 5, 8, 10, 14, 25};` // Khai báo và khởi tạo một mảng có tên arr kiểu nguyên (int) gồm 6 phần tử, giá trị các phần tử đã được xác định cụ thể.
- **Ví dụ 5:** `String nguHanh[] = {"kim", "mộc", "thủy", "hỏa", "thổ"};` // Khai báo và khởi tạo một mảng có tên nguHanh kiểu chuỗi (String) gồm 5 phần tử, giá trị các phần tử đã được xác định cụ thể.

❖ Truy cập đến phần tử của mảng

- Được thực hiện thông qua chỉ số
- Mảng arr cho ở trên có 6 phần tử, truy cập đến các phần tử của mảng thực hiện như sau: `arr[0]`, `arr[1]`, `arr[2]`, `arr[3]`, `arr[4]`, `arr[5]`

❖ Viết chương trình hiển thị các mảng cho ở ví dụ 4, ví dụ 5 (arr và nguHanh).

5. MẢNG (ARRAY)

❖ Ví dụ 6

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        int arr[] = {3, 5, 8, 10, 14, 25}; //Khai báo mảng 1 chiều arr kiểu int  
        int cs = 0; //Biến lưu chỉ số mảng  
        System.out.printf("Phần tử thứ 1: arr[%d] = %d", cs, arr[cs]);  
        System.out.printf("\nPhần tử thứ 2: arr[%d] = %d", cs=cs+1, arr[cs]);  
        System.out.printf("\nPhần tử thứ 3: arr[%d] = %d", cs=cs+1, arr[cs]);  
        System.out.printf("\nPhần tử thứ 4: arr[%d] = %d", cs=cs+1, arr[cs]);  
        System.out.printf("\nPhần tử thứ 5: arr[%d] = %d", cs=cs+1, arr[cs]);  
        System.out.printf("\nPhần tử thứ 6: arr[%d] = %d", cs=cs+1, arr[cs]);  
    }  
}
```



Console

<terminated> Example [Java Application]

```
Phần tử thứ 1: arr[0] = 3  
Phần tử thứ 2: arr[1] = 5  
Phần tử thứ 3: arr[2] = 8  
Phần tử thứ 4: arr[3] = 10  
Phần tử thứ 5: arr[4] = 14  
Phần tử thứ 6: arr[5] = 25
```

5. MẢNG (ARRAY)

❖ **Mảng đa chiều:** là mảng của các mảng

- Chứa các mảng 1 chiều;
- Không lưu giữ trực tiếp các phần tử, các phần tử được lưu giữ thông qua các mảng 1 chiều bên trong nó;
- Gồm các dòng và các cột, được truy cập thông qua chỉ số dòng và cột.

	Cột	Cột	Cột	Cột
Dòng	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Dòng	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Dòng	a[2][0]	a[2][1]	a[2][2]	a[2][3]
Dòng	a[3][0]	a[3][1]	a[3][2]	a[3][3]

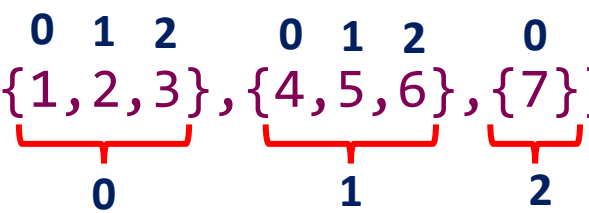
Mảng **a[4][4]** có 16 phần tử

5. MẢNG (ARRAY)

❖ Ví dụ 7

- `int arr1[][] = new int[2][3];` // Khai báo mảng 2 chiều arr1 kiểu int
- Cho biết số phần tử của mảng arr1.
- Liệt kê tất cả phần tử của mảng arr1 kèm với chỉ số. Ví dụ: `arr1[0][0] = 0, ...`
- Viết chương trình hiển thị các phần tử của mảng arr1.

❖ Ví dụ 8

- `int arr2[][] = {{1,2,3},{4,5,6},{7}};` // Khai báo mảng 2 chiều arr2 kiểu int

- Cho biết số phần tử của mảng arr2.
- Liệt kê tất cả phần tử của mảng arr2 kèm với chỉ số. Ví dụ: `arr2[0][0] = 1, ...`
- Viết chương trình hiển thị các phần tử của mảng arr2.

5. MẢNG (ARRAY)

❖ Ví dụ 9

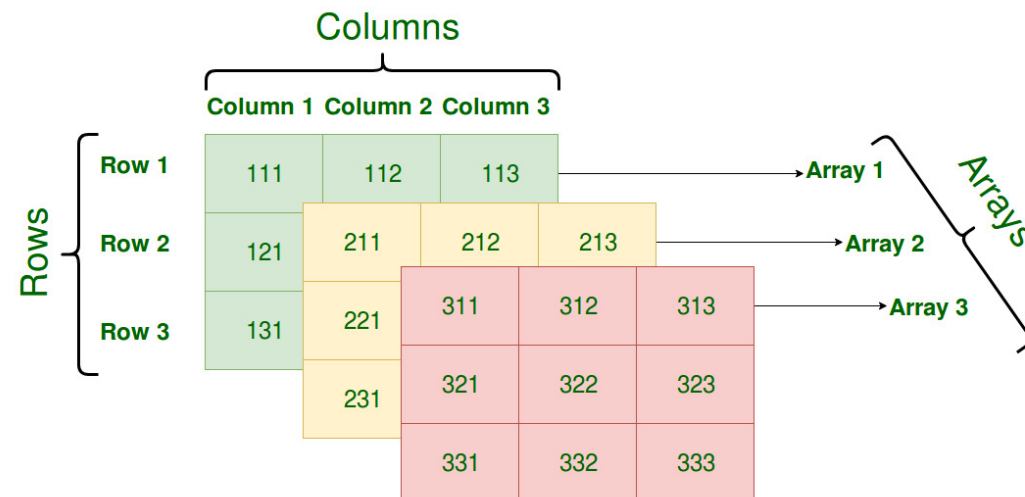
- `int arr3[][][] = {{{1,2,3},{4,5,6},{7}},{{5,6,7},{8,9}}};` // Khai báo mảng 3 chiều arr3 kiểu int
- Cho biết số phần tử của mảng arr3.
- Liệt kê tất cả phần tử của mảng arr3 kèm với chỉ số. Ví dụ: `arr3[0][0][0] = 1, ...`
- **Viết chương trình hiển thị các phần tử của mảng arr3.**

[array_index][row_index][column_index]

array_index: Chỉ số phần tử trong mảng 3 chiều

row_index: Chỉ số dòng mảng 2 chiều của phần tử tại array_index.

col_index: Chỉ số cột mảng 2 chiều của phần tử tại array_index.



5. MẢNG (ARRAY)

	0	1	2	3	4	5
Mảng 1 chiều a	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

		0	1	2	3	4	5
Mảng 2 chiều a	0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]
	1	a[1][0]	a[1][1]	a[1][2]			
chứa các mảng 1 chiều	2	a[2][0]	a[2][1]	a[2][2]	a[2][3]		
	3	a[3][0]					

Mảng 1 có 6 phần tử
Mảng 2 có 3 phần tử
Mảng 3 có 4 phần tử
Mảng 4 có 1 phần tử

		0	1	2
Mảng 3 chiều a	0	a[0][0]	a[0][1]	a[0][2]
	1	a[1][0]	a[1][1]	
chứa các mảng 2 chiều	2	a[2][0]		

	0	1
0	a[0][0]	a[0][1]
1	a[1][0]	

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

Mảng 1 có 3 phần tử
Mảng 2 có 2 phần tử
Mảng 3 có 1 phần tử

Mảng 2 chiều thứ 1

Mảng 1 có 2 phần tử
Mảng 2 có 1 phần tử

Mảng 2 chiều thứ 2

Mảng 3 chiều



Thời gian: 30 phút (17h20-17h50)

- `int a[] = {2,5,4,7,6};`
- `int b[][] = {{1,2,3},{4,5,6,8},{7},{8,9}};`
- `int c[][][] = {{{5,2,3},{4,5,6,8},{7,7}}},{{{5,6,7,3},{8,9}}};`

1. Hãy giải thích các khai báo cho ở trên. **(3 điểm)**
2. Liệt kê tất cả phần tử của mảng a, b, c kèm với chỉ số. Ví dụ: `a[0] = 2, ...;`
`b[0][0] = 1, ...;` `c[0][0][0] = 5, ...` **(3 điểm)**
3. Viết chương trình hiển thị các phần tử của mảng c. **(4 điểm)**

5. MẢNG (ARRAY)

❖ Sao
chép
mảng
sử
dụng
toán
tử gán

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        int arr[] = {3, 5, 8, 10, 14, 25}; //Khai báo mảng 1 chiều arr kiểu int  
        int cs = 0; //Biến lưu chỉ số mảng  
        int arr1[] = arr; //Gán mảng arr cho mảng arr1  
        System.out.printf("Phần tử thứ 1: arr1[%d] = %d", cs, arr[cs]);  
        System.out.printf("\nPhần tử thứ 2: arr1[%d] = %d", cs=cs+1, arr1[cs]);  
        System.out.printf("\nPhần tử thứ 3: arr1[%d] = %d", cs=cs+1, arr1[cs]);  
        System.out.printf("\nPhần tử thứ 4: arr1[%d] = %d", cs=cs+1, arr1[cs]);  
        System.out.printf("\nPhần tử thứ 5: arr1[%d] = %d", cs=cs+1, arr1[cs]);  
        System.out.printf("\nPhần tử thứ 6: arr1[%d] = %d", cs=cs+1, arr1[cs]);  
    }  
}
```

Console

<terminated> Example [Java Application]

Phần tử thứ 1: arr1[0] = 3
Phần tử thứ 2: arr1[1] = 5
Phần tử thứ 3: arr1[2] = 8
Phần tử thứ 4: arr1[3] = 10
Phần tử thứ 5: arr1[4] = 14
Phần tử thứ 6: arr1[5] = 25

5. MẢNG (ARRAY)

❖ Sao chép mảng sử dụng toán tử gán:

Thay đổi giá trị phần tử của mảng nguồn, thì mảng đích cũng thay đổi theo



```
public class Example {
    public static void main(String[] args) {
        //Các câu lệnh xử lý
        int arr[] = {3, 5, 8, 10, 14, 25}; // Khai báo mảng 1 chiều arr kiểu int
        int cs = 0; // Biến lưu chỉ số mảng
        int arr1[] = arr; // Gán mảng arr cho mảng arr1
        arr[0] = 9; // Thay đổi giá trị phần tử đầu tiên của mảng arr
        arr1[4] = 18; // Thay đổi giá trị phần tử thứ 5 của mảng arr1
        System.out.printf("Phần tử 1 arr: arr[%d] = %d | arr1: arr1[%d] = %d", cs, arr[cs], cs, arr1[cs]);
        System.out.printf("\nPhần tử 2 arr: arr[%d] = %d | arr1: arr1[%d] = %d", cs=cs+1, arr[cs], cs, arr1[cs]);
        System.out.printf("\nPhần tử 3 arr: arr[%d] = %d | arr1: arr1[%d] = %d", cs=cs+1, arr[cs], cs, arr1[cs]);
        System.out.printf("\nPhần tử 4 arr: arr[%d] = %d | arr1: arr1[%d] = %d", cs=cs+1, arr[cs], cs, arr1[cs]);
        System.out.printf("\nPhần tử 5 arr: arr[%d] = %d | arr1: arr1[%d] = %d", cs=cs+1, arr[cs], cs, arr1[cs]);
        System.out.printf("\nPhần tử 6 arr: arr[%d] = %d | arr1: arr1[%d] = %d", cs=cs+1, arr[cs], cs, arr1[cs]);
    }
}
```

Console

<terminated> Example [Java Application] C:\Program Files\Java\jdk-13\

```
Phần tử 1 arr: arr[0] = 9 | arr1: arr1[0] = 9
Phần tử 2 arr: arr[1] = 5 | arr1: arr1[1] = 5
Phần tử 3 arr: arr[2] = 8 | arr1: arr1[2] = 8
Phần tử 4 arr: arr[3] = 10 | arr1: arr1[3] = 10
Phần tử 5 arr: arr[4] = 18 | arr1: arr1[4] = 18
Phần tử 6 arr: arr[5] = 25 | arr1: arr1[5] = 25
```

Sao chép nông (shallow copy)

Sử dụng toán tử gán để sao chép mảng, kết quả mảng nguồn và mảng đích tham chiếu đến cùng một vị trí.

5. MẢNG (ARRAY)

- ❖ Sao chép mảng sử dụng vòng lặp: Tham khảo Chương 6
- ❖ Sao chép mảng sử dụng phương thức `arraycopy()`
 - Cú pháp: `System.arraycopy(arrSo, indexSo, arrDes, indexDes, num);`
 - *arrSo*: mảng nguồn; *arrDes*: mảng đích
 - *indexSo*: vị trí bắt đầu copy dữ liệu tại mảng nguồn
 - *indexDes*: vị trí bắt đầu tại mảng đích để copy dữ liệu đến
 - *num*: số phần tử muốn sao chép
- ❖ Sao chép mảng sử dụng phương thức `copyOfRange()`
 - Cú pháp: `arrDes = Arrays.copyOfRange(arrSo, indexBe, indexFi);`
 - *arrDes*: mảng đích; *arrSo*: mảng nguồn; *indexBe*: chỉ số bắt đầu; *indexFi*: chỉ số cuối (Không bao gồm phần tử có chỉ số cuối)

5. MẢNG (ARRAY)

❖ **Lưu ý:** Muốn sử dụng phương thức `toString()` và `copyOfRange()` phải import thêm package: *`import java.util.Arrays;`*

❖ **Ví dụ 10:**

- `int arrSo[] = {3, 5, 8, 10, 14, 25};`
- `int arrDes1[] = new int[5];`
- `int arrDes2[] = new int[5];`
- Thực hiện sao chép mảng `arrSo` sang mảng `arrDes1`: `indexSo = 0`, `indexDes = 0`, `num = 4`, `indexBe = 1`, `indexFi = 5`.
- Sử dụng `arraycopy()`: `System.arraycopy(arrSo, 0, arrDes1, 0, 4);`
- Sử dụng `copyOfRange()`: `arrDes2 = Arrays.copyOfRange(arrSo, 1, 5);`

5. MẢNG (ARRAY)

- ❖ Viết chương trình thực hiện sao chép dữ liệu từ mảng arrSo sang arrDes1 và arrDes2 theo ví dụ 10 sao cho kết quả hiển thị như hình bên dưới. Cho nhận xét.

arrDes1

Console

```
<terminated> Example [Java Application] C:\Program Files\Java\jdk-13\  
Kết quả sao chép arrSo -> arrDes1: [3, 5, 8, 10, 0]  
Số phần tử của mảng arrDes1: 5
```

arrDes2

Console

```
<terminated> Example [Java Application] C:\Program Files\Java\jdk-13\  
Kết quả sao chép arrSo -> arrDes2: [5, 8, 10, 14]  
Số phần tử của mảng arrDes2: 4
```

❖ Câu hỏi

- Trình bày khái niệm mảng?
- Trình bày cú pháp khai báo mảng?
- Phân biệt mảng 1 chiều với mảng đa chiều?
- Cho hai mảng: `int arr[15]` và `float arr1[4][4]`. Hãy giải thích?
- Cho biết cách xác định kích thước của mảng?
- Cho biết các xác định các phần tử của mảng 1 chiều?
- Cho biết cách xác định các phần tử của mảng 2 chiều?
- Cho biết các truy cập đến phần tử của mảng?

6. HÀM VÀ CHƯƠNG TRÌNH



Hàm hay phương thức (method) là một khối lệnh để thực hiện một hành động cụ thể.

Hàm là một đơn vị độc lập của chương trình. Tính độc lập của hàm thể hiện như sau:

- Không cho phép xây dựng hàm bên trong một hàm khác.
- Mỗi hàm có các biến, mảng, ... riêng của mình và chỉ được sử dụng nội bộ bên trong hàm. Hàm là đơn vị có tính chất khép kín.

❖ Cấu trúc hàm (method):

```
modifier returnType nameOfMethod (Parameter List) {  
    //method body  
}
```

- modifiers: từ khóa xác định phạm vi truy cập của hàm (public, private)
- returnType: kiểu dữ liệu trả về
- nameOfMethod: tên của hàm (method)
- Parameter List: các tham số đầu vào của hàm (có thể nhiều tham số với nhiều kiểu dữ liệu khác nhau)
- method body: các mã code bên trong hàm

❖ Hàm chia thành 2 loại:

- Có kết quả trả về (int, boolean, long, double, ...), trong hàm sử dụng từ khóa return để trả về kết quả cụ thể.

Cú pháp

```
modifier returnType nameOfMethod (Parameter List) {  
    //method body  
}
```

- Không có kết quả trả về (void).

Cú pháp

```
<modifier> void nameOfMethod(Parameter List) {  
    //body  
}
```

6. HÀM VÀ CHƯƠNG TRÌNH

❖ **Thao tác với hàm (method):** viết hàm (xây dựng hàm) và gọi hàm

- **Ví dụ 1:** Viết hàm tính tổng 2 số nguyên

```
//Hàm tính tổng hai số nguyên  
public static int tinhTong(int a, int b){  
    return a + b;  
}
```

- Gọi hàm: *tinhTong(10, 5); //Truyền tham số theo giá trị*
- **Ví dụ 2:** Viết hàm xuất ra dòng chữ “Hello world”

```
//Hàm xuất nội dung Hello world  
public static void hienThi(){  
    System.out.println("Hello world");  
}
```

- Gọi hàm: *hienThi(); //Gọi và sử dụng*
- *Nạp chồng phương thức: method overloading, method overriding (tham khảo)*

6. HÀM VÀ CHƯƠNG TRÌNH



Chương trình là một tập hợp các câu lệnh được viết theo thứ tự mà máy tính cần phải thực hiện. Trong ngôn ngữ Java, một chương trình là một tập hợp các lớp (class), mỗi lớp chứa một hoặc nhiều phương thức và bên trong chứa một tập các câu lệnh nhằm thực hiện một công việc cụ thể.

6. HÀM VÀ CHƯƠNG TRÌNH



Hàm `main()` là thành phần bắt buộc của chương trình. Chương trình bắt đầu thực hiện từ câu lệnh đầu tiên của hàm `main()` và kết thúc khi gặp dấu `}` cuối cùng của hàm này. Khi chương trình làm việc, máy tính có thể đi từ hàm này sang hàm khác.

6. HÀM VÀ CHƯƠNG TRÌNH

❖ Chương trình:

- Viết chương trình để tạo ra ứng dụng nhằm thực hiện nhiệm vụ cụ thể.
- Một ứng dụng có thể hoạt động đơn giản theo một cách cố định hoặc hoạt động theo nhiều cách khác nhau tùy thuộc vào thao tác của người dùng.
- Hoạt động của ứng dụng phụ thuộc hoàn toàn vào cách viết chương trình.

❖ Cấu trúc cơ bản của chương trình Java:

```
public class ClassName {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
    }  
}
```

```
package <package_name>;  
import <other_package>;  
public class ClassName {  
    <Variables (also known as fields)>;  
    <constructor method(s)>;  
    <other methods>;  
}
```

6. HÀM VÀ CHƯƠNG TRÌNH

❖ Ý nghĩa các thành phần trong chương trình Java: (^.-.^)

- `public class ClassName` khai báo một lớp (class) với tên `ClassName` do người dùng tạo. Tên lớp (class) này được dùng để đặt tên cho file `.java` chứa mã nguồn chương trình.
- `public static void main(String[] args)` là câu lệnh mặc định để khai báo **phương thức main** trong Java. **Phương thức main** là điểm bắt đầu chạy chương trình.
- `String[] args` là **đối số mặc định** trong **phương thức main**, có tác dụng nhập các chuỗi được phân cách bởi khoảng trắng từ thiết bị đầu cuối như bàn phím.

6. HÀM VÀ CHƯƠNG TRÌNH

❖ Ý nghĩa các thành phần trong chương trình Java: (^.-.^)

- Các **câu lệnh xử lý** được viết **giữa cặp dấu ngoặc nhọn {}**, mỗi câu lệnh được kết thúc **bởi dấu chấm phẩy ;** ;
- Các **cặp dấu ngoặc nhọn { }** cho biết **bắt đầu** và **kết thúc** một lớp (class) hoặc một phương thức, hoặc một khối lệnh.
- **public** cho biết phương thức được phép truy cập ở bất cứ nơi nào.
- **static** cho phép gọi **phương thức main** mà không cần phải khởi tạo **đối tượng main**.
- **void** là hàm không có giá trị trả về.

6. HÀM VÀ CHƯƠNG TRÌNH

❖ **Package** là một cơ chế để phân loại và nhóm các class, API sử dụng trong chương trình Java. (Vì ngôn ngữ Java được xây dựng trên nền tảng class, để viết một chương trình Java cần phải tạo ra nhiều lớp khác nhau không trùng tên).

❖ **Lưu ý:**

- Package tự xây dựng
- Hoặc sử dụng package có sẵn

Chức năng	Tên Package
Lớp cơ sở	java.lang
Truy cập	java.io, java.nio, java.net
Tính toán số học	java.math
Graphic	java.awt, javax.swing
Xử lý text	java.text
Xử lý hình ảnh	java.awt.image, javax.imageio
Bảo mật	java.security, javax.crypto
Cơ sở dữ liệu	java.sql
Script	java.script
Applet	java.applet
JavaBeans	java.beans

6. HÀM VÀ CHƯƠNG TRÌNH

❖ **Import** là từ khóa sử dụng để nạp các Package chứa class cần dùng vào chương trình Java.

- Cú pháp **import từng class** trong package: **import PackageName.ClassName;**
- Cú pháp **import toàn bộ package**: **import PackageName.*;**
- Trong đó **PackageName** là tên của Package cần import, và **ClassName** là tên class chứa trong Package, ***** toàn bộ class của Package cần nạp để sử dụng trong chương trình.

❖ Import trong Java

```
//Nạp package vào chương trình
import java.util.Scanner;
public class Example {
    private static Scanner sc;
    public static void main(String[] args) {
        //Các câu lệnh xử lý
        sc = new Scanner(System.in);
        System.out.print("Nhập dữ liệu: ");
        sc.next();
    }
}
```

6. HÀM VÀ CHƯƠNG TRÌNH

- ❖ **class** là từ khóa nhằm để định nghĩa lớp của Java. Nó đứng trước khai báo tên lớp của Java. Ngoài ra còn có từ khóa public, từ khóa này xác định phạm vi truy cập của lớp. Đặc tính này chính là tính đóng gói trong lập trình hướng đối tượng. (chúng ta sẽ tìm hiểu phần này ở học phần [Lập trình hướng đối tượng](#))
- ❖ **variables**: Biến hay còn gọi là trường, cũng có một số tài liệu gọi là thuộc tính trực thuộc lớp. Nó chứa thông tin cụ thể liên quan tới các đối tượng là thể hiện của lớp.
- ❖ **methods**: Phương thức hay còn gọi là hàm chứa các hành động thực thi của đối tượng. đương nhiên nội dung của phương thức chính là các đoạn mã thực thi của chính phương thức này.
- ❖ **constructors**: Phương thức khởi tạo (hàm khởi tạo) của đối tượng. Hình dạng của đối tượng được thể hiện ra sao sẽ phụ thuộc vào phương thức này.

❖ Quy tắc viết chú thích (comment) trong chương trình Java:

- Chỉ viết comment khi cần thiết, không viết thông tin thừa.
- Viết comment trước hàm và biến global.
- Nội dung comment phải đúng và không mâu thuẫn với code.
- Viết comment rõ ràng dễ hiểu, không gây nhiễu hoặc rối cho người đọc

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
    }  
}
```

6. HÀM VÀ CHƯƠNG TRÌNH

- ❖ **Ví dụ 3:** Viết chương trình tính tổng 2 số nguyên
- ❖ **Ví dụ 4:** Viết chương trình hiển thị nội dung “Hello world”

```
public class Example {  
    //Hàm tính tổng 2 số nguyên  
    public static int tinhTong(int a, int b){  
        return a + b;  
    }  
    public static void main(String[] args) {  
        //Gọi hàm và truyền tham số theo giá trị  
        System.out.println("Kết quả tính tổng của 10 và 5 bằng " + tinhTong(10,5));  
    }  
}
```

```
public class Example {  
    //Hàm xuất nội dung Hello world  
    public static void hienThi(){  
        System.out.println("Hello world");  
    }  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        hienThi(); //Gọi hàm sử dụng  
    }  
}
```

7. BIẾN, MẢNG TỰ ĐỘNG

- ❖ Các biến (mảng) khai báo bên trong thân của một hàm (kể cả hàm main) gọi là biến (mảng) tự động hay cục bộ. Các đối của hàm cũng được xem là biến tự động.
- ❖ **Phạm vi hoạt động:** Các biến (mảng) tự động chỉ có tác dụng bên trong thân của hàm mà tại đó chúng được khai báo.
- ❖ **Thời gian tồn tại:** Các biến (mảng) tự động của một hàm sẽ tồn tại (được cấp phát bộ nhớ) trong khoảng thời gian từ khi máy bắt đầu làm việc với hàm đến khi máy ra khỏi hàm.
- ❖ **Nhận xét:** Do chương trình bắt đầu làm việc từ câu lệnh đầu tiên của hàm main() và khi máy ra khỏi hàm main() thì chương trình kết thúc, nên các biến, mảng khai báo trong hàm main() sẽ tồn tại trong suốt thời gian làm việc của chương trình.

7. BIẾN, MẢNG TỰ ĐỘNG

- ❖ Biến, mảng tự động chưa được khởi đầu thì giá trị của chúng là hoàn toàn không xác định.

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        int a[];  
        float b;  
        System.out.printf("kết quả %d, %f",a[1],b);  
    }  
}
```

- ❖ Đoạn chương trình trên là vô nghĩa và không thể làm việc được, vì nó thực hiện ý định đưa ra màn hình giá trị của phần tử a[1] và biến b trong khi cả hai đều chưa được khởi đầu.

8. BIẾN, MẢNG NGOÀI

- ❖ Biến (mảng) khai báo bên ngoài các hàm gọi là biến (mảng) ngoài.
- ❖ **Thời gian tồn tại:** Biến (mảng) ngoài sẽ tồn tại (được cấp phát bộ nhớ) trong suốt thời gian làm việc của chương trình.
- ❖ **Phạm vi sử dụng:** Phạm vi hoạt động của biến (mảng) ngoài là từ vị trí khai báo của chúng cho đến cuối tệp chương trình. Nếu một biến (mảng) ngoài được khai báo ở đầu chương trình (đứng trước tất cả các hàm) thì nó có thể sử dụng trong bất kỳ hàm nào miễn là hàm đó không có các biến tự động trùng với biến (mảng) ngoài này.

9. BIẾN TĨNH, MẢNG TĨNH

- ❖ Để khai báo một biến (mảng) tĩnh sử dụng thêm từ khóa static ở đằng trước. Ví dụ:
static int num, arr[];
- ❖ Từ khóa static trong Java được sử dụng nhằm mục đích chính là quản lý bộ nhớ.
- ❖ Có thể sử dụng từ khóa static với biến, phương thức, khối hoặc các lớp lồng nhau.
- ❖ Từ khóa static thuộc về lớp chứ không thuộc về thể hiện của lớp.
- ❖ Biến static có thể được sử dụng để tham chiếu thuộc tính chung của tất cả đối tượng (mà không là duy nhất cho mỗi đối tượng).
- ❖ Biến static lấy bộ nhớ chỉ một lần trong phạm vi lớp (Class Area) tại thời gian tải lớp đó.
- ❖ Sử dụng biến static giúp chương trình sử dụng bộ nhớ hiệu quả hơn (tiết kiệm bộ nhớ).

9. BIẾN TĨNH, MẢNG TĨNH

❖ **Ví dụ 1:** không sử dụng biến tĩnh

```
public class Example {  
    int dem = 0; //lấy bộ nhớ khi instance được tạo ra  
    Example() {  
        dem++;  
        System.out.println(dem);  
    }  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        Example exa1 = new Example();  
        Example exa2 = new Example();  
        Example exa3 = new Example();  
    }  
}
```



Console

<terminated> Example

1

1

1

9. BIẾN TĨNH, MẢNG TĨNH

❖ **Ví dụ 2:** sử dụng biến tĩnh

```
public class Example {  
    static int dem = 0; //lấy bộ nhớ chỉ một lần  
    Example() {  
        dem++;  
        System.out.println(dem);  
    }  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        Example exa1 = new Example();  
        Example exa2 = new Example();  
        Example exa3 = new Example();  
    }  
}
```




Console ✕
<terminated> Example
1
2
3

9. BIẾN TĨNH, MẢNG TĨNH

❖ **Ví dụ 3:** không sử dụng mảng tĩnh

```
public class Example {  
    int arr[] = {1,2,3}; //lấy bộ nhớ khi instance được tạo ra  
    Example() {  
        arr[0]++;  
        System.out.println(arr[0]);  
    }  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        Example exa1 = new Example();  
        Example exa2 = new Example();  
        Example exa3 = new Example();  
    }  
}
```




Console ✕
<terminated> Example
2
2
2

9. BIẾN TĨNH, MẢNG TĨNH

❖ Ví dụ 4: sử dụng mảng tĩnh

```
public class Example {  
    static int arr[] = {1,2,3}; //lấy bộ nhớ chỉ một lần  
    Example() {  
        arr[0]++;  
        System.out.println(arr[0]);  
    }  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
        Example exa1 = new Example();  
        Example exa2 = new Example();  
        Example exa3 = new Example();  
    }  
}
```



Console

<terminated> Example

2

3

4

❖ Câu hỏi

- Phân biệt giữa hàm và chương trình?
- Phân biệt giữa hàm có giá trị trả về và hàm không có giá trị trả về?
- Phân biệt lời gọi hàm có giá trị trả về và lời gọi hàm không có giá trị trả về?
- Quy tắc đặt tên hàm (method)?
- Cho biết ý nghĩa của việc chú thích cho mã nguồn (comments)?
- Phân biệt giữa biến, mảng tĩnh với biến, mảng ngoài và biến, mảng tự động?
- Phân biệt giữa overriding với overloading? (***)