Java Method - Phương thức trong Java

Khái niệm

Phương thức xác định giao diện cho phần lớn các lớp. Trong khi đó Java cho phép bạn định nghĩa các lớp mà không cần phương thức. Bạn cần định nghĩa phương thức truy cập dữ liệu mà bạn đã lưu trong một lớp.

Phương thức được định nghĩa như một hành động hoặc một tác vụ thật sự của đối tượng. Nó còn được định nghĩa như một hành vi mà trên đó các thao tác cần thiết được thực thi.

Cú pháp

```
access_specifier modifier datatype method_name(parameter_list)
{
//body of method
}
```

Trong đó:

- **access_specifier**: Chỉ định truy cập vào phương thức.
- **modifier**: Cho phép bạn gán các thuộc tính cho phương thức.
- **datatype**: Kiểu dữ liệu mà giá trị của nó được phương thức trả về. Nếu không có một giá trị nào được trả về, kiểu dữ liệu có thể là void.
- **method_name**: Tên của phương thức
- **parameter_list**: Chứa tên của tham số được sử dụng trong phương thức và kiểu dữ liệu. Dấu phẩy được dùng để phân cách các tham số.

```
class Temp {
    static int x = 10; //variable
    public static void show() //method
    {
        System.out.println(x);
    }
    public static void main(String args[]) {
        Temp t = new Temp(); // object 1
```

```
        t.show(); //method call
        Temp t1 = new Temp(); // object 2
        t1.x = 20;
        t1.show();
    }
}
```

Chỉ định truy cập phương thức - Access specifier method

Các chỉ định truy xuất dùng để giới hạn khả năng truy nhập vào một phương thức. Java cung cấp các chỉ định truy xuất sau đây:

- **Public:** Phương thức có chỉ định truy xuất public có thể được nhìn thấy từ mọi gói hoặc mọi lớp.
- **Protected:** Các lớp mở rộng từ lớp hiện hành trong cùng một gói, hoặc tại các gói khác nhau có thể truy cập các phương thức sử dụng chỉ định truy xuất này.
- **Private:** Phương thức riêng tư có thể được truy cập nhờ phương thức công cộng trên cùng một lớp.

Loại phương thức - Modifier method

Các bổ nghĩa loại phương thức cho phép ta thiết lập các thuộc tính của phương thức. Java cung cấp các bổ nghĩa sau:

- **Tĩnh (static):** Các trạng thái mà phương thức có thể được thay đổi mà không cần đến đối tượng. Nó chỉ được sử dụng đối với các dữ liệu và các phương thức tĩnh.
- **Trừu tượng (abstract):** Ngụ ý rằng phương thức không có một mã cụ thể (code) và nó sẽ được bổ sung ở các lớp con (subclass). Loại phương thức này được sử dụng trong các lớp kế thừa.
- **Kết thúc (final):** Phương thức không thể được thừa kế hoặc ghi đè (Overridden).
- **Tự nhiên (native):** Chỉ ra rằng phần thân của phương thức được viết trên các ngôn ngữ khác Java ví dụ C, hoặc C++.
- **Đồng bộ (synchronized):** Sử dụng với phương thức trong quá trình thực thi threads. Nó cho phép chỉ một thread được truy cập vào khối mã vào một thời điểm.
- **Linh hoạt (volatile):** Được sử dụng với các biến để thông báo rằng giá trị của biến có thể được thay đổi vài lần khi thực thi chương trình và giá trị của nó không được ghi vào thanh ghi.

Nạp chồng - Overloading method

Phương thức được nạp chồng (overload) là phương thức trong cùng một lớp, có cùng một tên song có danh sách các tham số khác nhau. Sử dụng việc nạp chồng phương thức để thực thi các phương thức giống nhau đối với các kiểu dữ liệu khác nhau. Ví dụ phương thức swap() có thể bị nạp chồng (overload) bởi các tham số của kiểu dữ liệu khác như integer, double và float.

Phương thức nạp chồng là một hình thức đa hình (polymorphism) trong quá trình biên dịch (compile).

Đoạn chương trình sau mô tả nạp chồng phương thức được thực hiện như thế nào:

```
//defined once
protected void perfomTask(double salary) {
    System.out.prinln("Salary is: "+salary);….
}
//overloaded –defined the second time with different parameters
protected void performTask(double salary, int bonus) {
    System.out.println("Total Salary is: " + salary + bonus);
}
```

Ghi đè - Overriding method

Phương thức được ghi đè (overriden) là phương thức có mặt ở lớp cha (superclasss) cũng như ở các lớp kế thừa. Phương thức này cho phép một lớp tổng quát chỉ định các phương thức sẽ là phương thức chung trong các lớp con. Ví dụ lớp xác định phương thức tổng quát 'area()'. Phương thức này có thể được hiện thực trong một lớp con để tìm diện tích một hình cụ thể như hình chữ nhật, hình vuông...

Phương thức ghi đè là một hình thức đa hình trong quá trình thực thi (runtime).

Các đoạn mã sau đây mô tả việc thực thi ghi đè phương thức trong Java:

```
class SupperClass // Tạo lớp cơ b□n
{
    int a;
    SuperClass() // constuctor
    {}
    SuperClass(int b) //overloaded  constructor
    {
        a = b;
    }
    class Subclass Extends SuperClass { // derriving a class
        int a;
        SubClass(int a) { //subclass constructor
            This.a;
        }
        public void message() { // overiding the base class message()
            System.out.prinln("In the sub class");
```

```
        }
    }
}
```

Bây giờ chúng ta sẽ tạo ra một đối tượng lớp cha và gán một lớp nhỏ tham chiếu đến nó như sau:

```
SuperClasss spObj = new Subclass(22);
```

Câu lệnh 'spObj.message' thuộc phương thức nhóm con. Ở đây kiểu đối tượng được gán cho 'spObj' sẽ chỉ được xác định khi chương trình thực thi. Điều này được biết dưới khái niệm 'liên kết động' (dinamic binding).

Phương thức khởi tạo lớp - Constructor method

Phương thức khởi tạo lớp là một loại phương thức đặc biệt rất khác với các kiểu khởi tạo cơ bản. Nó không có kiểu trả về. Nó có tên trùng với tên của lớp. Hàm khởi tạo lớp thực thi như một phương thức hoặc một chức năng bình thường song nó không trả về bất cứ một giá trị nào. Nói chung chúng được dùng để khởi tạo các biến thành viên của một lớp và nó được gọi bất cứ lúc nào bạn tạo ra đối tượng của lớp đó.

Phương thức khởi tạo lớp có hai loại:

- Tường minh (explicit): Bạn có thể lập trình những phương thức khởi tạo lớp khi định nghĩa lớp. Khi tạo một đối tượng của một lớp, những giá trị mà bạn truyền vào phải khớp với những tham số của phương thức khởi tạo (số lượng, thứ tự và kiểu dữ liệu của các tham số)
- Ngầm định (Implicit): Khi bạn không định nghĩa một hàm khởi tạo cho một lớp, JVM cung cấp một giá trị mặc định hay một phương thức khởi tạo ngầm định.

Bạn có thể định nghĩa nhiều phương thức khởi tạo cho một lớp. Giống như các phương thức khác, phương thức khởi tạo lớp có thể bị nạp chồng (overload)

Ví dụ một phương thức khởi tạo:

Đoạn mã sau đây định nghĩa một phương thức khởi tạo tường minh (explicit) cho một lớp Employee. Phương thức khởi tạo bao gồm tên và tuổi. Chúng được coi như các tham số và gán các giá trị của chúng vào các biến của lớp. Chú ý rằng từ khoá 'this' được sử dụng để tham chiếu đến đối tượng hiện hành của lớp.

```
Class Employee {
    String name;
    int age;
    Employee(String varname, int varage) {
        this.name = varname;
        this.age = varage;
```

```
    }
    public static void main(String arg[]) {
        Employee e = new Employee("Allen", 30);
    }
}
```

Revision #2
Created 26 September 2019 16:08:42 by Laptrinh.vn
Updated 12 April 2020 14:38:53 by Laptrinh.vn