# Lớp giao tiếp (Interface)

# Nội dung

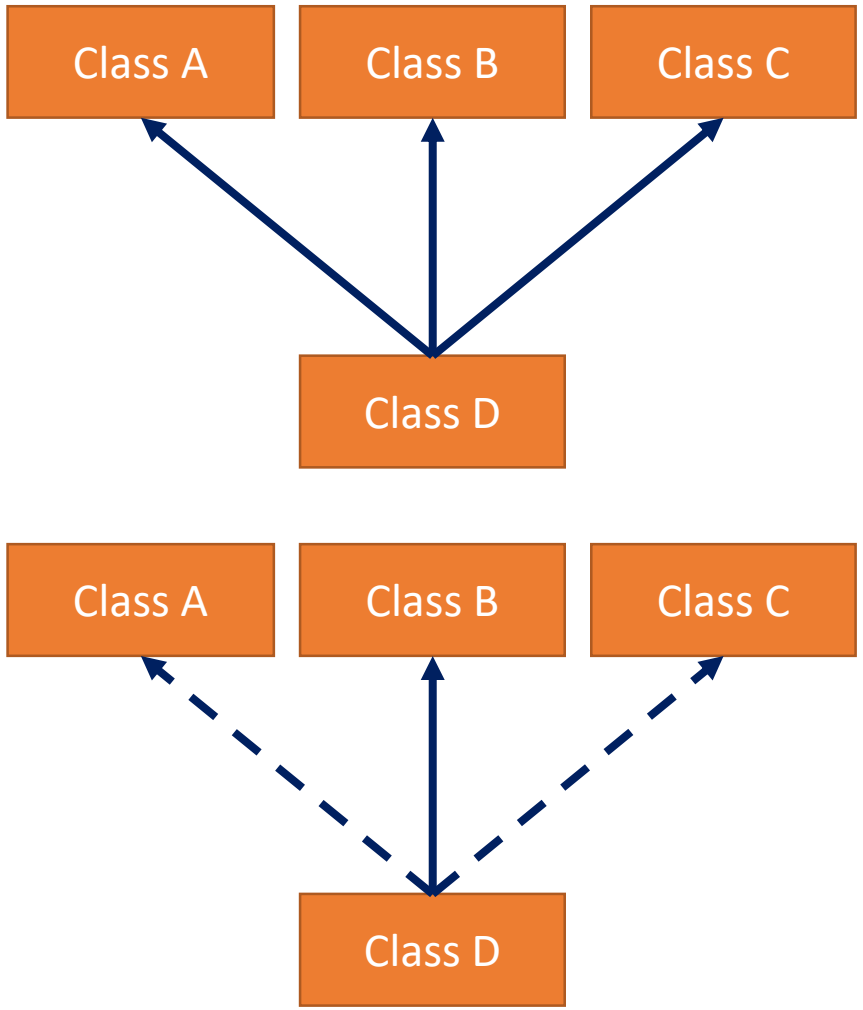
- Các vấn đề trong đa thừa kế
- Khái niệm về interface
- Mối quan hệ class và interface
- So sánh abstract class và interface
- Qui ước và ý nghĩa khi sử dụng interface
- Tóm tắt và kết luận



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Đa kế thừa vs Đơn kế thừa

- Đa thừa kế (Multiple Inheritance)
  - Một lớp có thể thừa kế từ nhiều lớp khác
  - C++ hỗ trợ đa kế thừa
- Đơn thừa kế (Single Inheritance)
  - Một lớp chỉ được kế thừa từ một lớp khác
  - Java chỉ hỗ trợ đơn kế thừa
  - ➢ Khái niệm về lớp giao tiếp (interface) ra đời ⇨ cho phép đa thừa kế

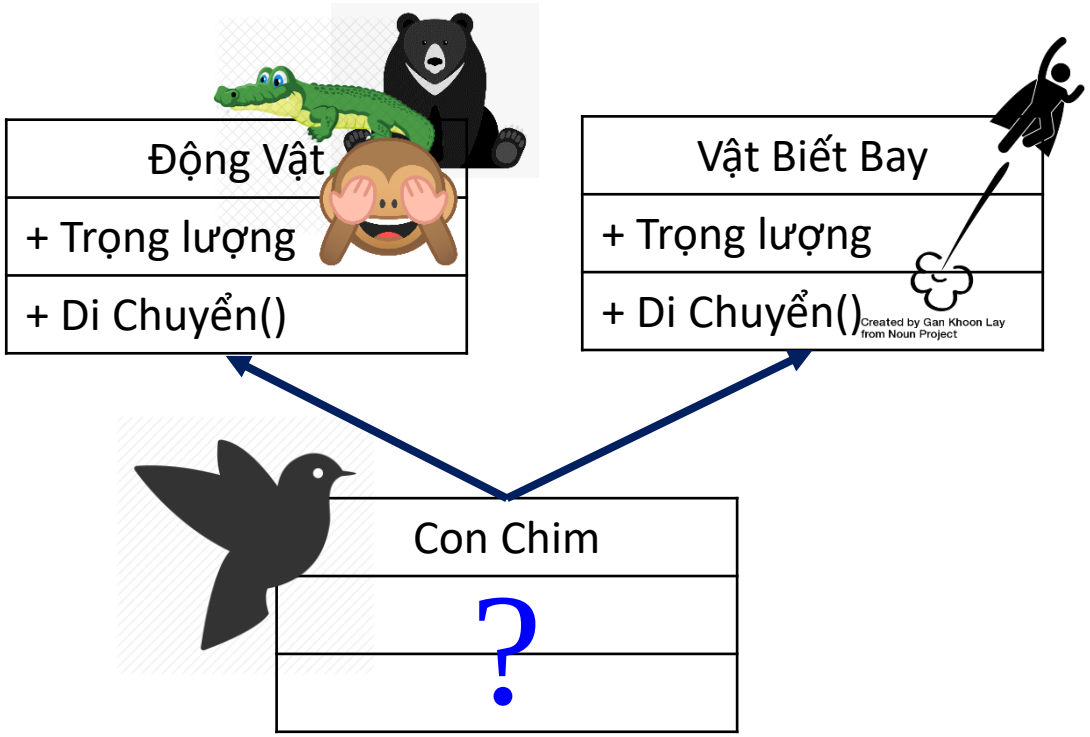




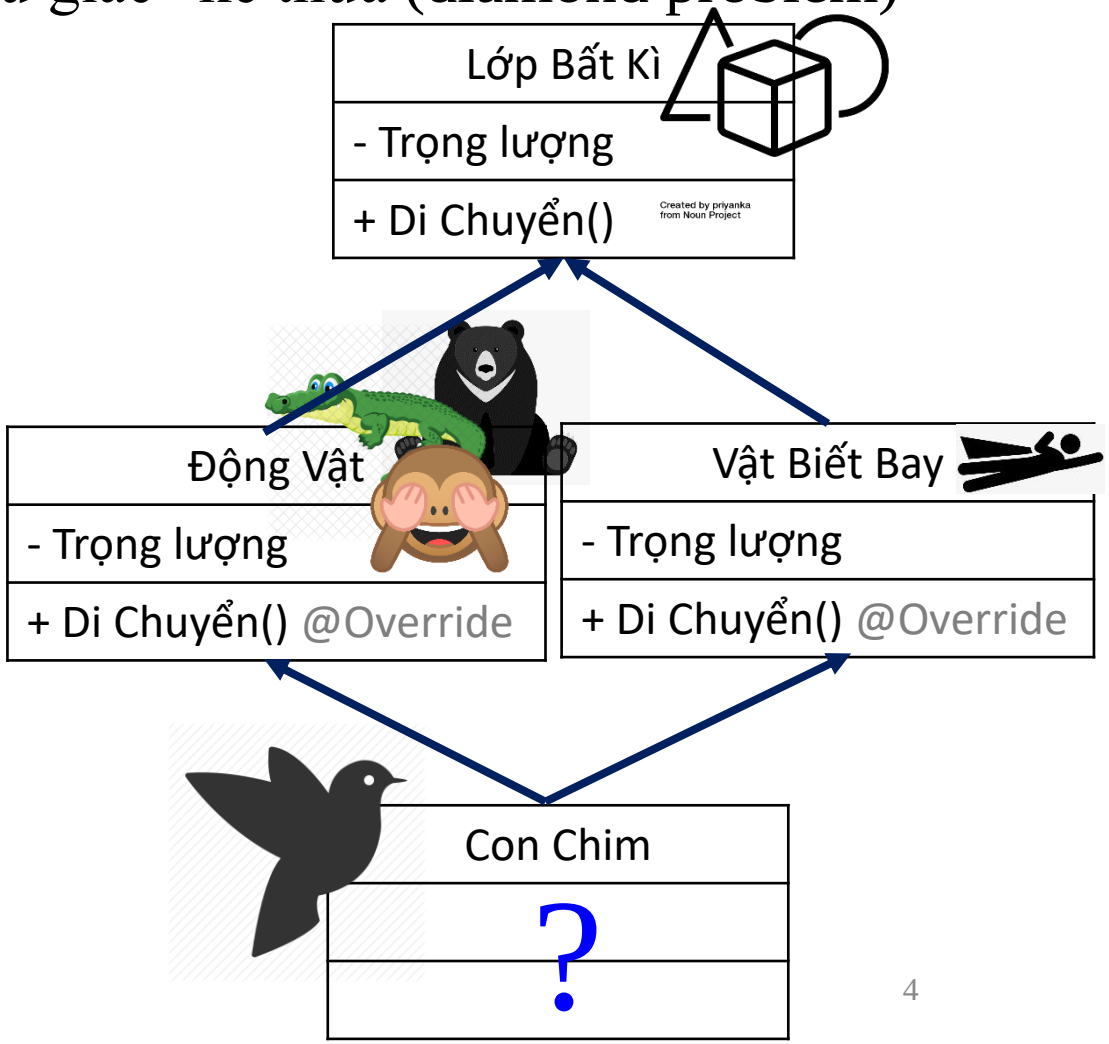

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Vấn đề gặp phải trong đa thừa kế

- Đụng độ giữa các thành phần được kế thừa




- "Tứ giác" kế thừa (diamond problem)






ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Vấn đề gặp phải trong đa thừa kế

- Đụng độ giữa các thành phần được kế thừa
- "Tứ giác" kế thừa (diamond problem)

➔ **Interface** được sinh ra để giải quyết các vấn đề về đa thừa kế



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface

- **Interface** có thể được coi là một lớp giao tiếp hay giao diện
- **Interface** thuộc kiểu dữ liệu tham chiếu (Reference Type) giống **class**
- **Interface** chỉ có thể chứa các thuộc tính hằng số (`static` & `final`) và các phương thức trừu tượng
- **Interface** không thể khởi tạo thành một đối tượng trực tiếp, do đó nó không có chứa phương thức khởi tạo (Constructor)



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface

- Ví dụ

```
interface HìnhĐaGiác{
    // Thành phần dữ liệu là các hằng số
    public static final String màuSắc = "blue";

    // Thành phần xử lý là các phương thức ảo
    public abstract void tínhDiệnTích();
}
```



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface

- Vì các thuộc tính trong **interface** luôn ở dạng hằng số (**static** & **final**) nên ta có thể không cần khai báo một cách rõ ràng nhưng trình biên dịch vẫn hiểu
- Tương tự với các phương thức trừu tượng trong **interface**

```
interface HìnhĐaGiác{
    // Thành phần dữ liệu là các hằng số
    public static final String màuSắc = "blue";

    // Thành phần xử lý là các phương thức ảo
    public abstract void tínhDiệnTích();
}
```

```
interface HìnhĐaGiác {
    // Thành phần dữ liệu
    public String màuSắc = "blue";

    // Thành phần xử lý
    public void tínhDiệnTích();
}
```



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface::Modifier

- Tầm vực truy xuất của một **interface** và tất cả các thành phần thuộc tính và phương thức trong **interface** đều là **public**
- Kể cả khi không cần khai báo, trình biên dịch vẫn luôn mặc định tầm vực truy xuất của **interface** và các thành phần là **public**

```
public interface HìnhĐaGiác{
    // Thành phần dữ liệu là các hằng số
    public static final String màuSắc = "blue";

    // Thành phần xử lý là các phương thức ảo
    public abstract void tínhDiệnTích();
}
```

```
interface HìnhĐaGiác {
    // Thành phần dữ liệu
    String màuSắc = "blue";

    // Thành phần xử lý
    void tínhDiệnTích();
}
```



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface::**Syntax**

- Cú pháp khai báo **interface** trên Java

```
interface <Tên lớp giao tiếp> {…}
```

- Ví dụ

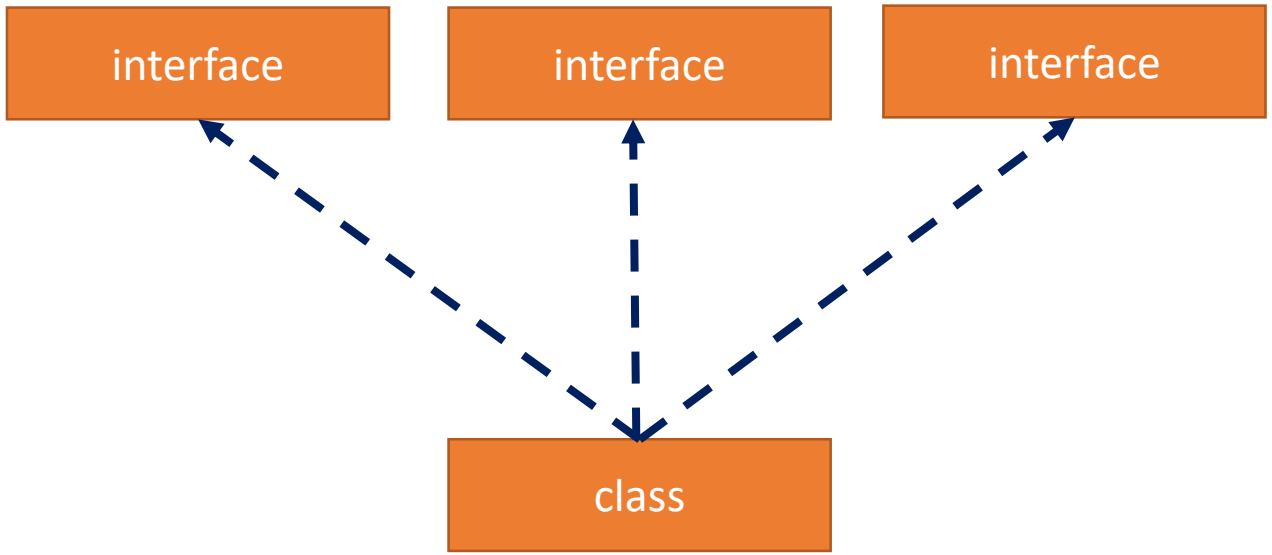
```
public interface DoiXung {…}

public abstract interface GiongNhau {…}

interface TinhTien {…}
```



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface vs Class

- Một **class** có thể kế thừa từ một hoặc nhiều **interface**, tuy nhiên khi một **class** kế thừa từ một **interface** thì nó có trách nhiệm phải hiện thực các phương thức trừu tượng được khai báo trong **interface** đó






ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface vs Class

- Cú pháp (khai báo)

```
[modifier] interface <InterfaceName> {
    // Các thuộc tính hằng số
    // Các phương thức trừu tượng
}
```

- Cú pháp (mở rộng)

```
[modifier] class <ClassName> implements [<Interface1>, <Interface2>,…]  {
    // Thân lớp
}
```



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface vs Class

- Ví dụ

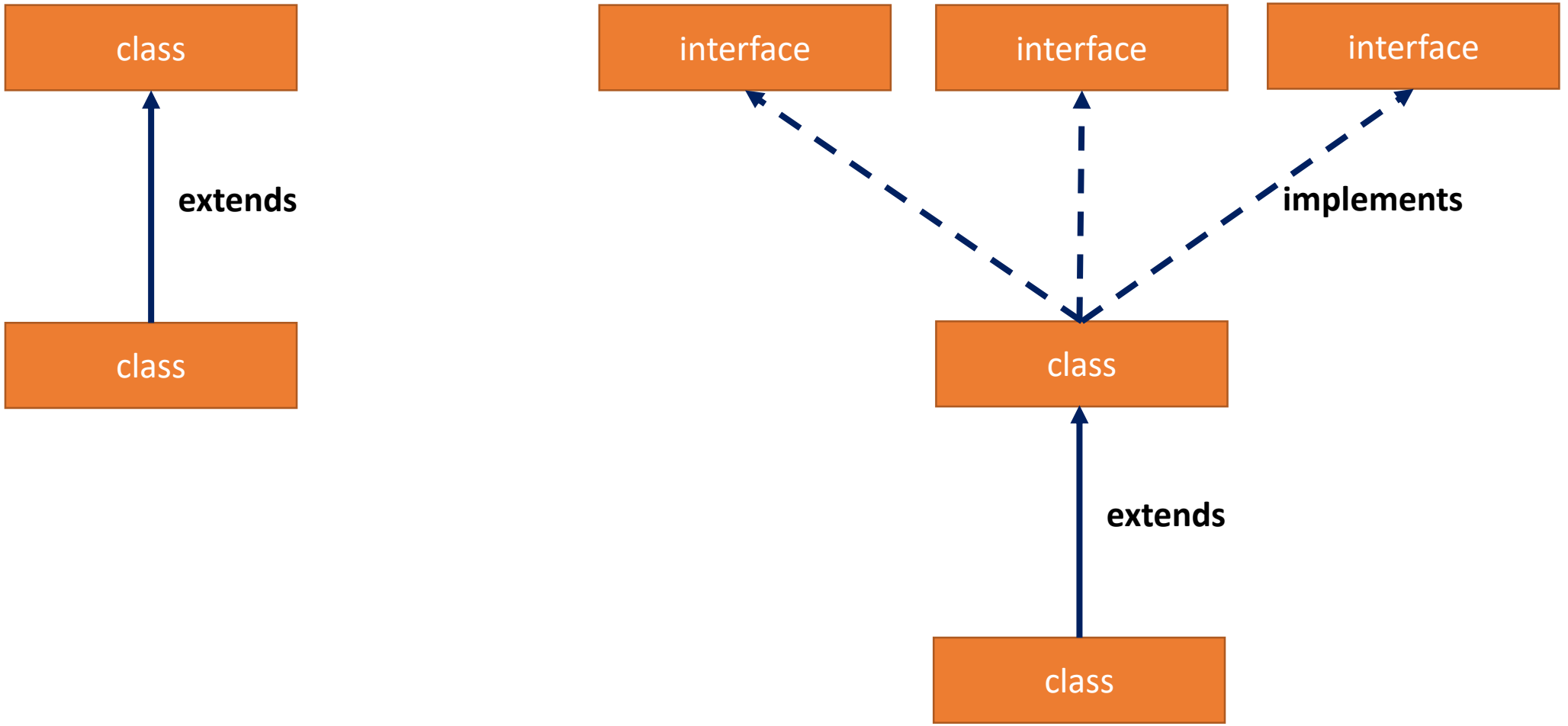
```
//Khả năng tịnh tiến
interface TinhTien {
    public abstract void tinhTienTheoX(double value);
}

// Lớp Đoạn Thẳng
public class DoanThang implements TinhTien {
    // Thành phần dữ liệu
    private double x1, y1;
    private double x2, y2;

    // Thành phần xử lý
    @Override
    public void tinhTienTheoX(double value) {
      this.x1 = this.x1 + value;
      this.x2 = this.x2 + value;
    }
}
```

Lớp DoanThang phải hiện thực phương thức tinhTienTheoX() thông qua cơ chế @Override



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

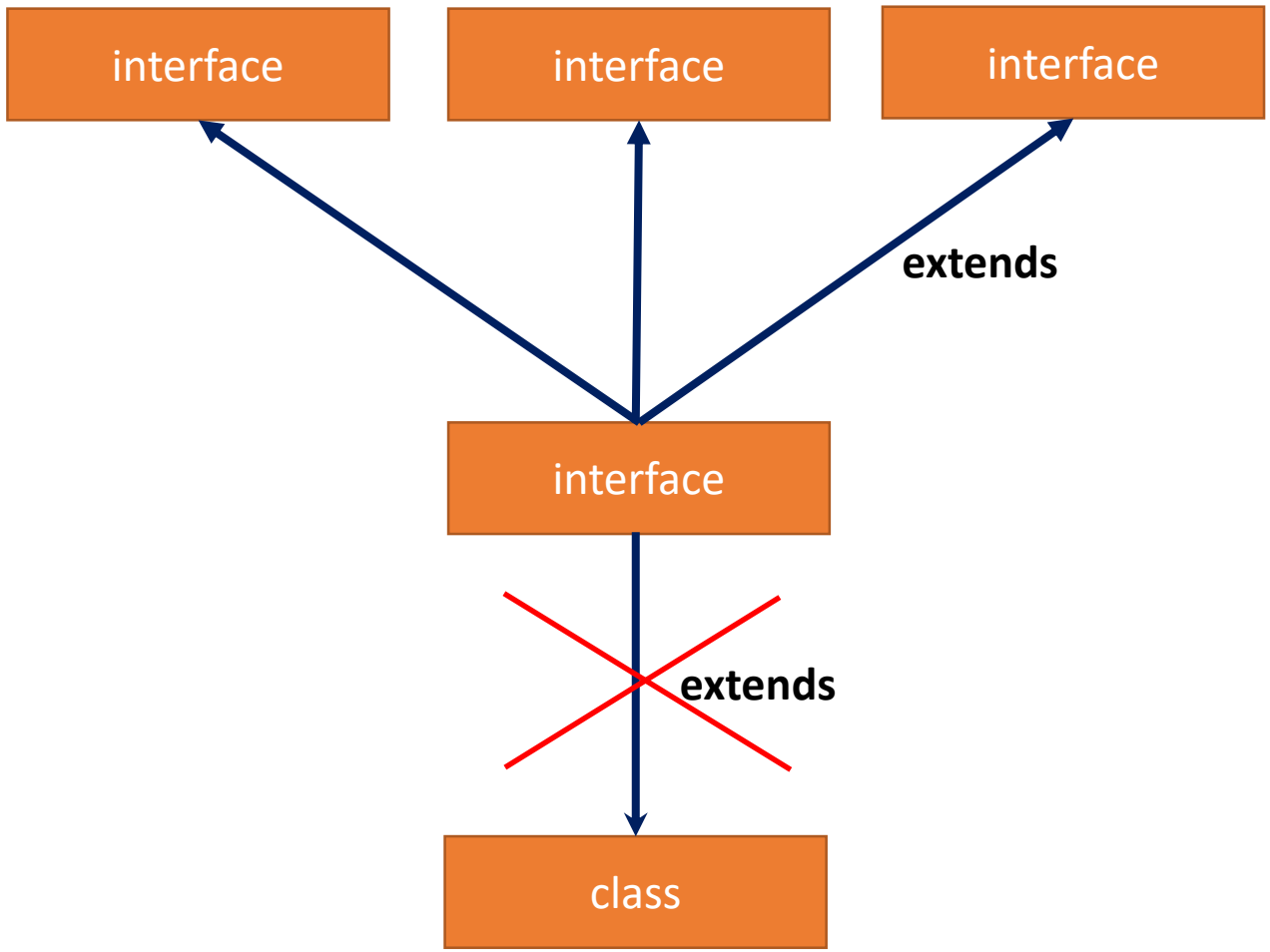
# Mối quan hệ giữa class và interface






ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Mối quan hệ giữa class và interface






ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Cú pháp hiện thực giữa interface vs class

- Cú pháp thừa kế giữa một **class** và một **interface**

  **class <tên lớp> implements <tên interface> {…}**

- Cú pháp thừa kế giữa một **class** và nhiều **interface**

  **class <tên lớp> implements <tên interface 1, tên interface 2,…> {…}**

- Cú pháp thừa kế kết hợp giữa **class** và **interface**

  **class <lớp con> extends <lớp cha> implements <tên interface 1, tên interface 2,…> {…}**



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Cú pháp hiện thực giữa interface vs class

- Ví dụ

**public class HìnhVuông implements ĐốiXứng {…}**

**public class HìnhVuông implements ĐốiXứng,DiChuyển {…}**

**public class HìnhVuông extends HìnhĐaGiác implements ĐốiXứng {…}**

**public class HìnhVuông extends HìnhĐaGiác implements ĐốiXứng, DiChuyển, XoayTròn {…}**



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface vs Class

- Ví dụ

```
public interface IPerson {
    // Nhập thông tin
    void getInfo();

    // Thể hiện thông tin
    void displayInfo();
}
```

```
public interface IStudent extends IPerson
{
    // Nhập điểm
    void setScore();
}
```

```
public class Student implements IStudent {

    @Override
    public void getInfo() {
    // code...
    }

    @Override
    public void displayInfo() {
    // code...
    }

    @Override
    public void setScore() { // Nhập điểm
    // code...
    }

}
```



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface vs Abstract Class

**Lưu ý**: khái niệm lớp giao tiếp (**interface**) rất dễ nhầm lẫn với khái niệm lớp trừu tượng (abstract class) vì

- Cả 2 loại lớp này đều có chứa các phương thức trừu tượng
- Cả 2 loại lớp đều không cho phép khởi tạo đối tượng
- Cả 2 loại lớp đều không có chứa phương thức khởi tạo (Constructor)



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface vs Abstract Class

| Lớp trừu tượng | Lớp giao tiếp |
|---|---|
| Lớp trừu tượng có thể chứa phương thức trừu tượng và không trừu tượng | Lớp giao tiếp chỉ có chứa phương thức trừu tượng, hoặc static |
| Lớp trừu tượng không hỗ trợ đa thừa kế | Lớp giao tiếp hỗ trợ đa kế thừa |
| Lớp trừu tượng có thể có các biến ở dạng final, non-final, static và non-static | Lớp giao tiếp chỉ có các thuộc tính ở dạng static và final |
| Lớp trừu tượng có thể hiện thực lớp giao tiếp | Lớp giao tiếp không thể hiện thực lớp trừu tượng |
| Lớp trừu tượng có thể kế thừa từ một lớp khác và hiện thực nhiều lớp giao tiếp | Lớp giao tiếp chỉ có thể kế thừa từ các lớp giao tiếp khác |
| Lớp trừu tượng có thể có các thành viên ở dạng private, default, protected, public | Lớp giao tiếp chỉ có các thành phần ở dạng public |



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface::Ý nghĩa

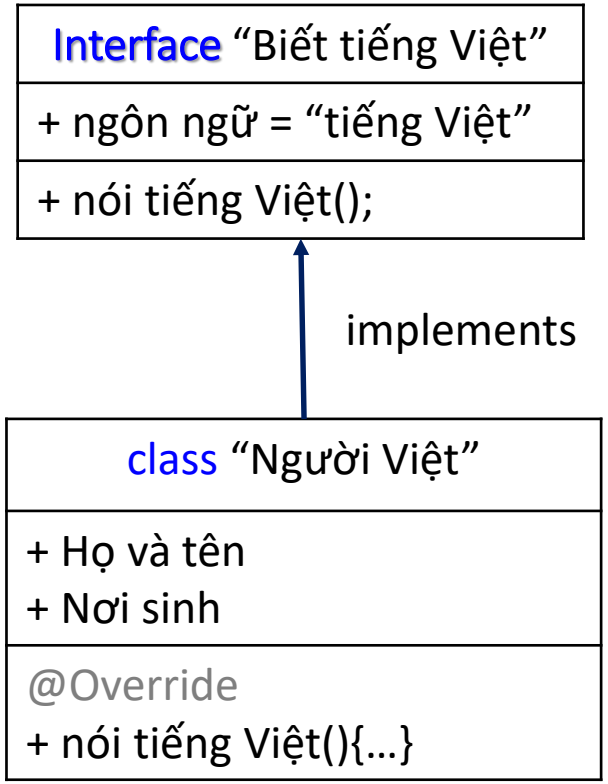
- Về mặt ý nghĩa, lớp giao tiếp (**interface**) giống như một bản vẽ phác thảo các khả năng mà một lớp con hiện thực nó có thể thực hiện được
- Khi một class implements một **interface** thì ta nói class hiện thực|triển khai **interface**



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface::Ý nghĩa

- Ví dụ: các đối tượng thuộc lớp người Việt thì có khả năng nói được tiếng Việt, nhưng việc nói tiếng Việt như thế nào thì lại tùy thuộc vào vùng miền sinh sống của đối tượng (Bắc, Trung, Nam)






ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface::Ý nghĩa

- Một **interface** giống như một bản hợp đồng hoặc một bản qui ước (contract) mà trong đó các nhóm phát triển phần mềm sẽ thống nhất sản phẩm của họ sẽ có những tính năng như thế nào. Sau đó, mỗi nhóm sẽ phát triển độc lập từng tính năng đó cho từng khách hàng cụ thể mà không dẫn đến việc trùng lắp mã nguồn của nhau



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Interface::Ý nghĩa

- Để phân biệt giữa một **interface** với một class về mặt tên gọi thì người ta qui ước đặt tên của **interface** dưới dạng các tính từ chỉ khả năng hành động
- Ví dụ: Moveable, Eatable, Drawable, Printable, …
- Trong trường hợp tên của **interface** là một danh từ thì người ta qui ước thêm vào chữ I ở phía trước để dễ phân biệt với tên của lớp thông thường
- Ví dụ: IPerson, IVietnamese, IStudent,…

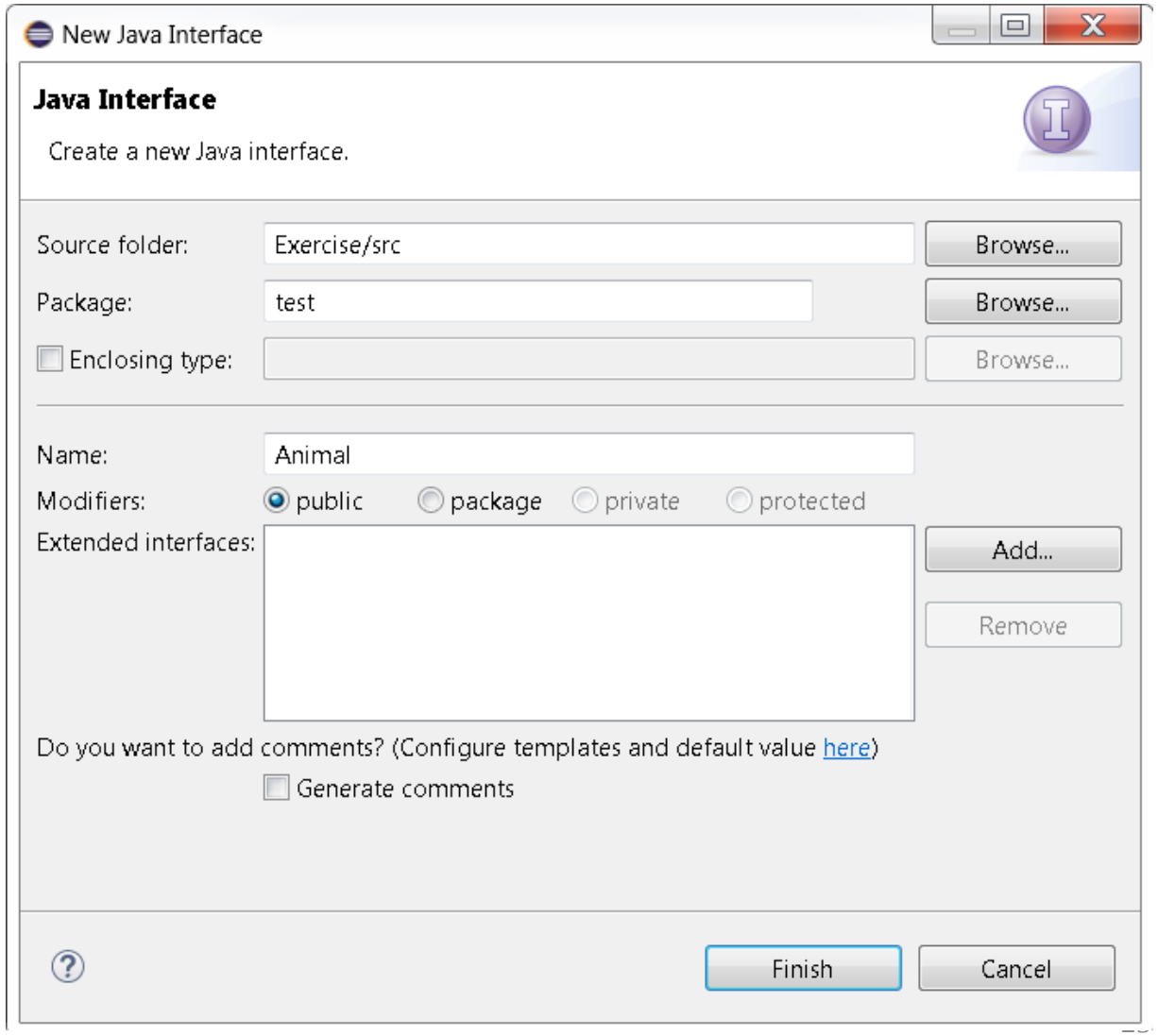


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Tạo interface trực tiếp từ IDE

- Trong Eclipse:

Tên package →New→Interface



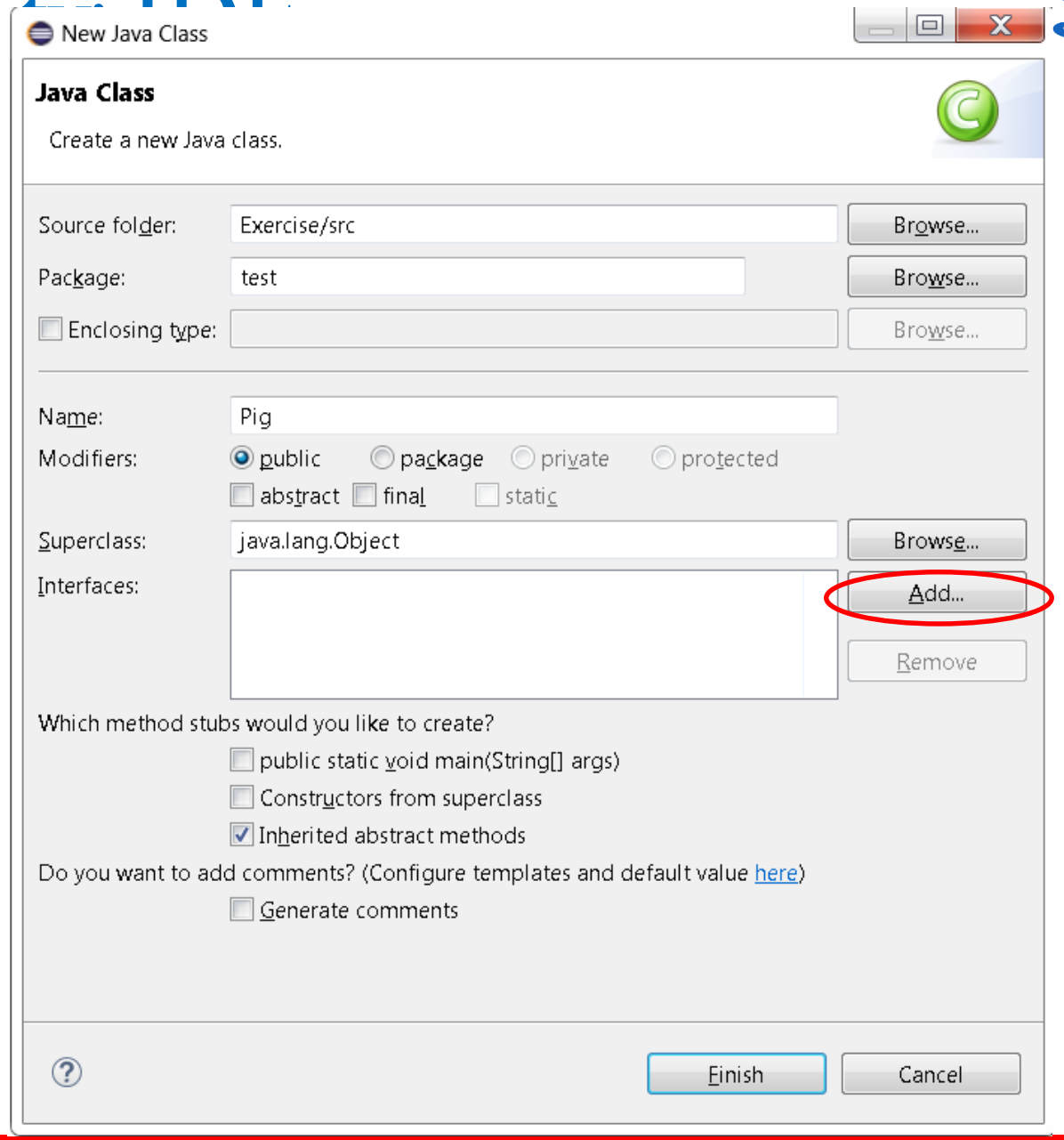



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Tạo interface trực tiếp từ IDE

- Trong Eclipse:

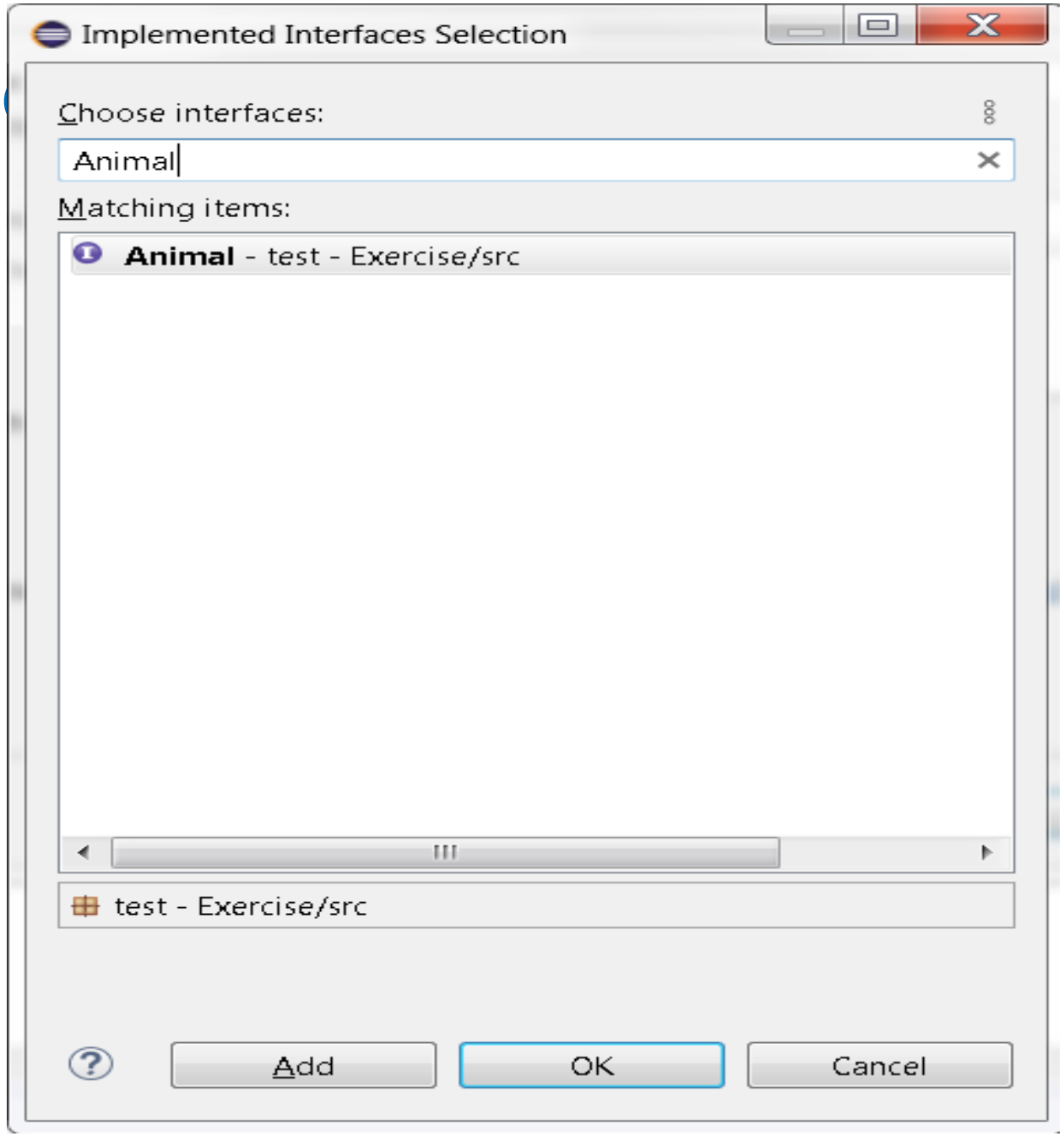
Click vào Add



# Tạo interface trực ti

- Trong Eclipse:

Gõ tên Interface



# Ví dụ: Khai báo interface

```
package mypackage;
interface myinterface{
    void mymethod1();
    void mymethod2();
}
```

# Ví dụ: hiện thực 1 class từ 1 interface

```
package mypackage;
class myclass implements myinterface{

    public void mymethod1(){ //phải là public
        System.out.println("Override my method 1");
    }

    public void mymethod2(){
        System.out.println("Override my method 2");
    }

    void mymethod3(){//không là method trong interface
        System.out.println("My method 3");
    }
}
```



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Ví dụ: sử dụng class đã hiện thực từ interface

```
public static void main (String a[]){

        myclass obj1 = new myclass();
        obj1.mymethod1();
        obj1.mymethod2();
        obj1.mymethod3();

        myinterface obj2 = new myclass();
        obj2.mymethod1();
        obj2.mymethod2();
        obj2.mymethod3();// error;
    }
}
```



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Ví dụ: thừa kế trong interface

```
interface A{
        void meth1();
        void meth2();
}
interface B extends A{
        void meth3();
}
class MyClass implements B{
        public void meth1(){
                System.out.println("Implements method 1");
        }

        public void meth2(){
                System.out.println("Implements method 2");
        }
       public void meth3(){
                System.out.println("Implements method 3");
        }
}
```



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Tóm tắt

- ✓Trong **interface** chỉ bao gồm các ***abstract method*** và các ***biến static & final***
- ✓Khi một class ***triển khai*** từ một **interface**, thì nó bắt buộc ***phải cụ thể hóa*** (*override*) tất cả các ***abstract method*** có trong **interface** tương ứng
- ✓Interface luôn có tầm vực truy xuất là ***public*** và có thể thừa kế
- ✓Một **interface** có thể được ***hiện thực*** bởi nhiều class khác nhau, và một class cũng có thể ***triển khai*** nhiều **interface** khác nhau



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

# Tài liệu tham khảo


- Y. Daniel Lang, "**Introduction to Java Programming Comprehension Version**" 10th Edition.
- Jose M. Garrido, "**Object-Oriented Programming: From Problem Solving to Java**"
- Paul Deitel, Harvey Deitel, "**Java : How to program**", 9th edition, 2012
- Oracle, "**The Java™ Tutorials**", https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html, 6:20PM, 18/01/2018
- Java tutorial, https://howtodoinjava.com/java/basics/, 15:00PM, 20/02/2020





ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn