**Overriding và Overloading trong Java**

**1. Overriding**

Trong các bài trước, các bạn đã được tìm hiểu sơ qua về overriding thông qua các ví dụ. Trong Java, overriding được định nghĩa là ghi đè phương thức, có nghĩa là khi một lớp con kế thừa trực tiếp từ một lớp cha thì lớp con đó có thể định nghĩa lại một phương thức đã có trong lớp cha cho phù hợp với mục đích của nó.

### **Quy tắc ghi đè phương thức**

* Danh sách các đối số phải giống với phương thức được ghi đè.
* Kiểu trả về phải giống với kiểu trả về được khai báo trong phương thức được ghi đè của lớp cha.
* Mức truy cập của phương thức ghi đè phương thức ở lớp cha không được nhỏ hơn phương thức đó trong lớp cha. Ví dụ: phương thức trong lớp cha được khai báo là public thì phương thức ghi đè phương thức đó trong lớp con không thể có phạm vi truy cập là private hoặc protected.
* Một phương thức được khai báo là final hoặc static thì phương thức không thể được ghi đè.
* Các hàm tạo của lớp cha không thể được ghi đè.

### **Ví dụ**

**Superclass.java**

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

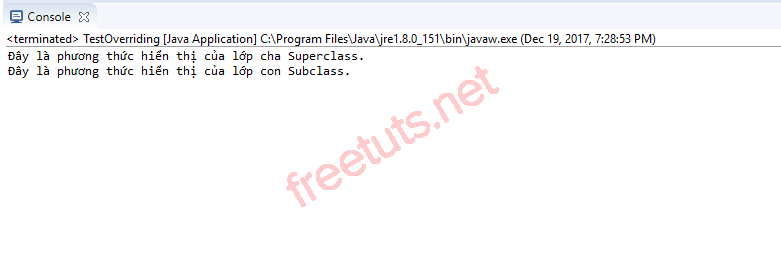
**Subclass.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | package viduoverriding;    public class Subclass extends Superclass {        @Override      public void hienThi() {          System.out.println("Đây là phương thức hiển thị của lớp con Subclass.");      }    } |

**TestOverriding.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | package viduoverriding;    public class TestOverriding {        public static void main(String[] args) {          // khai báo đối tượng của lớp Superclass          Superclass superclass = new Superclass();            // khai báo đối tượng có bản chất là Superclass          // nhưng đóng vai trò là 1 Subclass          // vì vậy sẽ chạy những hàm của Subclass          Superclass subclass = new Subclass();            // gọi phương thức hienThi() của lớp cha          superclass.hienThi();            // gọi phương thức hienThi() của lớp con          subclass.hienThi();      }    } |

Kết quả sau khi biên dịch chương trình:



**2. Overloading**

Trong Java, overloading được định nghĩa là nạp chồng phương thức, có nghĩa là nếu trong một lớp có nhiều phương thức cùng tên nhưng:

* Khác nhau về số đối số truyền vào và các đối số có cùng kiểu dữ liệu.
* Có cùng số đối số truyền vào và các đối số không có cùng kiểu dữ liệu.
* Khác nhau trình tự kiểu dữ liệu của các đối số.

Chúng ta có 2 cách để thực hiện nạp chồng phương thức đó là thay đổi số đối số truyền vào, thay đổi kiểu dữ liệu của các đối số và thay đổi trình tự của các đối số đó.

**Lưu ý:**

* Trong Java, chúng ta không thể thực hiện nạp chồng phương thức chỉ bằng cách thay đổi kiểu trả về của phương thức đó.
* Hàm tạo cũng có thể được nạp chồng.

### **2.1. Thay đổi số đối số truyền vào**

Ví dụ sau sẽ minh họa cách thức thực hiện nạp chồng phương thức bằng cách thay đổi số đối số truyền vào của phương thức:

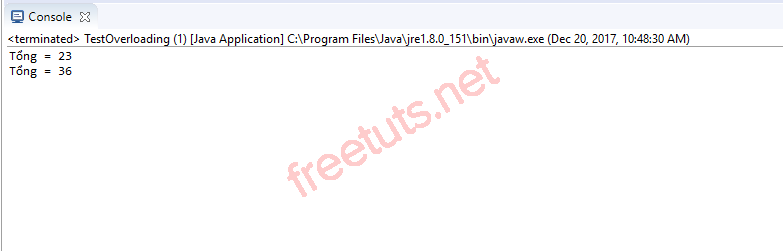
**PhepCongHaiSo.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | package viduoverloading;    public class PhepCongHaiSo {      public int add(int a, int b) {          return a + b;      }        public int add(int a, int b, int c) {          return a + b + c;      }    } |

**TestOverriding.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package viduoverloading;    public class TestOverloading {        public static void main(String[] args) {          PhepCongHaiSo test = new PhepCongHaiSo();          System.out.println("Tổng = " + test.add(11, 12));          System.out.println("Tổng = " + test.add(11, 12, 13));      }    } |

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



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

Trong ví dụ trên, chúng ta có 2 phương thức là add(int a, int b) và add(int a, int b, int c). Hai phương thức này có tên giống nhau, kiểu dữ liệu của các đối số giống nhau nhưng khác nhau về số đối số truyền vào: phương thức add() đầu tiên có 2 đối số có kiểu int là a và b, còn phương thức add() thứ hai có 3 đối số có kiểu int là a, b và c. Đây là trường hợp thứ nhất của nạp chồng phương thức.

### **2.2. Thay đổi kiểu dữ liệu của đối số truyền vào**

Ví dụ sau sẽ minh họa cách thức thực hiện nạp chồng phương thức bằng cách thay đổi kiểu dữ liệu của đối số truyền vào:

**DisplayOverloading.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | package viduoverloading;    public class PhepCongHaiSo {      public int add(int a, int b) {          return a + b;      }        public float add(float a, float b) {          return a + b;      }    } |

**TestOverloading.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package viduoverloading;    public class TestOverloading {        public static void main(String[] args) {          PhepCongHaiSo test = new PhepCongHaiSo();          System.out.println("Tổng = " + test.add(11, 12));          System.out.println("Tổng = " + test.add(11.1f, 12.2f));      }    } |

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



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

Trong ví dụ trên, chúng ta có 2 phương thức là add(int a, int b) và add(double a, double b). Hai phương thức này có tên giống nhau, số đối số truyền vào giống nhau nhưng có kiểu dữ liệu của các đối số khác nhau: phương thức add() đầu tiên có kiểu dữ liệu của các đối số là int và phương thức add() thứ hai có kiểu dữ liệu của các đối số là float. Đây là trường hợp thứ hai của nạp chồng phương thức.

### **2.3. Thay đổi trình tự kiểu dữ liệu của các đối số**

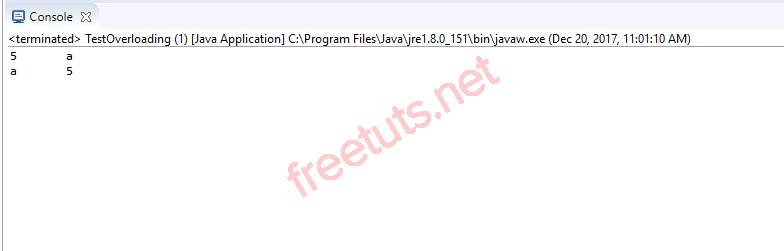
**DisplayOverloading.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | package viduoverloading;    public class DisplayOverloading {        public void hienThi(int a, char b) {          System.out.println(a + "\t" + b);      }        public void hienThi(char b, int a) {          System.out.println(b + "\t" + a);      }  } |

**TestOverloading.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package viduoverloading;    public class TestOverloading {        public static void main(String[] args) {          DisplayOverloading displayOverloading = new DisplayOverloading();          displayOverloading.hienThi(5, 'a');          displayOverloading.hienThi('a', 5);      }    } |

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



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

Trong ví dụ trên, chúng ta có 2 phương thức là hienThi(int a, char b) và hienThi(char b, int a). Hai phương thức này có tên giống nhau, số đối số truyền vào giống nhau nhưng khác nhau về trình tự kiểu dữ liệu của các đối số truyền vào: phương thức hienThi() đầu tiên có 2 đối số có kiểu dữ liệu là int và char, còn phương thức hienThi() thứ hai có 2 đối số có kiểu dữ liệu là char và int. Đây là trường hợp thứ ba của nạp chồng phương thức.

Ngoài ra, trong trường hợp chúng ta chưa xác định số đối số cần truyền vào thì trong Java có một cách nạp chồng phương thức nâng cao như sau:

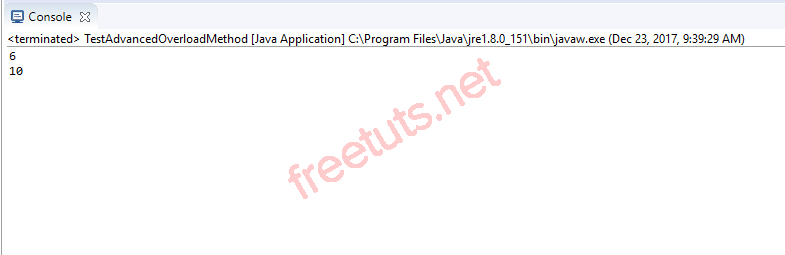
**AdvancedOverloadMethod.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | package viduoverloading;    public class AdvancedOverloadMethod {      // Dấu ... là quy tắc tạo Parameter List      // ...x lúc này là một mảng một chiều linh động      // tức là nó tự động co giãn được đối với các đối số truyền vào.      public int fn4(int ...x) {          int sum = 0;            // duyệt qua từng phần tử trong mảng một chiều x          for (int i : x) {              sum += i;          }          return sum;      }  } |

**TestAdvancedOverloadMethod.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package viduoverloading;    public class TestAdvancedOverloadMethod {        public static void main(String[] args) {          AdvancedOverloadMethod demo = new AdvancedOverloadMethod();          System.out.println(demo.fn4(1, 2, 3));          System.out.println(demo.fn4(1, 2, 3, 4));      }    } |

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



Trong lớp TestAdvancedOverloadMethod tôi khởi tạo đối tượng demo của lớp AdvancedOverloadMethod và tiến hành gọi phương thức fn4() của lớp này với số đối số truyền vào khác nhau. Lần gọi đầu tiên, tôi truyền vào 3 số 1, 2, 3 trong khi ở lần gọi thứ hai, tôi truyền vào 4 số là 1, 2, 3, 4. Thực chất của trường hợp nạp chồng phương thức này chính là nạp chồng phương thức với số lượng các đối số khác nhau.

**3. Khi nào thì sử dụng overriding và overloading?**

Overriding được sử dụng khi trong cùng một phương thức, chúng ta tại muốn thay đổi phần thân của phương thức đó.

Overloading được sử dụng khi trong cùng một phương thức, chúng ta lại muốn làm thêm một công việc khác.