


Bài 28: Các constructor

- ✓ Mục đích sử dụng
- ✓ Cú pháp tổng quát
- ✓ Các đặc trưng
- ✓ Sử dụng keyword this
- ✓ Cấm một lớp tạo đối tượng
- ✓ Minh họa & bài tập thực hành



Mục đích sử dụng

- 
- Tạo đối tượng với những thông tin ban đầu nào đó
 - Mọi lớp muốn tạo đối tượng đều phải có phương thức khởi tạo với access modifier khác private
 - Không sử dụng phương thức khởi tạo với các mục đích khác

Cú pháp tổng quát

➤ Cú pháp:

```
access name(params) {  
    // các câu lệnh cần thực hiện  
}
```

➤ Trong đó:

- Phần access là access modifier, có thể là public, protected, private hoặc để trống.
- Ý nghĩa của access modifier là giới hạn khả năng được nhìn thấy và sử dụng của một thành phần với thế giới bên ngoài lớp
- Phần name là tên phương thức khởi tạo, tên phải giống y hệt tên của lớp
- Phần params trong ngoặc tròn là danh sách tham số của constructor. Một constructor có thể có 0, 1 hoặc nhiều tham số

Cú pháp tổng quát

➤ Cú pháp:

```
access name(params) {  
    // các câu lệnh cần thực hiện  
}
```

➤ Trong đó:

- Thân phương thức nằm trong cặp {} chứa các câu lệnh thực hiện việc gán giá trị cho các thuộc tính và gọi các phương thức cần thiết khác

Ví dụ

```
public class Car {  
    public String name;  
    public int year;  
    public float weight;  
    // phương thức khởi tạo không tham số  
    public Car() {  
  
    }  
    // phương thức khởi tạo 1 tham số  
    public Car(String name) {  
        this.name = name;  
    }  
    // phương thức khởi tạo 3 tham số  
    public Car(String name, int year, float weight) {  
        this.name = name;  
        this.year = year;  
        this.weight = weight;  
    }  
}
```

Lưu ý

- Nếu các thuộc tính không được khởi tạo thì chúng sẽ được gán giá trị mặc định tương ứng của kiểu:
 - Các kiểu số là 0
 - Kiểu boolean là false
 - Các kiểu tham chiếu là null

Các đặc trưng

- Phần access của constructors chỉ có thể chứa tối đa 1 keyword hoặc là bạn sẽ bị báo lỗi
- Phương thức khởi tạo có thể được nạp chồng
- Phương thức khởi tạo không có kiểu trả về
- Không thực hiện lời gọi đệ quy tới chính nó
- Phương thức khởi tạo không thể là final, abstract, static, native, strictfp hay synchronized
- Một lớp luôn có một phương thức khởi tạo mặc định không tham số nếu bạn không định nghĩa bất kì phương thức khởi tạo nào
- Phương thức khởi tạo mặc định luôn có cùng access modifier với access modifier của lớp chứa nó

Ví dụ

```
public class Car {  
    public String name;  
}
```

// tương đương với:

```
public class Car {  
    public String name;  
    // phương thức khởi tạo mặc định  
    public Car() {  
        super();  
    }  
}
```

// do đó ta có thể thực hiện việc tạo đối tượng của lớp Car:
Car myCar = new Car(); *// ok*

Keyword this

- Về bản chất, this là keyword đại diện cho đối tượng hiện thời. Đây là đối tượng đang thực hiện các hành động ở thời điểm hiện tại
- Sử dụng this trong trường hợp tên tham số trùng tên thuộc tính. Keyword this sẽ giúp phân biệt đâu là thuộc tính và đâu là tham số
- Sử dụng this để gọi tới 1 phương thức khởi tạo khác trong cùng lớp để tái sử dụng code
- Sử dụng this trong các lời gọi mà đối tượng cần thao tác là đối tượng hiện thời



Keyword this

- Không áp dụng this cho biến cục bộ, các thành phần static
- Lời gọi tới this khi gọi phương thức khởi tạo khác phải là lệnh đầu tiên trong constructor nếu không bạn sẽ bị lỗi

Ví dụ sử dụng this

```
public class Car {  
    public String name;  
    public int year;  
    public float weight;  
    // phương thức khởi tạo không tham số  
    public Car() {  
  
    }  
    // phương thức khởi tạo 1 tham số  
    public Car(String name) {  
        this.name = name;  
    }  
  
    public Car(String name, int year) {  
        this(name); // gọi tới constructor 1 tham số  
        this.year = year; // year ở this.year là thuộc tính  
                          // year ở vế phải dấu = là tham số  
    }  
}
```

Ví dụ sau bị lỗi

```
public class Car {
    public String name;
    public int year;
    public float weight;
    // phương thức khởi tạo không tham số
    public Car() {

    }
    // phương thức khởi tạo 1 tham số
    public Car(String name) {
        this.name = name;
    }

    public Car(String name, int year) {
        this.year = year; // year ở this.year là thuộc tính
                          // year ở về phải dấu = là tham số
        this(name); // error! lời gọi này phải là lời gọi đầu tiên
    }
    // phương thức khởi tạo 3 tham số
    public Car(String name, int year, float weight) {
        this(name, year, weight); // gọi đệ quy chính nó -> error!

        this.weight = weight; // gán giá trị trong tham số weight cho
                              // thuộc tính weight
    }
}
```

Ví dụ minh họa

- Để thực hiện điều này ta sử dụng default hoặc private constructors
- Default constructors sẽ ngăn cản việc tạo đối tượng bên ngoài gói
- Private constructors ngăn cản việc tạo đối tượng bên ngoài lớp chứa nó
- Hữu ích trong việc kiểm soát các đối tượng được tạo ra từ một lớp

Ví dụ

➤ Ví dụ sau tạo private constructor:

```
public class Car {  
    public String name;  
    public int year;  
    public float weight;  
    // phương thức khởi tạo private  
    private Car() {  
    }  
  
    public Car createNewCar() {  
        return new Car(); // ok vì đang thao tác trong cùng Lớp Car  
    }  
}  
  
// Lớp Test.java  
public class Test {  
    public static void main(String[] args) {  
        Car myCar = new Car(); // error!  
    }  
}
```

Nội dung tiếp theo

Tính đóng gói dữ liệu