**Khái niệm**

* Phương thức hiểu đơn giản là tập hợp của các câu lệnh cùng thực hiện trọn vẹn một chức năng được đặt dưới 1 cái tên chung.
* Mỗi phương thức chỉ thực hiện một nhiệm vụ. Hay còn gọi tác vụ, task. Do đó phương thức thường dùng thể hiện hành động của một đối tượng nào đó. Hoặc dùng để mô tả việc thực hiện một nhiệm vụ cụ thể.
* Không nhồi nhét nhiều hơn 1 nhiệm vụ cho một phương thức. Không nên tạo những phương thức quá dài, ví dụ trên 40 dòng code.

Ví dụ 1: phương thức main, thứ đã quen thuộc ngay từ đầu.

public class Lesson18 {

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

**System.*out*.println("Hello World!");**

**}**

}

Ví dụ 2: Phương thức kiểm tra số nguyên tố:

**public static boolean isPrime(int n) {**

**if (n < 2) {**

**return false;**

**}**

**for (int i = 2; i <= Math.*sqrt*(n); i++) {**

**if (n % i == 0) {**

**return false;**

**}**

**}**

**return true;**

**}**

**Cú pháp tổng quát**

Cú pháp liên quan đến phương thức bao gồm cú pháp định nghĩa phương thức và cú pháp gọi/sử dụng phương thức:

**Định nghĩa phương thức:**

access non-access type name(params) {

*// statements*

}

* Trong đó:
  + Phần access là access modifier, hiểu là cấp độ truy cập hay khả năng phương thức được nhìn thấy từ bên ngoài lớp. Các access modifier sẽ được trình bày cụ thể trong phần chương kế tiếp nói về OOP. Để cho đơn giản hiện tại ta cho phần này là public: ở đâu cũng nhìn thấy và sử dụng được.
  + Phần non-access là phần có thể có hoặc không. None-access modifier có nhiệm vụ làm rõ tính chất của một phương thức. Chúng ta cũng sẽ tìm hiểu dần trong các bài sau. Hiện tại bạn cứ mặc định phần này là static. Ý nghĩa là cho phép truy cập không cần đối tượng của lớp(vì đối tượng ta chưa học).
  + Phần type là kiểu của phương thức. Có thể là bất kì kiểu hợp lệ nào trong ngôn ngữ Java như các kiểu nguyên thủy, void, String, các struct, enum, class, interface… Mỗi phương thức chỉ có thể có 1 kiểu.
  + Phần name là tên phương thức. Tên của phương thức là các hành động, nhiệm vụ nên nó thường là động từ hoặc cụm động từ. Trừ tên các phương thức đặc thù ví dụ như của lớp Math là ngoại lệ. Tên phương thức bắt đầu với chữ cái thường. Nếu tên có nhiều từ thì các từ viết liền và viết hoa chữ cái đầu từ. Tên phương thức đặt sao cho nó thể hiện hành động mà phương thức đảm nhiệm.
  + Sau tên phương thức là cặp ngoặc tròn. Đây là dấu hiệu nhận biết phương thức. Trong ngoặc tròn chứa danh sách các tham số.
  + Phần params là danh sách tham số, làm nhiệm vụ nhận dữ liệu truyền vào khi thực hiện lời gọi phương thức. Phần tham số có thể có 0, 1 hoặc nhiều tham số. Nếu có nhiều tham số thì sẽ phân tách nhau bởi dấu phẩy. Tham số là các biến hoặc hằng số có tên, kiểu cụ thể.
  + Phần thân phương thức bao bởi cặp {} chứa các câu lệnh cùng thực hiện chức năng mà phương thức đảm nhiệm.
  + Nếu phương thức trả về giá trị, kiểu của phương thức khác void và trong thân phương thức có lệnh return giá\_trị; Nếu phương thức không trả về thì kiểu của phương thức là void.
* Ví dụ:

*// phương thức trả về kiểu boolean, nhận 1 tham số*

public static boolean isPrime(int n) {

    if (n < 2) {

        return false; *// trả về false*

}

    for (int i = 2; i <= Math.*sqrt*(n); i++) {

        if (n % i == 0) {

            return false; *// trả về false*

}

    }

    return true; *// trả về true*

}

*// phương thức không trả về, không có tham số*

public static void move() {

    System.*out*.println("Tôi đang bò trên mặt đất");

}

**Gọi phương thức để sử dụng.**

* Cú pháp:

name (params);

*// nếu muốn lưu lại giá trị trả về từ phương thức:*

variable = name (params);

* Trong đó:
  + Lời gọi phương thức cần cung cấp đầy đủ tên và danh sách, số lượng, thứ tự đối số.
  + Phần name là tên phương thức cần gọi.
  + Đi sau tên là cặp ngoặc tròn kết thúc bởi dấu chấm phẩy ;
  + Trong ngoặc tròn là danh sách, thứ tự đối số.
  + Khi định nghĩa phương thức thì thứ trong ngoặc tròn sau tên hàm là tham số. Tham số là khai báo biến.
  + Khi gọi phương thức thì thứ trong ngoặc tròn của lời gọi là đối số. Đối số có thể là biến, hằng số, đối tượng hoặc giá trị cụ thể nào đó.
  + Số lượng đối số phải tương ứng với số lượng tham số.
* Sau đây là 1 số lời gọi phương thức:

public static void main(String[] args) {

*// gọi phương thức khởi tạo của lớp scanner*

var input = new Scanner(System.*in*); *// gán giá trị có đc cho input*

*// gọi phương thức đọc vào giá trị số nguyên:*

int i = input.nextInt(); *// gán giá trị nhận đc cho i*

*// gọi phương thức hiển thị dữ liệu*

System.*out*.println("Hello World!"); *// chỉ hiển thị k gán cho biến nào*

*// gọi phương thức kiểm tra số ng.tố, gán giá trị cho biến prime*

boolean prime = *isPrime*(500); *// truyền vào đối số là 500*

*// gọi phương thức move:*

*move*(); *// không truyền gì vào*

}

**Kiểu trả về của phương thức**

Bây giờ ta sẽ làm rõ khái niệm trả về và không trả về.

* Một phương thức gọi là trả về nếu nó return một giá trị nào đó. Ví dụ phương thức nextInt() trả về giá trị int. Hoặc phương thức isPrime(500) trả về giá trị boolean. Trong thân phương thức có cú pháp return giá\_trị;
* Phương thức trả về có kiểu khác void. Ví dụ: int, float, boolean, String…
* Sử dụng phương thức trả về khi ta muốn đánh giá một tiêu chí nào đó; hoặc giá trị tính toán/có được từ phương thức đó sẽ là đầu vào hoặc được sử dụng bởi các thành phần thức khác của chương trình.
* Về bản chất phương thức trả về là phương thức gửi cho nơi đã gọi nó kết quả sau khi thực hiện nhiệm vụ. Như vậy ta biết được nhiệm vụ đó ra sao. Giống như đi thi cần biết kết quả thi để còn tính xem có phải học lại hay không vậy.
* Ví dụ như việc đọc vào một giá trị, tính tổng các chữ số, kiểm tra một số có phải số đẹp hay không…
* Phương thức không trả về là các phương thức có kiểu void. Phương thức không trả về có thể sử dụng return nhưng không có giá trị theo sau return.
* Sử dụng phương thức không trả về khi mà nhiệm vụ cần thực hiện không cần báo kết quả cho ai cả hoặc kết quả nhận được khi thực hiện nhiệm vụ là không quan trọng, không được sử dụng vào mục đích gì nữa.
* Ví dụ như phương thức hiển thị kết quả, hiện thông báo…

Tên phương thức kiểu boolean thường bắt đầu với is…

**Ví dụ minh họa**

import java.util.Scanner;

public class Lesson18 {

    public static void main(String[] args) {

*// gọi phương thức khởi tạo của lớp scanner*

var input = new Scanner(System.*in*); *// gán giá trị có đc cho input*

*// gọi phương thức đọc vào giá trị số nguyên:*

int i = input.nextInt(); *// gán giá trị nhận đc cho i*

*// gọi phương thức hiển thị dữ liệu*

System.*out*.println("Hello World!"); *// chỉ hiển thị k gán cho biến nào*

*// gọi phương thức kiểm tra số ng.tố, gán giá trị cho biến prime*

boolean prime = *isPrime*(500); *// truyền vào đối số là 500*

*// gọi phương thức move:*

*move*(); *// không truyền gì vào*

*// tính tổng hai số:*

int sum = *add*(100, 200); *// gán cho biến sum*

*// kiểm tra một số có phải số đẹp:*

boolean isBeautiNum = *isRevert*(1223221L); *// gán cho biến isBeautiNum*

}

*// phương thức tính tổng*

public static int add(int a, int b) {

        return a + b;

    }

*// phương thức kiểm tra số đẹp*

public static boolean isRevert(long n) {

        long m = n;

        long rev = 0;

        while (m > 0) {

            rev = rev \* 10 + m % 10;

            m /= 10;

        }

        return m == rev;

    }

*// phương thức trả về kiểu boolean, nhận 1 tham số*

public static boolean isPrime(int n) {

        if (n < 2) {

            return false; *// trả về false*

}

        for (int i = 2; i <= Math.*sqrt*(n); i++) {

            if (n % i == 0) {

                return false; *// trả về false*

}

        }

        return true; *// trả về true*

}

    public static void move() {

        System.*out*.println("Tôi đang bò trên mặt đất");

    }

}