**Tính đa hình (Polymorphism) trong Java**

**1.Khái niệm tính đa hình**

Kỹ thuật đa hình trong các ngôn ngữ lập trình hướng đối tượng tạo điều kiện cho các lập trình viên gia tăng khả năng tái sử dụng những đoạn mã nguồn được viết một cách tổng quát và có thể thay đổi cách ứng xử một cách linh hoạt tùy theo loại đối tượng.

Tính đa hình (Polymorphism) trong Java được hiểu là trong từng trường hợp, hoàn cảnh khác nhau thì đối tượng có hình thái khác nhau tùy thuộc vào từng ngữ cảnh. Đối tượng có tính đa hình được xem như một đối tượng đặc biệt vì có lúc đối tượng này trở thành một đối tượng khác và cũng có lúc đối tượng này trở thành một đối tượng khác nữa (tùy vào từng hoàn cảnh). Sự "nhập vai" vào các đối tượng khác nhau này giúp cho đối tượng đa hình ban đầu có thể thực hiện những hành động khác nhau của từng đối tượng cụ thể.

**Ví dụ:** Khi bạn ở trong trường học là sinh viên thì bạn có nhiệm vụ học, nghe giảng,..., nhưng khi bạn ở nhà thì bạn lại đóng vai trò là thành viên trong gia đình và bạn có nhiệm vụ phải làm việc nhà, rồi khi bạn vào siêu thị thì bạn đóng vai trò là khách hàng đi mua hàng. Vì vậy, chúng ta có thể hiểu đa hình của đối tượng là trong từng trường hợp, hoàn cảnh khác nhau thì đối tượng có khả năng thực hiện các công việc khác nhau.

Để thể hiện tính đa hình, chúng ta cần đảm bảo 2 điều kiện sau:

* Các lớp phải có quan hệ kế thừa với 1 lớp cha nào đó.
* Phương thức đa hình phải được ghi đè (override) ở lớp con. Tính đa hình chỉ được thể hiện ghi đã ghi đè lên phương thức của lớp cha.

**2. Ví dụ về tính đa hình**

Ví dụ dưới đây sẽ minh họa cách sử dụng tính đa hình trong Java.

**Shape.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | package vidu;    public class Shape {      public void show() {          System.out.println("Đây là phương thức show() của lớp Shape");      }  } |

**Rectangle.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | package vidu;    public class Rectangle extends Shape {      public void show() {          System.out.println("Đây là phương thức show() của lớp Rectangle");      }  } |

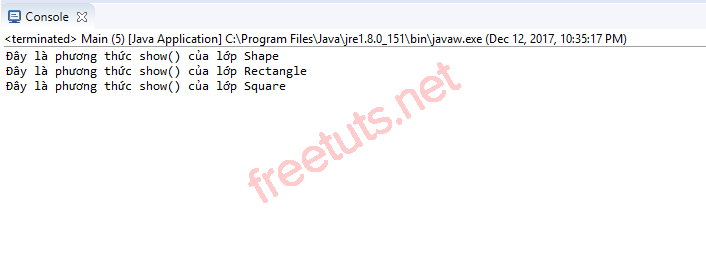
**Square.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | package vidu;    public class Square extends Shape {      public void show() {          System.out.println("Đây là phương thức show() của lớp Square");      }  } |

**Main.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | package vidu;    public class Main {        public static void main(String[] args) {          Shape shape = new Shape();          shape.show();   // hiển thị dòng "Đây là phương thức show() của lớp Shape"            // bản chất của shape là Shape, nhưng vì khai báo Rectangle nên chúng ta chỉ nhìn thấy những gì mà Rectangle có          // vì vậy sẽ chạy những hàm của Rectangle          shape = new Rectangle();          shape.show();   // hiển thị dòng "Đây là phương thức show() của lớp Rectangle"            // tương tự lúc này shape sẽ đóng vai trò là 1 Square          shape = new Square();          shape.show();   // hiển thị dòng "Đây là phương thức show() của lớp Square"      }    } |

**Kết quả sau khi biên dịch chương trình:**



**Giải thích hoạt động của chương trình trên:**

Trong chương trình trên, tôi có tạo ra 3 lớp là Shape, Rectangle và Square với Shape là lớp cha và 2 lớp còn lại là lớp con. Cả 3 lớp này đều có chung một phương thức show() nhưng có nội dung phương thức khác nhau. Trong lớp Main, tôi tiến hành gọi 3 phương thức show() của 3 lớp này. Nếu làm như các bài trước thì để gọi phương thức show() ứng với từng lớp thì chúng ta phải tạo một đối tượng của lớp tương ứng, nhưng đối với tính đa hình thì chúng ta không cần phải tạo ra 3 đối tượng của 3 lớp mà chúng ta sẽ chỉ cần khai báo đối tượng của lớp Shape có khả năng đóng vai trò là lớp con thông qua 2 dòng code shape = new Rectangle(); và shape = new Square();, lúc này đối tượng shape sẽ đóng vai trò là lớp con tương ứng. Đây chính là ý nghĩa của tính đa hình.