**Polymorphism**

Tính đa hình là khả năng một đối tượng có thể thực hiện một tác vụ theo nhiều cách khác nhau.

Đối với tính chất này, nó được thể hiện rõ nhất qua việc gọi phương thức của đối tượng. Các phương thức hoàn toàn có thể giống nhau, nhưng việc xử lý luồng có thể khác nhau. Nói cách khác: Tính đa hình cung cấp khả năng cho phép người lập trình gọi trước một phương thức của đối tượng, tuy chưa xác định đối tượng có phương thức muốn gọi hay không. Đến khi thực hiện (run-time), chương trình mới xác định được đối tượng và gọi phương thức tương ứng của đối tượng đó. Kết nối trễ giúp chương trình được uyển chuyển hơn, chỉ yêu cầu đối tượng cung cấp đúng phương thức cần thiết là đủ.

Trong Java, chúng ta sử dụng nạp chồng phương thức (method overloading) và ghi đè phương thức (method overriding) để có tính đa hình.

Nạp chồng (Overloading): Đây là khả năng cho phép một lớp có nhiều thuộc tính, phương thức cùng tên nhưng với các tham số khác nhau về loại cũng như về số lượng. Khi được gọi, dựa vào tham số truyền vào, phương thức tương ứng sẽ được thực hiện.

Ghi đè (Overriding): là hai phương thức cùng tên, cùng tham số, cùng kiểu trả về nhưng thằng con viết lại và dùng theo cách của nó, và xuất hiện ở lớp cha và tiếp tục xuất hiện ở lớp con. Khi dùng override, lúc thực thi, nếu lớp Con không có phương thức riêng, phương thức của lớp Cha sẽ được gọi, ngược lại nếu có, phương thức của lớp Con được gọi.

**Đa hình tại runtime trong Java**

Ví dụ 1

**public** **class** Bike {

**public** **void** run() {

        System.out.println("running");

    }

}

**public** **class** Splender **extends** Bike {

**public** **void** run() {

        System.out.println("running safely with 60km");

    }

**public** **static** **void** main(String args[]) {

        Bike b = **new** Splender();

        b.run();

    }

}

Ví dụ 1: chúng ta tạo hai lớp Bike và Splendar. Lớp Splendar kế thừa lớp Bike và ghi đè phương thức run() của nó. Chúng ta gọi phương thức run bởi biến tham chiếu của lớp cha. Khi nó tham chiếu tới đối tượng của lớp con và phương thức lớp con ghi đè phương thức của lớp cha, phương thức lớp con được triệu hồi tại runtime.

Kết quả: running safely with 60km

Ví dụ 2

**class** Bank {

**int** getRateOfInterest() {

**return** 0;

    }

}

**class** VCB **extends** Bank {

**int** getRateOfInterest() {

**return** 8;

    }

}

**class** AGR **extends** Bank {

**int** getRateOfInterest() {

**return** 7;

    }

}

**class** CTG **extends** Bank {

**int** getRateOfInterest() {

**return** 9;

    }

}

**class** Test3 {

**public** **static** **void** main(String args[]) {

        Bank b1 = **new** VCB(); // upcasting

        Bank b2 = **new** AGR(); // upcasting

        Bank b3 = **new** CTG(); // upcasting

        System.out.println("VCB lai suat la: " + b1.getRateOfInterest());

        System.out.println("AGR lai suat la: " + b2.getRateOfInterest());

        System.out.println("CTG lai suat la: " + b3.getRateOfInterest());

    }

}

Giả sử Bank là một lớp cung cấp phương thức để lấy lãi suất. Nhưng lãi suất lại khác nhau giữa từng ngân hàng. Ví dụ, các ngân hàng VCB, AGR và CTG có thể cung cấp các lãi suất lần lượt là 8%, 7% và 9%.

Kết quả:

VCB lai suat la: 8

AGR lai suat la: 7

CTG lai suat la: 9

**Nạp chồng phương thức (method overloading)**

Nếu một lớp có nhiều phương thức cùng tên nhưng khác nhau về kiểu dữ liệu hoặc số lượng các tham số, thì đó là nạp chồng phương thức (Method Overloading).

Sử dụng nạp chồng phương thức giúp tăng khả năng đọc hiểu chương trình.

Nạp chồng phương thức được sử dụng để thu được tính đa hình lúc biên dịch (compile).

Có 2 cách nạp chồng phương thức trong java

* Thay đổi số lượng các tham số
* Thay đổi kiểu dữ liệu của các tham số

**Nạp chồng phương thức: thay đổi số lượng các tham số**

Ví dụ: tạo 2 phương thức có cùng kiểu dữ liệu: phương thức add() đầu tiên thực hiện việc tính tổng của 2 số, phương thức thứ hai thực hiện việc tính tổng của 3 số.

**class** Adder {

**static** **int** add(**int** a, **int** b) {

**return** a + b;

    }

**static** **int** add(**int** a, **int** b, **int** c) {

**return** a + b + c;

    }

}

**class** TestOverloading1 {

**public** **static** **void** main(String[] args) {

        System.out.println(Adder.add(5, 5));

        System.out.println(Adder.add(5, 5, 5));

    }

}

**Nạp chồng phương thức: thay đổi kiểu dữ liệu của các tham số**

Ví dụ: tạo 2 phương thức có kiểu dữ liệu khác nhau: phương thức add() đầu tiên nhận 2 đối số có kiểu giá trị là integer, phương thức thứ hai nhận 2 đối số có kiểu giá trị là double.

**class** Adder {  
**static int** add(int a, int b) {  
**return** a + b;  
}

**static double** add(double a, double b) {  
**return** a + b;  
}  
}

**class** TestOverloading2 {  
**public static void** main(String[] args) {  
System.out.println(Adder.add(5, 5));  
System.out.println(Adder.add(4.3, 5.6));  
}  
}

**Ghi đè phương thức (method overriding)**

Ghi đè phương thức trong java xảy ra nếu lớp con có phương thức giống lớp cha.

Nói cách khác, nếu lớp con cung cấp sự cài đặt cụ thể cho phương thức đã được cung cấp bởi một lớp cha của nó được gọi là ghi đè phương thức (method overriding) trong java.

Ghi đè phương thức được sử dụng để thu được tính đa hình tại runtime.

Nguyên tắc ghi đè phương thức:

* Phương thức phải có tên giống với lớp cha.
* Phương thức phải có tham số giống với lớp cha.
* Lớp con và lớp cha có mối quan hệ kế thừa.

Ví dụ về ghi đè phương thức (method overriding)

Ví dụ 1: chúng ta định nghĩa phương thức run() trong lớp con giống như đã được định nghĩa trong lớp cha, nhưng được cài đặt rõ ràng trong lớp con. Tên và tham số của phương thức là giống nhau, 2 lớp cha và con có quan hệ kế thừa.

**class** Vehicle {

**void** run() {

        System.out.println("Vehicle is running");

    }

}

**class** Bike **extends** Vehicle {

**void** run() {

        System.out.println("Bike is running safely");

    }

**public** **static** **void** main(String args[]) {

        Bike obj = **new** Bike();

        obj.run();

    }

}