**Mục đích sử dụng**

* Mục đích của phương thức khởi tạo là để tạo ra đối tượng với những dữ liệu 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**

* Sau đây là cú pháp tổng quát của phương thức khởi tạo:

**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 việc cho phép phương thức khởi tạo được gọi từ bên ngoài lớp ở mức độ nào. Nếu access là public có nghĩa là mọi nơi đều có thể nhìn thấy và gọi phương thức khởi tạo để tạo đối tượng từ lớp. Nếu là private thì chỉ có thể gọi phương thức khởi tạo từ bên trong lớp. Riêng protected sẽ nói sau.
  + Phần name là tên phương thức khởi tạo. Bắt buộc tên của phương thức khởi tạo phải giống chính xác như tên lớp định nghĩa nó. Nếu không sẽ không có phương thức khởi tạo hợp lệ.
  + Tiếp sau đó là cặp ngoặc tròn chứa danh sách tham số bên trong. Đây là phần params. Giống như phương thức thông thường, phương thức khởi tạo cũng có thể có 0, 1 hoặc nhiều tham số. Nhằm cung cấp các giá trị khởi đầu của đối tượng khi ta tạo ra nó.
  + Phần thân phương thức nằm trong cặp {} và chỉ chứa các câu lệnh thực hiện việc khởi tạo hoặc gọi tới 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;

    }

}

**Các đặc trưng của phương thức khởi tạo**

* Phần access của phương thức khởi tạo chỉ có thể chứa 1 keyword. Nếu có sự xuất hiện từ 2 keyword trở lên kể cả việc lặp lại keyword trước đó bạn sẽ bị báo lỗi.
* Phương thức khởi tạo có thể được nạp chồng. Không được phép tồn tại hai phương thức khởi tạo cùng có bộ tham số y hệt nhau.
* Phương thức khởi tạo không có kiểu trả về.
* Không thực hiện gọi đệ quy tới chính nó.
* Phương thức khởi tạo không thể là final, abstract, static, native, strictfp hay synchronized.
* Nếu một lớp không được khai báo phương thức khởi tạo tường minh nào. Java sẽ tự động tạo cho lớp đó một phương thức khởi tạo ngầm định.
* Phương thức khởi tạo mặc định có cùng access modifier, cùng tên với tên lớp, không chứa tham số, không có thực thi nào cả.
* Nếu có một phương thức khởi tạo có tham số được định nghĩa tường minh thì phương thức khởi tạo mặc định sẽ bị ẩn đi. Muốn sử dụng ta phải khai báo tường minh phương thức này ra.
* Các thuộc tính của lớp nếu không được khởi tạo thì nó sẽ nhận giá trị mặc định của kiểu mà nó đang mang. Các kiểu số giá trị mặc định là 0, kiểu boolean là false, kiểu tham chiếu là null.
* 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*

**Sử dụng keyword this**

* Trong ví dụ trên ta đã thấy sự xuất hiện của từ khóa this. Về bản chất this chính là đối tượng của lớp, nó đang thực hiện các hành động được đề cập tại thời điểm hiện tại.
* Và sau đây là các trường hợp sử dụng phổ biến của this:
  + Sử dụng this để chỉ rõ thuộc tính của lớp nếu như tham số của phương thức trùng với tên của thuộc tính. Để đơn giản hóa vấn đề.
  + Sử dụng this để gọi tới phương thức khởi tạo khác trong cùng một lớp. Mục đích tái sử dụng lại code đã có.
  + Sử dụng this để đại diện cho đối tượng hiện thời của lớp.
* Lưu ý this không áp dụng cho các biến cục bộ, các thành phần static.
* Với lời gọi tới phương thức khởi tạo khác qua this thì lời gọi này phải là lệnh đầu tiên trong phương thức khởi tạo.
* 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*

}

*// phương thức khởi tạo 3 tham số*

public Car(String name, int year, float weight) {

        this(name, year); *// gọi phương thức khởi tạo 2 tham số*

this.weight = weight; *// gán giá trị trong tham số weight cho*

*// thuộc tính weight*

}

}

Ví dụ sau là các trường hợp 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 vế 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*

}

}

**Không cho phép 1 lớp tạo đối tượng**

* Để ngăn cấm việc tạo đối tượng từ bên ngoài lớp ta sử dụng default hoặc private constructor.
* Default constructor sẽ ngăn cản việc tạo đối tượng từ bên ngoài một gói(package).
* Còn private constructor sẽ ngăn cản việc tạo đối tượng từ bên ngoài lớp.
* Việc này rất hữu dụng nếu bạn muốn quản lý việc tạo đối tượng của lớp.
* Ví dụ khi kết nối ứng dụng đến cơ sở dữ liệu. Bạn muốn kiểm soát đối tượng quản lý việc truy xuất DB và giới hạn nó về 1 đối tượng duy nhất. Lúc này giải pháp của bạn là sử dụng singleton parttern. Nguyên lý của phương pháp này là tạo ra một đối tượng duy nhất của lớp để dùng chung và constructor của lớp sẽ là private.
* Ví dụ sau minh họa sử dụng 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!*

}

}