Bài 47: Ghi đè phương thức

- ✓ Mục đích sử dụng
- ✓ Cú pháp
- ✓ Các thành phần final
- ✓ Quy tắc ghi đè
- ✓ super keyword
- ✓ Bài tập thực hành

Mục đích sử dụng

- Override hay ghi đè là kĩ thuật trong Java cho phép định nghĩa lại phương thức của lớp cha theo cách của lớp con
- Mục đích chính của override là lớp cha tạo ra phương thức chung cho tất cả các lớp con sau đó lớp con được phép override lại cho phù hợp với đặc trưng và ngữ cảnh sử dụng
- Ví dụ: tất cả động vật đều di chuyển nhưng chim di chuyển bằng cách bay, rùa thì bò trên mặt đất bằng tứ chi còn cá thì bơi dưới nước bằng vây
- Override trong hướng đối tượng chỉ xảy ra ở lớp con trong mối quan hệ kế thừa. Các phương thức final, private, static không thể override

Cú pháp

- Để override một phương thức của lớp cha, ta gõ lại y nguyên dấu hiệu nhận biết: kiểu trả về, tên phương thức, kiểu, số lượng và thứ tự tham số
- Phần access modifier của phương thức override chỉ có thể có mức khả dụng >= access modifier trong phương thức nguyên bản của lớp cha
- Nội dung của phương thức được override thường thay đổi cho phù hợp với ngữ cảnh của lớp con nhưng vẫn giữ nguyên tinh thần của phương thức gốc

Cú pháp

- Có thể tái sử dụng phương thức gốc của lớp cha bằng lời gọi super.tên_phương_thức
- Sử dụng @Override trước tên phương thức override trong lớp con để đánh dấu rằng đó là phương thức override hay tạo mới bởi lớp con

```
@Override
public void work() { // ok
    super.work();
    System.out.println("Sinh viên đang học bài...");
}
```

Ví dụ

➢Ví dụ về phương thức override:

```
public class Person {
    private String fullName;
    private Date dateOfBirth;
    //...
    // phương thức gốc trong lớp cha
    public void work() {
        System.out.println("Người đang làm việc...");
    }
}

class Student extends Person {
    private String studentId;
    //...

    @Override
    public void work() { // ghi đè phương thức gốc trong lớp cha
        super.work(); // tái sử dụng phương thức nguyên bản trong lớp cha
        System.out.println("Sinh viên đang học bài...");
    }
}
```

Ví dụ

- Khi chạy, chương trình sẽ tự xác định chính xác phương thức cần gọi dựa vào đối tượng đang tham chiếu:

```
public class Test {
    public static void main(String[] args) {
        Person person = new Person();
        Student student = new Student();
        Person nam = new Student();
        person.work(); // gọi phương thức từ đối tượng của person
        System.out.println("====================================");
        student.work(); // gọi phương thức từ đối tượng của Student
        System.out.println("====================================");
        nam.work(); // gọi phương thức từ đối tượng person tham chiếu đến
Student
    }
}
```

```
Người đang làm việc...
====================================
Người đang làm việc...
Sinh viên đang học bài...
====================================
Người đang làm việc...
Sinh viên đang học bài...
```

Các lớp final

- Lớp final là các lớp có keyword final trong khai báo
- Lớp final không thể bị kế thừa
- Mục đích sử dụng final class:
  - Ngăn cấm lớp khác kế thừa lớp hiện thời
  - Để tạo các lớp immutable(bất biến), tức lớp mà khi đối tượng của nó được tạo thì giá trị của đối tượng không thể thay đổi
- Ví dụ lớp bất biến: lớp String
- Ví dụ lớp nhằm cấm kế thừa: các lớp bao của các kiểu nguyên thủy như Float, Long…

Các phương thức final

- Phương thức final là phương thức có non-access modifier là final
- Phương thức final không thể override
- Ví dụ phương thức final:

```
public class Person {
    private String fullName;
    private Date dateOfBirth;

    public final void setFullName(String fullName) {
        this.fullName = fullName;
    }

    public void work() {
        System.out.println("Người đang làm việc...");
    }
}
```

Ví dụ

```java
public class Person {
    private String fullName;
    private Date dateOfBirth;

    public final void setFullName(String fullName) {
        this.fullName = fullName;
    }

    public void work() {
        System.out.println("Người đang làm việc...");
    }
}

class Student extends Person {
    private String studentId;
    //...

    @Override
    public void work() { // ok
        super.work();
        System.out.println("Sinh viên đang học bài...");
    }

    @Override
    public void setFullName(String fullName) { // error
        super.setFullName(fullName);
    }
```

Quy tắc override

- Danh sách tham số của phương thức override phải giống chính xác như trong phương thức gốc
- Kiểu trả về, tên và chức năng của phương thức override phải giống với phương thức nguyên bản
- Cấp độ khả dụng của access modifier của phương thức override phải rộng hơn hoặc bằng của phương thức nguyên bản
- Phương thức được override phải nằm trong lớp cha của lớp hiện thời
- Phương thức final của lớp cha không thể override bởi lớp con

Quy tắc override

- Phương thức static của lớp cha không thể override bởi lớp con nhưng có thể được khai báo lại bởi lớp con
- Lớp con cùng gói với lớp cha có thể override các phương thức không private, static, final của lớp cha
- Lớp con không cùng gói với lớp cha chỉ có thể override lại các phương thức public, protetcted và không final, static của lớp cha
- Phương thức khởi tạo của lớp cha không thể override bởi lớp con

Keyword super

- Sử dụng super keyword để:
  - Gọi đến constructor của lớp cha
  - Gọi đến các trường được kế thừa từ lớp cha
  - Gọi đến các phương thức khác
  - Gọi đến phương thức nguyên bản của phương thức hiện thời trong lớp cha

Minh họa cụ thể

- Thực hiện trong công cụ lập trình

Nội dung tiếp theo

So sánh overloading và overriding