**Tính kế thừa (Inheritance) trong Java**

**1. Khái niệm tính kế thừa**

### **1.1. Khái niệm tổng quát hóa và đặc biệt hóa**

Quan hệ tổng quát hóa và đặc biệt hóa là 1 quan hệ thực sự rất tự nhiên và thông dụng trong thế giới thực. Đối tượng đặc biệt hóa của một dạng đối tượng tổng quát sẽ có đầy đủ tính chất của loại đối tượng ban đầu và có thêm các đặc điểm riêng của nó. Ví dụ hình chữ nhật lại là khái niệm tổng quát hóa của hình vuông, và ngược lại hình vuông là một trường hợp đặc biệt của hình chữ nhật. Hình vuông có đầy đủ các tính chất của hình chữ nhật bởi vì hình chữ nhật là tổng quát của hình vuông; hơn nữa, hình vuông có những tính chất đặc biệt mà hình chữ nhật không có bởi vì hình vuông được đặc biệt hóa từ hình chữ nhật.

Rất nhiều lớp đối tượng trong thế giới thực là đặc biệt hóa hay tổng quát hóa của lớp đối tượng khác. Quan hệ tổng quát hóa và đặc biệt hóa đóng vai trò rất quan trọng, hầu hết các ngôn ngữ lập trình hướng đối tượng đều hỗ trợ người lập trình cài đặt quan hệ này.

### **1.2. Khái niệm tính kế thừa**

Khái niệm "kế thừa" thường được dùng trong các ngôn ngữ lập trình hướng đối tượng để chuyển quan hệ tổng quát hóa và đặc biệt hóa của thế giới thực vào các chương trình [máy tính](https://freetuts.net/tinh-ke-thua-inheritance-trong-java-1127.html). Tính kế thừa cho phép ta xây dựng một lớp mới dựa trên các định nghĩa của một lớp đã có.

*Bài viết này được đăng tại [free tuts .net]*

Nếu một lớp A là lớp đặc biệt hóa của lớp B (nghĩa là lớp B là lớp tổng quát hóa của lớp A) thì trong lập trình hướng đối tượng lớp A được cài đặt là lớp kế thừa của lớp B (nói ngắn gọn là "lớp A kế thừa lớp B"). Khi lớp A kế thừa lớp B thì lớp B được gọi là "lớp cơ sở" hay "lớp cha" (trong mối quan hệ kế thừa đang xét), còn lớp A thì được gọi là "lớp kế thừa", "lớp con" hay "lớp dẫn xuất".

### **1.3. Lợi ích của tính kế thừa**

Sau đây là một số lợi ích có được khi vận dụng tính kế thừa trong lập trình:

* Lớp con (lớp A) có thể tận dụng lại các thuộc tính và phương thức của lớp cha (lớp B) (nghĩa là các thuộc tính và phương thức của lớp B có thể được tái sử dụng bởi lớp A).
* Lớp A có thể định nghĩa thêm thuộc tính và phương thức mới của riêng nó và có thể định nghĩa lại (hay còn gọi là ghi đè phương thức, overriding) phương thức được kế thừa từ lớp B cho phù hợp với mục đích của nó.

### **1.4. Các dạng kế thừa**

Trong Java, chúng ta có 3 dạng kế thừa chính đó là: **kế thừa từ Class, kế thừa từ lớp trừu tượng (**Abstract class**) và kế thừa từ Interface**. Trong phạm vi của bài này, chúng ta sẽ chỉ tìm hiểu về dạng kế thừa từ Class. Sang các bài sau, tôi sẽ trình bày về 2 dạng kế thừa còn lại.

**2. Xây dựng lớp con và lớp cha trong tính kế thừa**

Khi lập trình với tính kế thừa trong Java, việc đầu tiên và quan trọng nhất đó là chúng ta phải xây dựng được lớp con và lớp cha dựa vào những thông tin đã có. Vậy để làm được việc này, chúng ta sẽ làm như sau:

**Xây dựng lớp cha:** những thông tin nào chung giữa các đối tượng (bao gồm thuộc tính và phương thức) thì chúng ta tập hợp lại tạo thành lớp cha.

**Xây dựng lớp con:** những thông tin nào chỉ có trong từng đối tượng cụ thể thì chúng ta tập hợp lại tạo thành lớp con.

**2.1. Ví dụ về tính kế thừa**

Để cài đặt tính kế thừa, chúng ta sẽ sử dụng từ khóa extends. Giả sử chúng ta có 1 lóp A kế thừa từ lớp B thì chúng ta sẽ khai báo lớp B theo cú pháp như sau:

**Cú pháp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | public class A {      ...  }      public class B extends A {      ...  } |

Để hiểu hơn về tính kế thừa, các bạn theo dõi ví dụ sau:

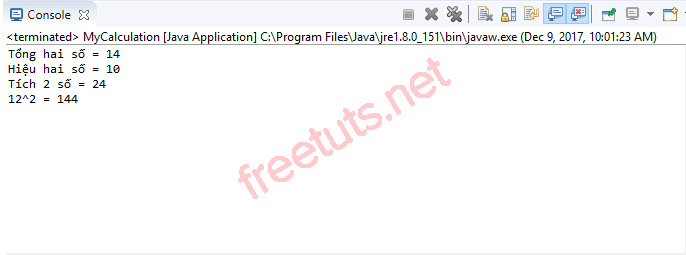
**Calculate.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | package vidu;    public class Calculation {      protected int c;      public void phepCong(int a, int b) {          c = a + b;          System.out.println("Tổng hai số = " + c);      }        public void phepTru(int a, int b) {          c = a - b;          System.out.println("Hiệu hai số = " + c);      }  } |

**MyCalculation.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | package vidu;    public class MyCalculation extends Calculation {      public void phepNhan(int a, int b) {          c = a \* b;          System.out.println("Tích 2 số = " + c);      }        public void phepLuyThua(int a, int b) {          c = (int) Math.pow(a, b);          System.out.println(a + "^" + b + " = " + c);      }        public static void main(String[] args) {          int a = 12, b = 2;          MyCalculation myCalculation = new MyCalculation();          myCalculation.phepCong(a, b);          myCalculation.phepTru(a, b);          myCalculation.phepNhan(a, b);          myCalculation.phepLuyThua(a, b);      }  } |

Kết quả sau khi biên dịch chương trình:



Trong ví dụ này, tôi tạo ra 2 lớp Calculation và MyCalculation kế thừa từ lớp Calculation. Trong lớp Calculation tôi có khai báo 1 thuộc tính c có kiểu int và phạm vi truy cập là protected nhằm mục đích cho phép các lớp con của lớp này có thể dùng lại thuộc tính c (các bạn có thể dùng từ khóa public hoặc không dùng từ khóa nhưng không được khai báo là private) cùng với 2 phương thức là phepCong(int a, int b) để thực hiện phép cộng và phepTru(int a, int b) để thực hiện phép trừ.

Lớp MyCalculation là lớp kế thừa từ lớp Calculation. Lớp này dùng lại thuộc tính c và 2 phương thức phepCong(int a, int b) và phepTru(int a, int b) của lớp Calculation, vì vậy trong lớp MyCalculation không khai báo lại c, phepCong(int a, int b) và phepTru(int a, int b) vì tất cả chúng đều được lấy lại của lớp Calculation. Ngoài ra, lớp MyCalculation còn khai báo 2 phương thức riêng của nó đó là phepNhan(int a, inb b) và phepLuyThua(int a, int b) để thực hiện phép nhân và phép lũy thừa hai số.

Hàm main() của lớp MyCalculation minh họa việc khai báo và sử dụng đối tượng myCalculation (kiểu MyCalculation). Trong hàm này, tôi có 2 dòng code myCalculation.phepCong(a, b); và myCalculation.phepTru(a, b); dùng để gọi phương thức phepCong(int a, int b) và phepTru(int a, int b), mã nguồn của 2 phương thức này được kế thừa từ lớp Calculation mà không cần viết lại cho lớp MyCalculation.

**Lưu ý:**

* Qua ví dụ trên, chúng ta có thể rút ra kết luận là một lớp con có thể kế thừa tất cả các thành phần của lớp cha (thuộc tính, phương thức,...) nhưng không thể kế thừa được lớp con. Tuy nhiên, hàm tạo của lớp cha có thể được gọi từ lớp con thông qua từ khóa super.
* Một lớp con chỉ có thể kế thừa trực tiếp từ một lớp cha. Nếu lớp con không kế thừa từ một lớp cha nào thì mặc định nó sẽ kế thừa từ một lớp cha tên là Object.

**3. Từ khóa super**

Từ khóa super được sử dụng trong các trường hợp sau:

* Nó được sử dụng để phân biệt các thành phần có cùng tên giữa lớp cha và lớp con.
* Nó được sử dụng để gọi hàm tạo của lớp cha từ lớp con.

### **3.1. Phân biệt các thành phần có cùng tên giữa lớp cha và lớp con.**

Nếu chúng ta có một lớp con thừa kế từ một lớp cha và các thành phần của lớp cha có cùng tên với các thành phần của lớp con thì để phân biệt các thành phần của lớp cha từ lớp con, chúng ta sẽ sử dụng từ khóa super với cú pháp như sau:

**Cú pháp**

|  |  |
| --- | --- |
| 1  2 | super.tên\_thuộc\_tính;  super.tên\_phương\_thức(); |

Dưới đây là ví dụ minh họa cách sử dụng từ khóa super trong trường hợp này.

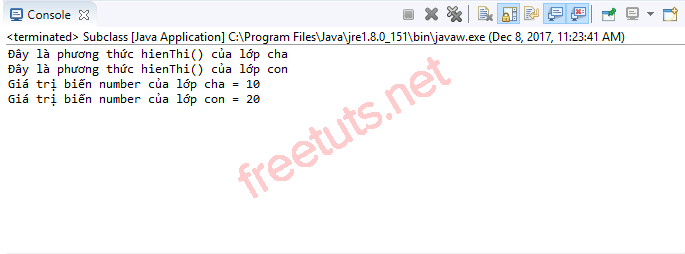
**Superclass.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | package vidu;    public class Superclass {      int number = 10;        public void hienThi() {          System.out.println("Đây là phương thức hienThi() của lớp cha");      }  } |

**Subclass.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | package vidu;    public class Subclass extends Superclass {      int number = 20;        public void hienThi() {          System.out.println("Đây là phương thức hienThi() của lớp con");      }        public void subclassMethod() {          Subclass subclass = new Subclass();            // gọi phương thức hienThi() của lớp cha          // sử dụng từ khóa super()          super.hienThi();            // gọi phương thức hienThi() của lớp con          subclass.hienThi();            // hiển thị giá trị biến number của lớp cha          System.out.println("Giá trị biến number của lớp cha = " + super.number);            // hiển thị giá trị biến number của lớp con          System.out.println("Giá trị biến number của lớp con = " + subclass.number);      }          public static void main(String[] args) {          Subclass objSubclass = new Subclass();          objSubclass.subclassMethod();      }    } |

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



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

Trong ví dụ này chúng ta có 2 lớp: lớp cha tên là Superclass và lớp con tên là Subclass. Hai lớp này cùng có một phương thức tên là hienThi() với nội dung phương thức khác nhau và cùng có một biến number với giá trị biến khác nhau. Trong lớp Subclass chúng ta sẽ tiến hành gọi đến phương thức hienThi() và thuộc tính number của 2 lớp này. Đối với thuộc tính và phương thức của lớp Superclass, chúng ta sẽ gọi thông qua từ khóa super với 2 dòng code là super.hienThi() và super.number, còn đối với thuộc tính và phương thức của lớp Subclass thì chúng ta gọi bình thường thông qua tên đối tượng của nó.

### **3.2. Gọi hàm tạo của lớp cha**

Nếu chúng ta có 1 lớp là lớp con của một lớp khác thì lớp con đó có thể tự động gọi được hàm tạo mặc định của lớp cha. Nhưng trong trường hợp chúng ta muốn gọi hàm tạo có đối số của lớp cha, chúng ta cần sử dụng từ khóa super với cú pháp như sau:

**Cú pháp**

|  |  |
| --- | --- |
| 1 | super(tên\_đối\_số); |

**Lưu ý:** từ khóa super trong trường hợp này chỉ có thể được dùng ngay trong hàm tạo và phải được khai báo đầu tiên bên trong hàm tạo đó, nếu chúng ta để nó vào trong các phương thức bình thường khác thì hệ thống sẽ báo lỗi.

Dưới đây là ví dụ minh họa cách sử dụng từ khóa super trong trường hợp này.

**Superclass.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | package vidu;    public class Superclass {        public Superclass(int number) {          System.out.println("Đây là hàm tạo có đối số của lớp Superclass, giá trị"                  + " biến number = " + number);      }  } |

**Subclass.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | package vidu;    public class Subclass extends Superclass {        public Subclass(int number) {          super(number);  // gọi hàm tạo của lớp Superclass          System.out.println("Đây là hàm tạo của lớp Subclass, giá trị biến number = " + number);      }        public static void main(String[] args) {          Subclass subclass = new Subclass(10);      }    } |

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

