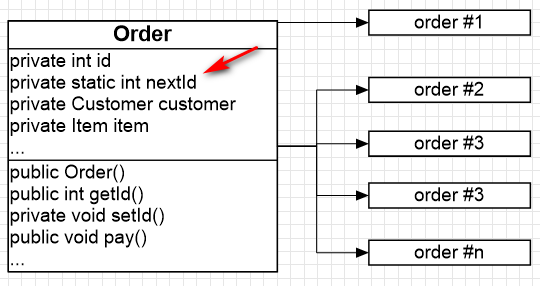
Kết luận đầu tiên khi nói đến thành phần static của lớp là: mọi thành phần static là thành phần thuộc sở hữu lớp. Và chúng vận hành không phụ thuộc vào đối tượng nào của lớp. Tức là có thể dùng chung, chia sẻ chung cho mọi đối tượng của cùng lớp hoặc lớp khác tùy vào access modifier.

**Các trường static**

* Các trường static là các thuộc tính của lớp nhưng được khai báo với keyword static.
* Các trường static là thành phần dùng chung cho toàn bộ các đối tượng của lớp. Nếu access modifier là public thì trường static này có thể dùng chung cho cả các đối tượng của lớp khác.
* Khi ta tạo ra một đối tượng từ lớp với keyword new, đối tượng đó sẽ có một bản copy riêng các thuộc tính không phải static. Nhưng lại dùng chung các thành phần static với lớp. Đó là lý do tại sao khi ta thay đổi giá trị thuộc tính riêng của đối tượng A thì không ảnh hưởng gì đến thuộc tính này của đối tượng B.
* Ví dụ áp dụng: giả sử ta cần xây dựng hệ thống thương mại điện tử. Ở trong đó có lớp đơn hàng. Mã đơn hàng của ta là một số nguyên tự động tăng mỗi khi có đơn hàng mới được đặt. Lúc này ta sẽ thiết kế ra một trường nextId static dùng chung cho các đối tượng đơn hàng của lớp đơn hàng(Order).

[](https://braniumacademy.net/wp-content/uploads/2020/09/vi-du-ve-thanh-phan-static-1.png)Lớp(trái) và đối tượng(phải)

* Cụ thể:

*// lớp mô tả thông tin đơn hàng*

public class Order {

    private int id;

    private static int *nextId* = 100; *// bắt đầu mã đơn hàng từ 100*

private Customer customer;

    private Item item;

    private int quantity;

    private float price;

*//...*

public Order() {

*//...*

setId();

    }

    public int getId() {

        return id;

    }

*// việc thiết lập mã đơn hàng không cần thiết*

*// phải public ra ngoài nên ta để là private*

private void setId() {

        this.id = *nextId*;

*nextId*++;

    }

}

*// lớp mô tả thông tin mặt hàng*

class Item{

    private int id;

    private String name;

*//...*

}

*// lớp mô tả thông tin của khách hàng*

class Customer{

    private String id;

    private String name;

*//...*

}

Mỗi khi khách hàng ấn nút mua hàng thì đơn hàng sẽ được tạo ở đâu đó trong hệ thống. Phần code sau minh họa việc tạo đơn hàng:

public static void main(String[] args) {

    Order order1 = new Order(); *// tạo đơn hàng thứ nhất*

Order order2 = new Order(); *// tạo đơn hàng thứ hai*

*// .. tạo đơn hàng thứ n...*

System.*out*.println("order1 id = " + order1.getId());

    System.*out*.println("order2 id = " + order2.getId());

}

Kết quả hiển thị ra màn hình là:

order1 id = 100

order2 id = 101

**Các hằng số static**

* Các trường static tương đối hiếm được sử dụng nhưng các hằng số static thì lại thường xuyên được sử dụng. Điển hình nhất là các hằng số PI, E của lớp Math. Chúng ta có thể sử dụng chúng bất kì lúc nào nếu muốn và khi sử dụng ta không cần phải có đối tượng, chỉ cần tên lớp Math.Pi hay Math.E là xong.
* Nhắc lại rằng hằng số static trên không vận hành bởi đối tượng của lớp Math. Ta chỉ cần cung cấp tên lớp chứa nó và tên hằng là sử dụng được. Tương tự cho các hằng static bạn tự tạo ra.
* Ví dụ khác là System.out. Đây là một hằng số ta dùng rất nhiều từ đầu khóa học đến giờ. Bản chất nó chính là một hằng số:

public static final PrintStream *out* = null;

* Do hằng số là thành phần sử dụng chung nên ta có thể để public nếu muốn cho các lớp khác cùng sử dụng

**Các phương thức static**

* Phương thức static là phương thức cũng không vận hành dựa trên đối tượng.
* Nếu sử dụng trong nội bộ lớp, ta chỉ cần tên lớp và danh sách đối số tương ứng.
* Nếu sử dụng bên ngoài lớp, ta cung cấp thêm tên lớp ở đầu làm tham chiếu.
* Các thành phần static nói chung không sử dụng được với từ khóa this vì static không thuộc sở hữu của đối tượng nào cả.
* Phương thức static chỉ dùng để thao tác với các thành phần static. Không dùng để thao tác với các thuộc tính hay các phương thức non-static.
* Ví dụ:

public class StaticExample {

    public static void main(String[] args) {

*doSomeStatic*(); *// ok*

doSomething(); *// error*

}

*// phương thức vận hành không cần object*

public static void doSomeStatic() {

        System.*out*.println("Bên trong doSomeStatic()");

    }

*// phương thức vận hành trên object*

public void doSomething(){

        System.*out*.println("Bên trong doSomething()");

    }

}

* Ta sử dụng phương thức static trong hai trường hợp:
  + Khi phương thức không cần truy cập đến thuộc tính của đối tượng vì tất cả những gì cần thiết đã được cung cấp qua các đối số của lời gọi phương thức
  + Khi 1 phương thức chỉ cần truy cập đến các trường static của lớp

**Phương thức main**

* Để ý rằng phương thức main là một static method. Ta dùng phương thức main để khởi chạy chương trình.
* Phương thức main không vận hành dựa vào đối tượng. Thực tế rằng khi chương trình bắt đầu chạy, không có bất kì đối tượng nào cả. Lúc này chương trình vào main, khởi chạy và các đối tượng cần thiết lần lượt được tạo ra.
* Mỗi một lớp đều có thể có cho riêng mình một phương thức main. Điều này rất hữu ích cho việc thực hiện unit testing.
* Ví dụ hình ảnh của phương thức main:

public static void main(String[] args) {

    Order order1 = new Order(); *// tạo đơn hàng thứ nhất*

Order order2 = new Order(); *// tạo đơn hàng thứ hai*

*// .. tạo đơn hàng thứ n...*

System.*out*.println("order1 id = " + order1.getId());

    System.*out*.println("order2 id = " + order2.getId());

}