Bài 3
Mảng và phương thức trong Java

Module: ADVANCED PROGRAMMING WITH JAVA

Mục tiêu

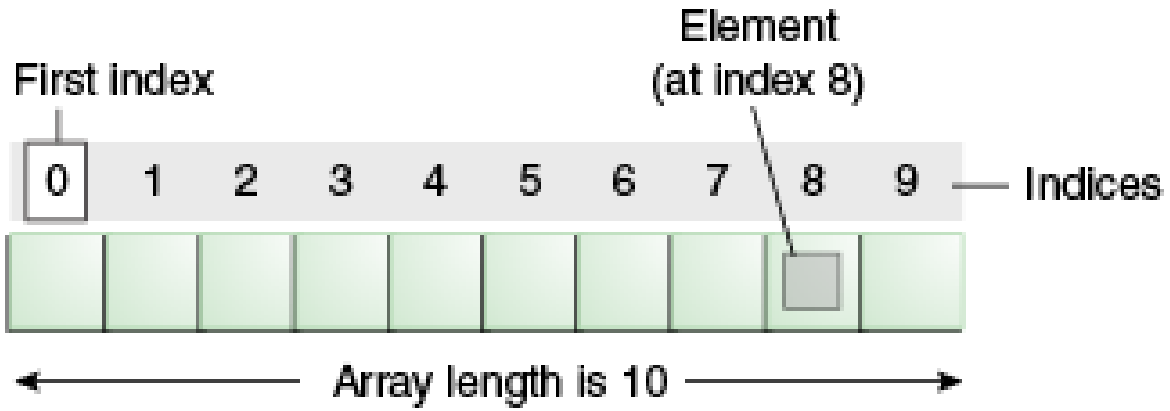
- Mô tả được cú pháp khai báo và sử dụng mảng
- Sử dụng được cú pháp Java để thao tác với phương thức

Thảo luận

Mảng

Mảng

- Mảng là một biến tham chiếu đến một loạt giá trị liên tiếp nhau
- Các giá trị được lưu trữ trong mảng có cùng kiểu dữ liệu
- Các khái niệm của mảng:
  - Tên mảng: Tuân thủ theo quy tắc đặt tên của biến
  - Phần tử: Các giá trị được lưu trữ trong mảng
  - Chỉ số: Vị trí của các phần tử trong mảng. Chỉ số bắt đầu từ 0.
  - Độ dài: Số lượng tối đa các phần tử mà mảng có thể lưu trữ



Khai báo mảng

- Cú pháp:

elementType[] arrayRefVar;

Trong đó:

- elementType: Kiểu dữ liệu của các phần tử trong mảng
- arrayRefVar: Tên của mảng

- Ví dụ, khai báo một mảng có tên **myList** lưu trữ giá trị kiểu double:

```
double[] myList;
```

Lưu ý: Có thể sử dụng cú pháp *elementType arrayRefVar[]* để khai báo mảng.

Biến mảng là biến tham chiếu

- Khi khai báo các biến kiểu dữ liệu nguyên thuỷ thì chúng được cấp phát bộ nhớ tương ứng để lưu trữ dữ liệu SAI
- Khi khai báo biến mảng thì sẽ không có việc cấp phát bộ nhớ ngay cho các phần tử của mảng. Chỉ có việc cấp phát bộ nhớ cho tham chiếu đến mảng SAI

khởi tạo

- Nếu không gắn tham chiếu đến mảng thì giá trị của biến mảng là **null** SAI
- Không thể gán các phần tử cho mảng nếu chưa khởi tạo mảng

Khởi tạo mảng

- Sử dụng từ khoá new để khởi tạo mảng:

arrayRefVar = ***new*** *elementType[arraySize];*

Câu lệnh trên thực hiện 2 việc:

1. ***new*** *elementType[arraySize]*: Khởi tạo một mảng mới
2. Gán tham chiếu của mảng vừa tạo cho biến *arrayRefVar*

- Có thể gộp chung việc khai báo mảng, khởi tạo mảng và gán tham chiếu cho biến mảng:

elementType[] arrayRefVar = ***new*** *elementType[arraySize];*

Hoặc:

elementType arrayRefVar[] = ***new*** *elementType[arraySize];*

Ví dụ khởi tạo mảng

- Khai báo và khởi tạo một mảng **double** tên là **myList** có thể chứa 10 phần tử:

```
double[] myList = new double[10];
```

Hoặc:

```
double myList[] = new double[10];
```

- Khai báo và khởi tạo một mảng String tên là names có thể chứa 30 phần tử:

```
String[] names = new String[30];
```

Hoặc:

```
String names[] = new String[30];
```

Gán giá trị cho các phần tử mảng

- Cú pháp:

arrayRefVar[index] = value;

Trong *đó*:

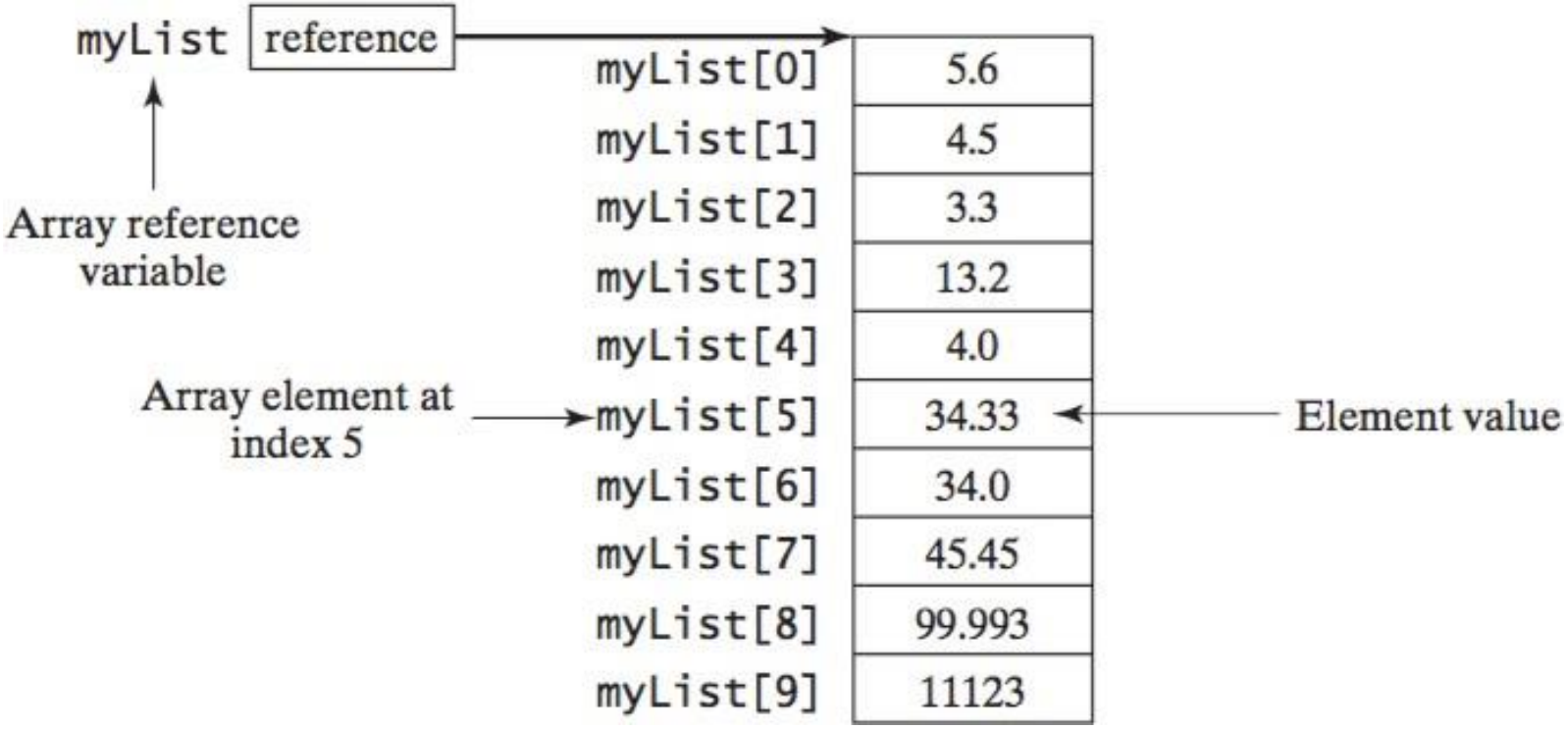
- *index*: Chỉ số (vị trí) của phần tử muốn gán giá trị. Chỉ số của phần tử đầu tiên là 0
- *value*: Giá trị muốn gán cho phần tử tại vị trí *index*

Lưu ý: Một mảng sau khi được khởi tạo mà không được gán giá trị cho các phần tử của nó thì các phần tử sẽ có giá trị mặc định tuỳ theo kiểu dữ liệu của mảng. *(0, false, null, \u0000)*

Ví dụ gán giá trị cho phần tử mảng

```
double[] myList = new double[10];
```

```
myList[0] = 5.6;
myList[1] = 4.5;
myList[2] = 3.3;
myList[3] = 13.2;
myList[4] = 4.0;
myList[5] = 34.33;
myList[6] = 34.0;
myList[7] = 45.45;
myList[8] = 99.993;
myList[9] = 11123;
```



Phân biệt đối tượng mảng với biến mảng

- *Đối tượng mảng* (array object) là khác với biến mảng (array variable)
- Ví dụ, với khai báo:

```
double[] myList = new double[10];
```

- Chúng ta cần hiểu: *myList là một biến chứa tham chiếu đến một mảng double có 10 phần tử*
- Tuy nhiên, thông thường chúng ta cũng có thể nói ngắn gọn: *myList là một mảng double chứa 10 phần tử*

Lưu ý: Việc phân biệt biến tham chiếu và đối tượng là rất quan trọng. Vấn đều này sẽ được đề cập đến trong phần về đối tượng.

Độ dài của mảng

- Khi khởi tạo một mảng thì cần quy định độ dài (length) của mảng đó
- Độ dài của mảng là số lượng phần tử tối đa mà mảng có thể chứa
- Độ dài của mảng giúp máy tính biết dung lượng bộ nhớ cần cấp phát (allocate) cho mảng
- Độ dài của mảng còn được gọi là kích thước (size) của mảng
- Không thể thay đổi kích thước của mảng sau khi đã khởi tạo
- Để lấy được độ dài của mảng thì sử dụng thuộc tính *length*

Ví dụ:

```
int x = myList.length; //x có giá trị là 10
```

Chỉ số của các phần tử mảng

- Chỉ số (index) của phần tử còn được gọi là vị trí (position) của phần tử đó
- Chỉ số của phần tử đầu tiên là *0*
- Chỉ số của phần tử cuối cùng là *n -1*, trong đó *n* là độ dài của mảng

Khởi tạo nhanh mảng

- Java cung cấp cú pháp ngắn gọn để khởi tạo nhanh mảng, còn được gọi là *array initializer*
- Cú pháp:

elementType[] arrayRefVar = {value0, value1, …, valuek};

- Ví dụ:

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

Demo

Tạo mảng

Duyệt mảng

Thảo luận

Duyệt mảng

Sử dụng vòng lặp for

- Sử dụng vòng lặp for để duyệt qua tất cả các phần tử của mảng

```
int[] mylist = {1,3,5,7,9};
for (int i=0; i< mylist.length; i++){
    System.out.println(mylist[i]);
}
```

Sử dụng for-each

- Sử dụng vòng lặp foreach để duyệt qua tất cả các phần tử của mảng

```
for (int item: mylist){
    System.out.println(item);
}
```

Thảo luận

Phương thức

Phương thức

- Phương thức (method) là một nhóm các câu lệnh thực hiện một nhiệm vụ nhất định
- *Phương thức* là thuật ngữ được sử dụng phổ biến trong Lập trình hướng đối tượng. Trong nhiều trường hợp khác, các tên gọi được sử dụng là *hàm* (function) và *thủ tục* (procedure)
- System.***out***.println(), Math.*pow*(), Math.*random*() là các phương thức đã được định nghĩa sẵn cho chúng ta sử dụng

Lưu ý: Mặc dù tên gọi phương thức, hàm, procedure đôi khi có thể sử dụng thay thế cho nhau, nhưng giữa 3 khái niệm này có sự khácnhau.

Khai báo phương thức

- Cú pháp:

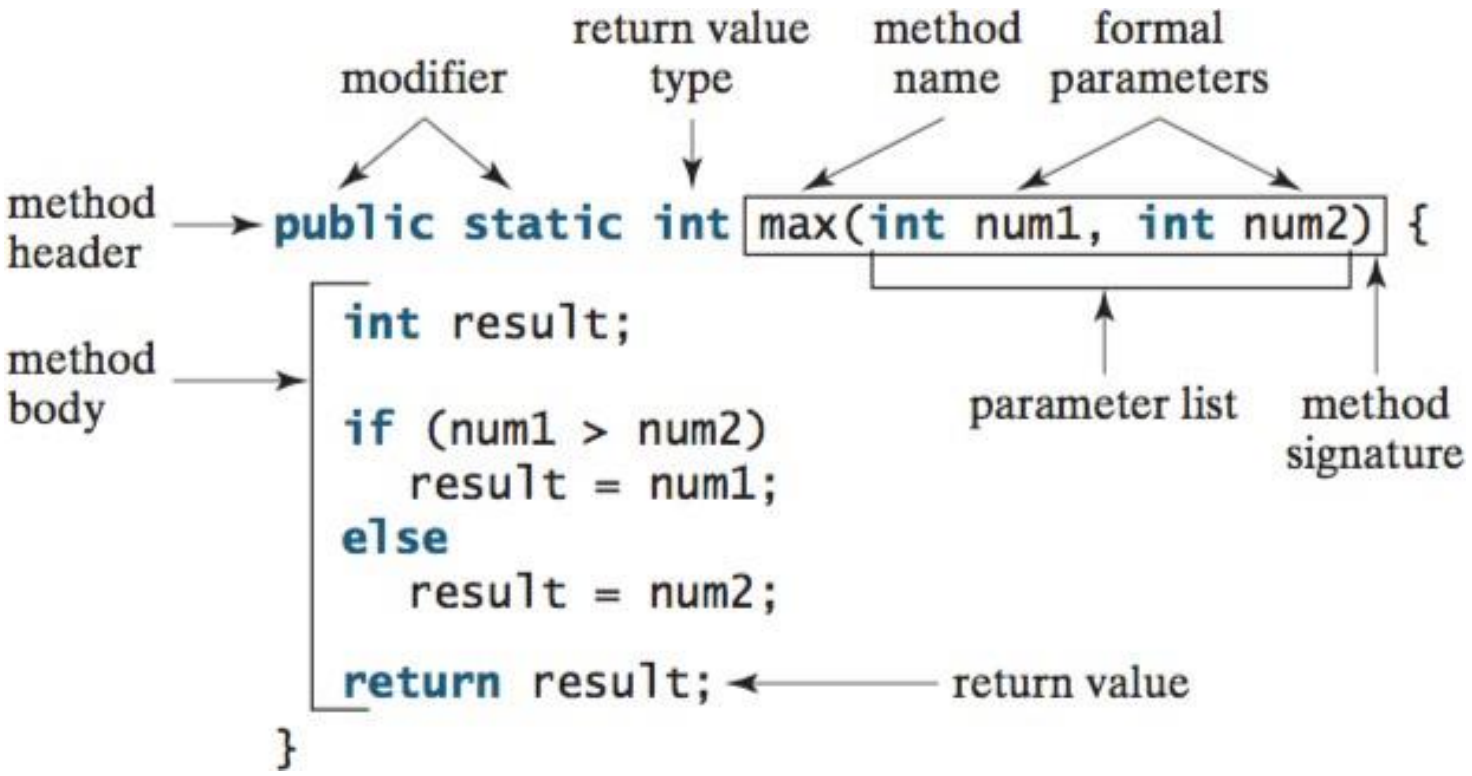
```
modifier returnValueType methodName(listof parameters) {
   // Method body;
}
```

Trong đó:

- modifier có thể là các từ khoá để quy định các tính chất khác nhau của phương thức
- returnValueType là kiểu dữ liệu trả về của phương thức
- methodName là tên gọi của phương thức
- list of parameters là danh sách các tham số của phương thức
- Method body là phần thân của phương thức

Ví dụ: Cấu phần của một phương thức

- Phương thức xác định số lớn nhất trong 2 số:



Kiểu dữ liệu trả về

- Một phương thức có thể trả về một giá trị
- Nếu phương thức không trả về giá trị thì có kiểu dữ liệu trả về **void**
- Ví dụ, System.*out*.println() là một phương thức void
- Ví dụ, phương thức kiểm tra số chẵn có kiểu dữ liệu trả về là boolean:

```
public static boolean isEven(int number){
    return number % 2 == 0;
}
```

Tham số (parameter) và đối số (argument)

- Tham số (còn *được gọi đầy đủ* là tham số hình thức – formal parameter) là các biến *được* khai báo trong phần header
- Khi gọi phương thức thì giá trị của các biến này sẽ *được* truyền vào. Các giá trị này *được* gọi là tham số thực (actual parameter) hoặc *đối số* (argument)
- Ví dụ:

parameter

```
public static boolean isEven(int number){
    return number % 2 == 0;
}
```

argument

```
isEven(5);
```

Gọi phương thức

- Gọi (*call* hoặc *invoke*) phương thức là cách để thực thi một phương thức đã được định nghĩa trước đó
- Khi gọi phương thức thì cần truyền đối số vào
- Ví dụ, gọi phương thức void:

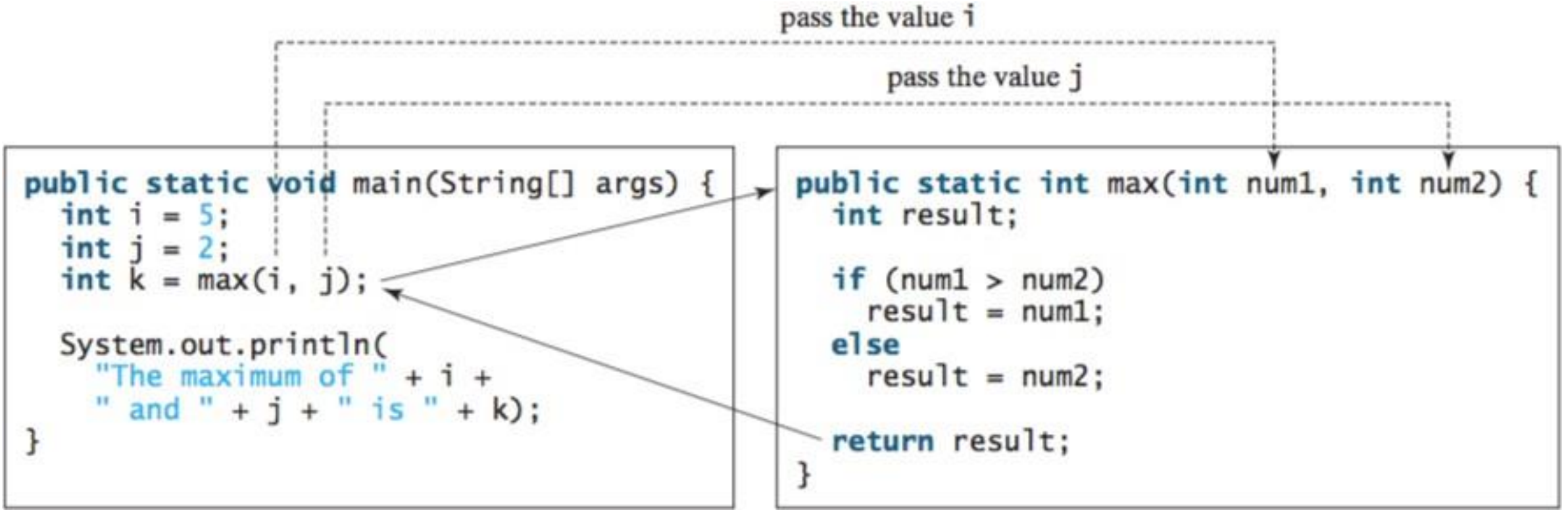
```
System.out.println("Welcome to Java!");
```

- Ví dụ, gọi phương thức có giá trị trả về:

```
int larger = max(3, 4);
```

Luồng điều khiển (control flow)

- Khi gọi phương thức, luồng điều khiển được chuyển cho phương thức đó



Phương thức main()

- Phương thức main() là một phương thức đặc biệt trong Java
- Phương thức main() là điểm khởi đầu (entry point) cho một chương trình
- Phương thức main() được gọi bởi JVM
- Header của phương thức main() được quy định sẵn

```
public static void main(String[] args){

}
```

Demo

Sử dụng phương thức

Tóm tắt bài học

- Các khái niệm của mảng: Tên mảng, kiểu dữ liệu, kích thước, phần tử, chỉ số
- Tên của mảng tuân theo quy tắc của tên biến
- Chỉ số của phần tử đầu tiên là 0
- Chỉ số của phần tử cuối cùng là length – 1
- Có thể sử dụng vòng lặp for và for-each để duyệt mảng

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: *Lớp và đối tượng*