

Bài 49: Lớp Object

- ✓ Giới thiệu lớp Object
- ✓ Các phương thức thường dùng
- ✓ Ghi đè toString()
- ✓ Ghi đè equals()





- >Lớp Object là tổ tiên của mọi lớp trong Java
- Nếu một lớp không chỉ rõ lớp cha trực tiếp thì lớp cha trực tiếp của nó chính là Object
- Lớp Object hữu ích trong trường hợp bạn muốn tham chiếu đến bất kì đối tượng nào có kiểu chưa biết trước

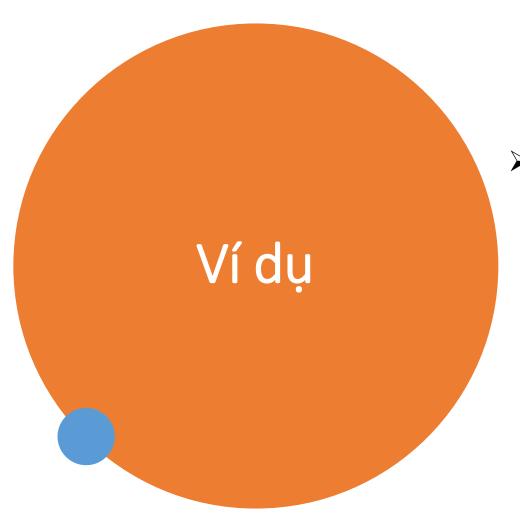




➤ Hai lớp sau là tương đương:

```
public class Car {
    private String brand;
    private String name;
    private float weight;
    public void speedUp() {
       //...
public class Car extends Object {
    private String brand;
    private String name;
    private float weight;
    public void speedUp() {
       //...
```





```
➤Sử dụng lớp Object:
```

```
public class Test {
    public static void main(String[] args) {
        Car car = new Car(); // ok
        Object aCar = new Car(); // ok
    }
}
```





Sau đây là một số phương thức thường dùng và mô tả của nó:

Phương thức	Mô tả
Class getClass()	Trả về kiểu(lớp) của đối tượng hiện thời.
boolean equals(Object obj)	Trả về giá trị kiểu boolean kết luận liệu hai đối tượng đang so sánh có
	tương đương hay không.
int hashCode()	Trả về mã băm định danh của đối tượng hiện thời. Mỗi đối tượng
	trong Java khi được tạo ra sẽ có một mã băm duy nhất giống như số
	chứng minh thư hay căn cước công dân.
String toString()	Trả về một String mô tả vắn tắt thông tin về đối tượng hiện thời.





- ➤ Phương thức toString() dùng để:
 - > Mô tả thông tin về tên lớp
 - > Mô tả các trường và dữ liệu tương ứng
- ➤ Khi ta dùng đối tượng của một lớp làm đối số của phương thức như printf, print, println thì phương thức toString() của đối tượng tự động được gọi
- ➤ Ví dụ minh họa:





Dể lấy đầy đủ thông tin về tên gói và tên lớp, ta có thể sử dụng phương thức getClass().getName() như sau:





➤ Khi chạy chương trình:

```
public class Test {
   public static void main(String[] args) {
       Car car = new Car("Vinfast", "President", 2250);
       System.out.println(car);
       System.out.println("========");
       // gán giá trị ra một String:
       String carInfo = car.toString();
       System.out.println(carInfo);
lesson49.Car[brand='Vinfast', name='President', weight=2250.0]
```

```
lesson49.Car[brand='Vinfast', name='President', weight=2250.0]
```





➤ Nếu lớp cha đã override toString() thì lớp con chỉ cần bổ sung thêm thông tin:

```
@Override
public String toString() {
    return getClass().getName() + "[" +
            "brand='" + brand + '\'' +
            ", name='" + name + '\'' +
            ", weight=" + weight +
@Override
public String toString() {
    return super.toString() + "[" +
            "voltage=" + voltage +
            ", bateryCapacity=" + bateryCapacity +
            ']';
```



➤ Khi chạy chương trình:





- Phương thức equals dùng để kiểm tra xem hai đối tượng có tương đương hay không
- Theo mặc định, hai đối tượng tương đương khi chúng là một
- Hai đối tượng của cùng lớp là hai đối tượng độc lập nhau dù chúng có chung các thông tin
- ➤ Vậy nếu ta giả sử rằng hai đối tượng cùng thông tin của cùng lớp là tương đương, ta sẽ phải override lại phương thức equals
- ➤ Ví dụ: hai sinh viên cùng tên, cũng mã sinh viên và ngày tháng năm sinh có thể coi là tương đương nhau





- Bước 1: đặt tên tham số của phương thức là otherObject để phân biệt với đối tượng hiện thời
- ▶ Bước 2: kiểm tra xem nếu đối tượng hiện thời và tham số là một, return true
- ➤ Bước 3: nếu otherObject là null, return false
- ➤ Bước 4: nếu otherObject không cùng kiểu với đối tượng hiện thời, return false
- ➢ Bước 5: ép kiểu của otherObject sang kiểu của lớp hiện thời
- ➤ Bước 6: So sánh các trường dữ liệu và trả về kết quả. Nếu dữ liệu kiểu nguyên thủy, dùng ==. Nếu dữ liệu kiểu tham chiếu, dùng Objects.equals()





➤ Ví dụ override lại equals:



➤ Tại nơi sử dụng:





➤ Thực hiện trong công cụ lập trình



Các chuẩn thiết kế kế thừa

