Fpt University

FPT POLYTECHNIC

cdio™

LẬP TRÌNH JAVA 1

BÀI 8: KIẾN THỨC NÂNG CAO VỀ PHƯƠNG THỨC VÀ LỚP

PHẦN 1

www.poly.edu.vn

Mục tiêu

- Kết thúc bài học này bạn có khả năng
  - Hiểu sâu hơn về hàm tạo
  - Phân biệt được tham biến và tham trị
  - Sử dụng tham số biến đổi
  - Biết cách sử dụng static, final
  - Hiểu thuật toán đệ qui

VẤN ĐỀ VỀ CONSTRUCTOR

1) Nếu một lớp không định nghĩa constructor thì Java tự động cung cấp **constructor mặc định** (không tham số) cho lớp.
2) Trong một constructor muốn gọi constructor khác cùng lớp thì sử dụng **this(tham số)**, muốn gọi constructor của lớp cha thì sử dụng **super(tham số)**
3) Nếu trong constructor không gọi constructor khác thì nó tự gọi constructor không tham số của lớp cha **super()**
4) Lời gọi constructor (super() hoặc this()) khác phải là lệnh đầu tiên
5) Khi đã định nghĩa các constructor cho một lớp thì chỉ được phép sử dụng các constructor này để tạo đối tượng

- Hãy cho biết đoạn mã lệnh sau sai ở đâu? vì sao?

```
public class Parent{
    public Parent(int x){}
}
```

```
public class Child extends Parent{
}
```

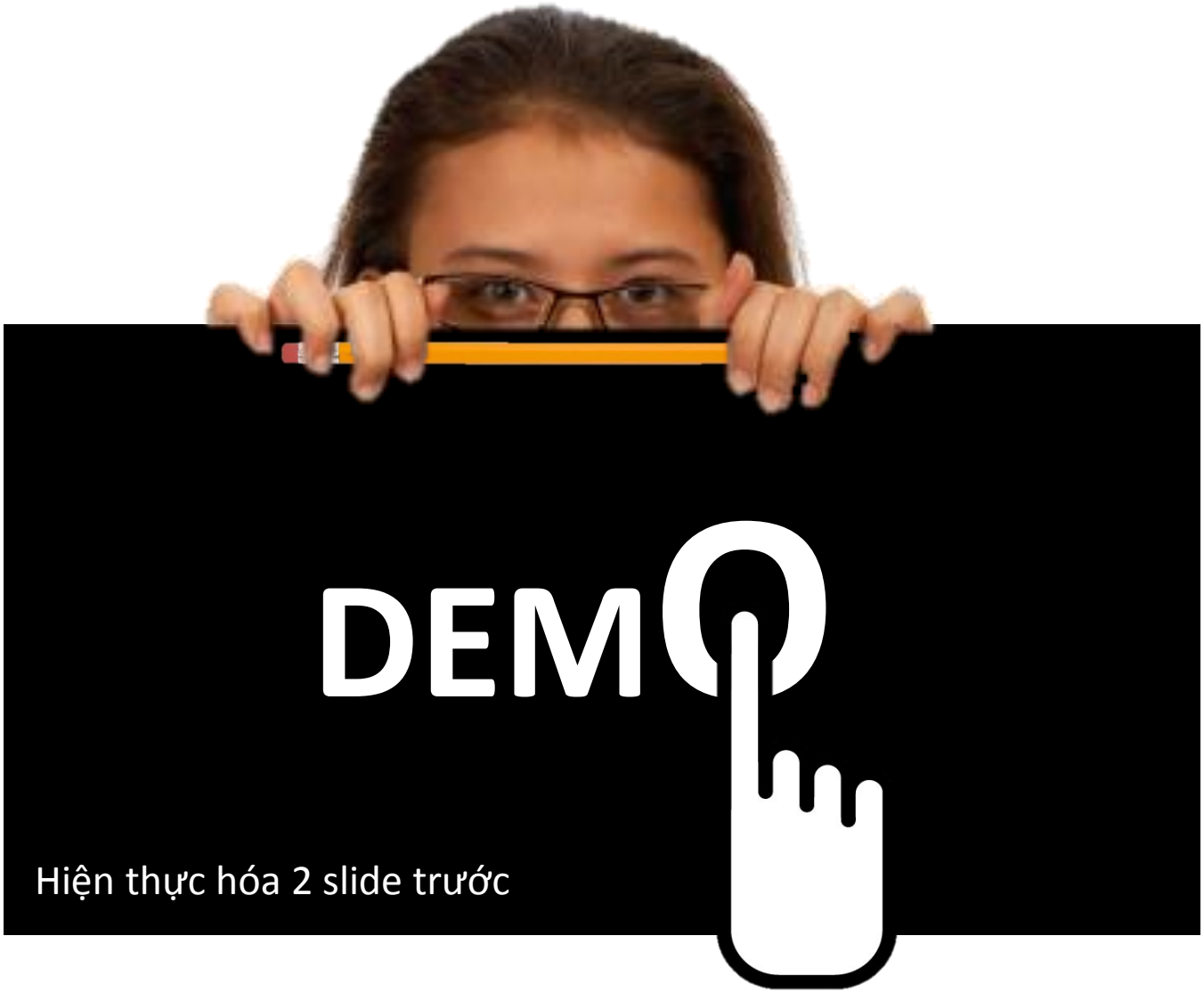
GIẢI THÍCH

- Chiếu theo điều 1) và điều 3) slide trước ta có sơ đồ tương đương

```
public class Parent{
    public Parent(int x){}
}
```

```
public class Child extends Parent{
    public Child(){
        super()
    }
}
```

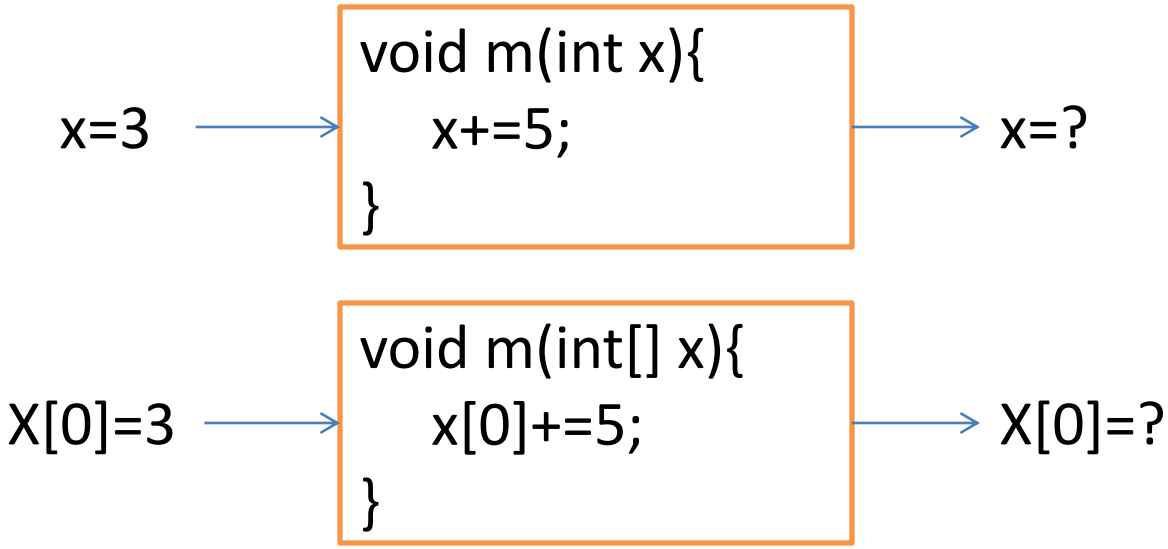
Chiếu theo điều 4 thì Parent không có constructor không tham số nên gây lỗi lúc dịch



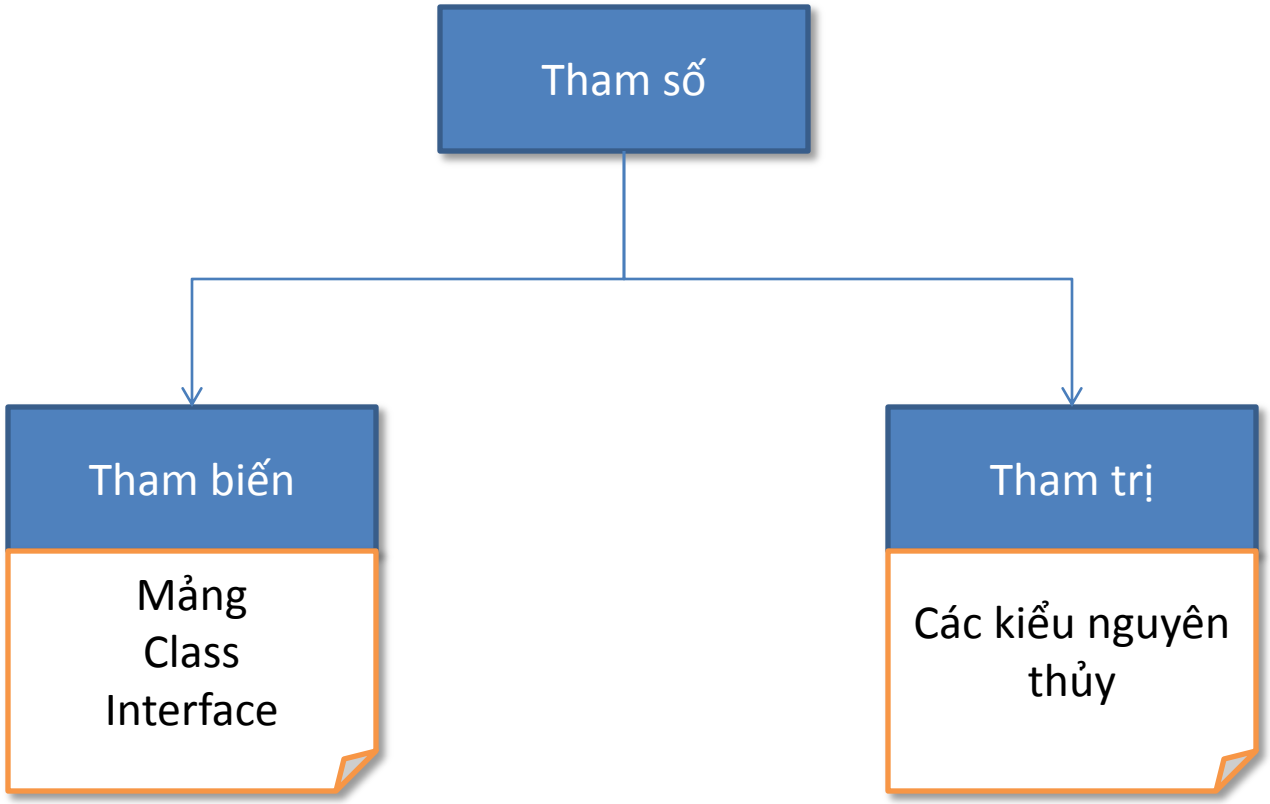
Hiện thực hóa 2 slide trước

THAM SỐ PHƯƠNG THỨC

- Khi truyền tham số vào một phương thức, nếu phương thức có làm thay đổi giá trị của tham số thì giá trị của tham số sau khi gọi phương thức có bị thay đổi hay không?

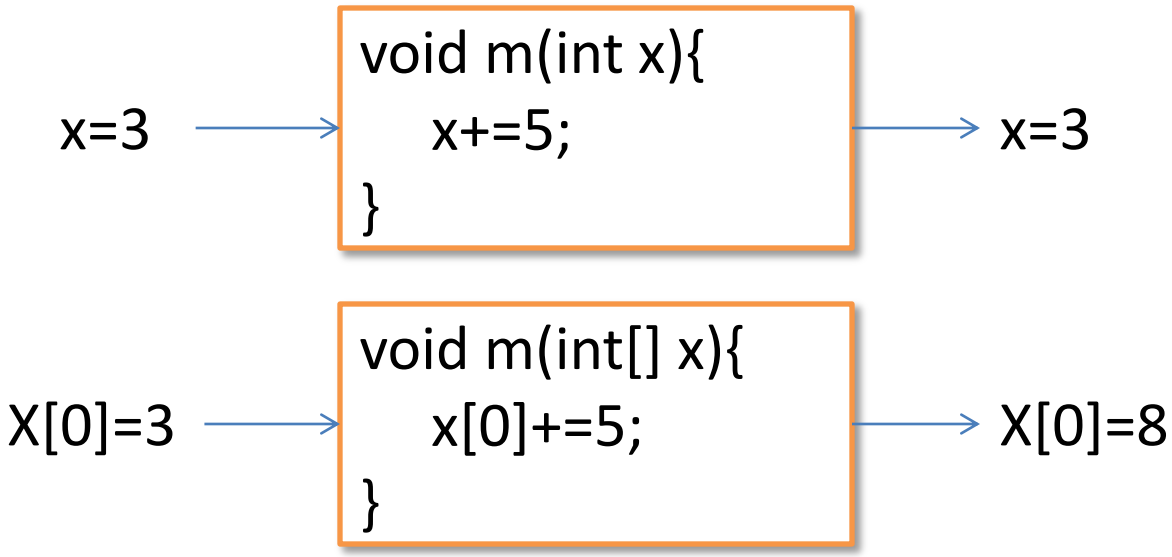


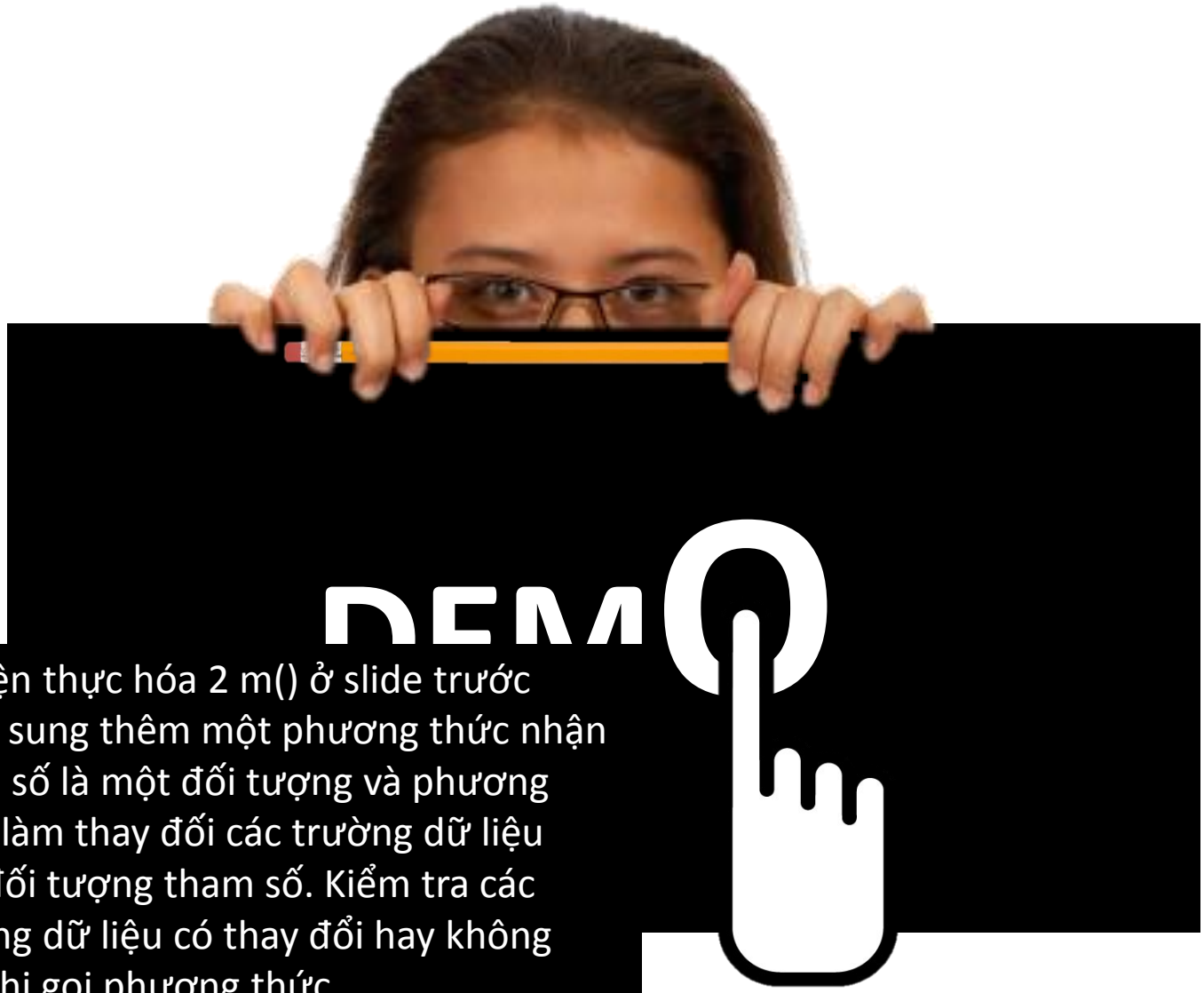
PHÂN LOẠI THAM SỐ



TRUYỀN THAM SỐ CHO PHƯƠNG THỨC

- Khi phương thức làm thay đổi giá trị của tham số thì
  - Nếu là tham trị: giá trị của tham số sẽ không bị thay đổi
  - Nếu là tham biến: giá trị của tham số sẽ bị thay đổi theo

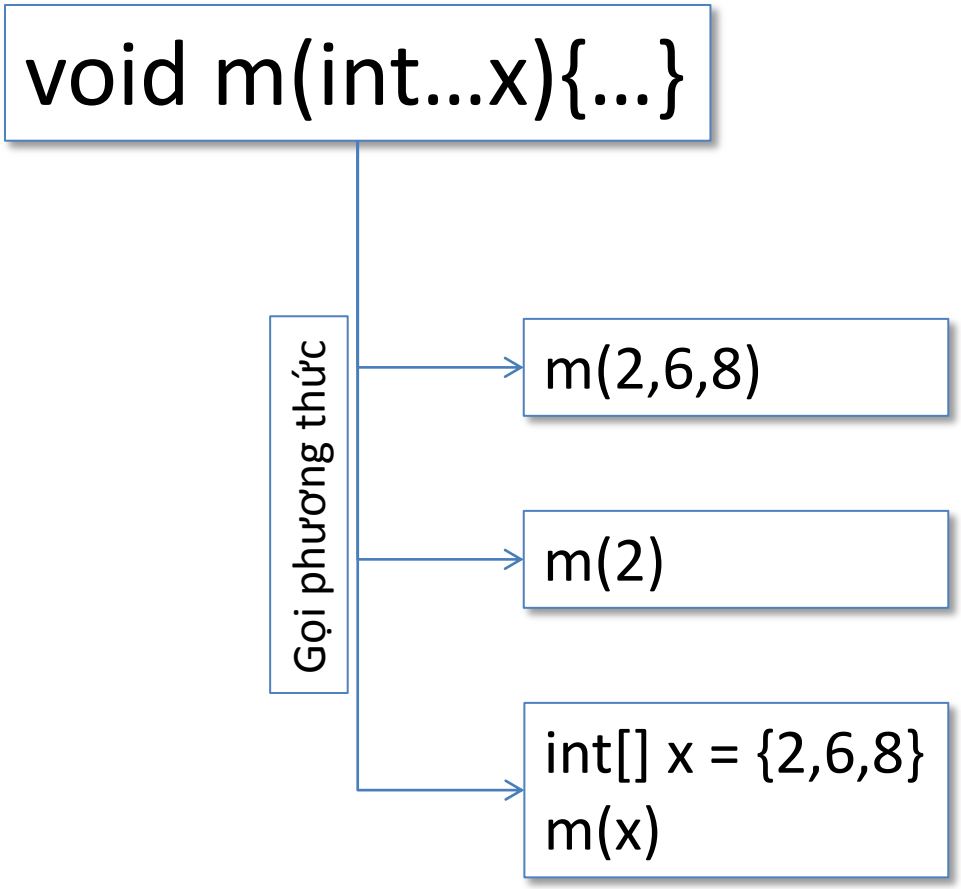




1. Hiện thực hóa 2 m() ở slide trước
2. Bổ sung thêm một phương thức nhận tham số là một đối tượng và phương thức làm thay đổi các trường dữ liệu của đối tượng tham số. Kiểm tra các trường dữ liệu có thay đổi hay không sau khi gọi phương thức

THAM SỐ BIẾN ĐỔI (VARARGS)

- Tham số biến đổi là tham số khi truyền vào phương thức với số lượng tùy ý (phải cùng kiểu).



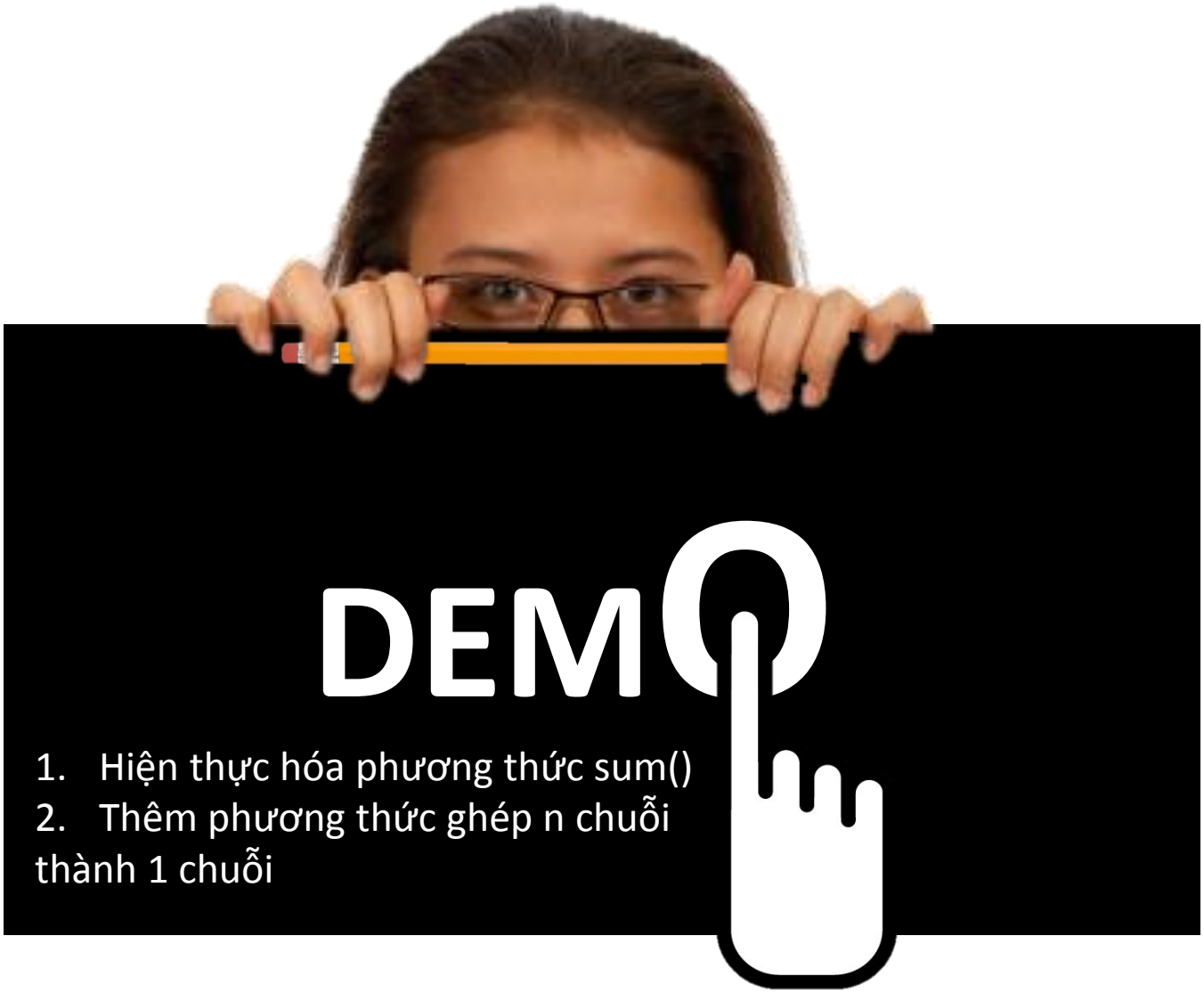
TRUYỀN THAM BIẾN ĐỔI (VARARGS)

- Bản chất của tham số biến đổi là mảng nhưng khi truyền tham số bạn có thể truyền vào nguyên mảng hoặc liệt kê các phần tử
- Trong một hàm, chỉ có thể khai báo **duy nhất** một tham số kiểu varargs và phải là tham số **cuối cùng**

```
int sum(int...x){
    int s = 0;
    for(int a : x){
        s += a;
    }
    return s;
}
```

```
int s1 = sum(2,7)
```

```
int s2 = sum(3,8,3,7,4)
```


DEMO
1. Hiện thực hóa phương thức sum()
2. Thêm phương thức ghép n chuỗi thành 1 chuỗi

LAB 8 BUỔI 1

- Lab 8 – bài 1
- Lab 8 – bài 2

Fpt University

FPT POLYTECHNIC

cdio™

LẬP TRÌNH JAVA 1

BÀI 8: KIẾN THỨC NÂNG CAO VỀ PHƯƠNG THỨC VÀ LỚP

PHẦN 2

www.poly.edu.vn

- Từ khóa static được sử dụng để định nghĩa cho khối và các thành viên tĩnh (**lớp nội, phương thức, trường**).

```
public class MyClass{
    static public int X;
    static{
        X+=100;
    }
    static public void method(){
        X+=200;
    }
    static class MyInnerClass{}
}
```



```
MyClass.X = 700;
MyClass.method()
```

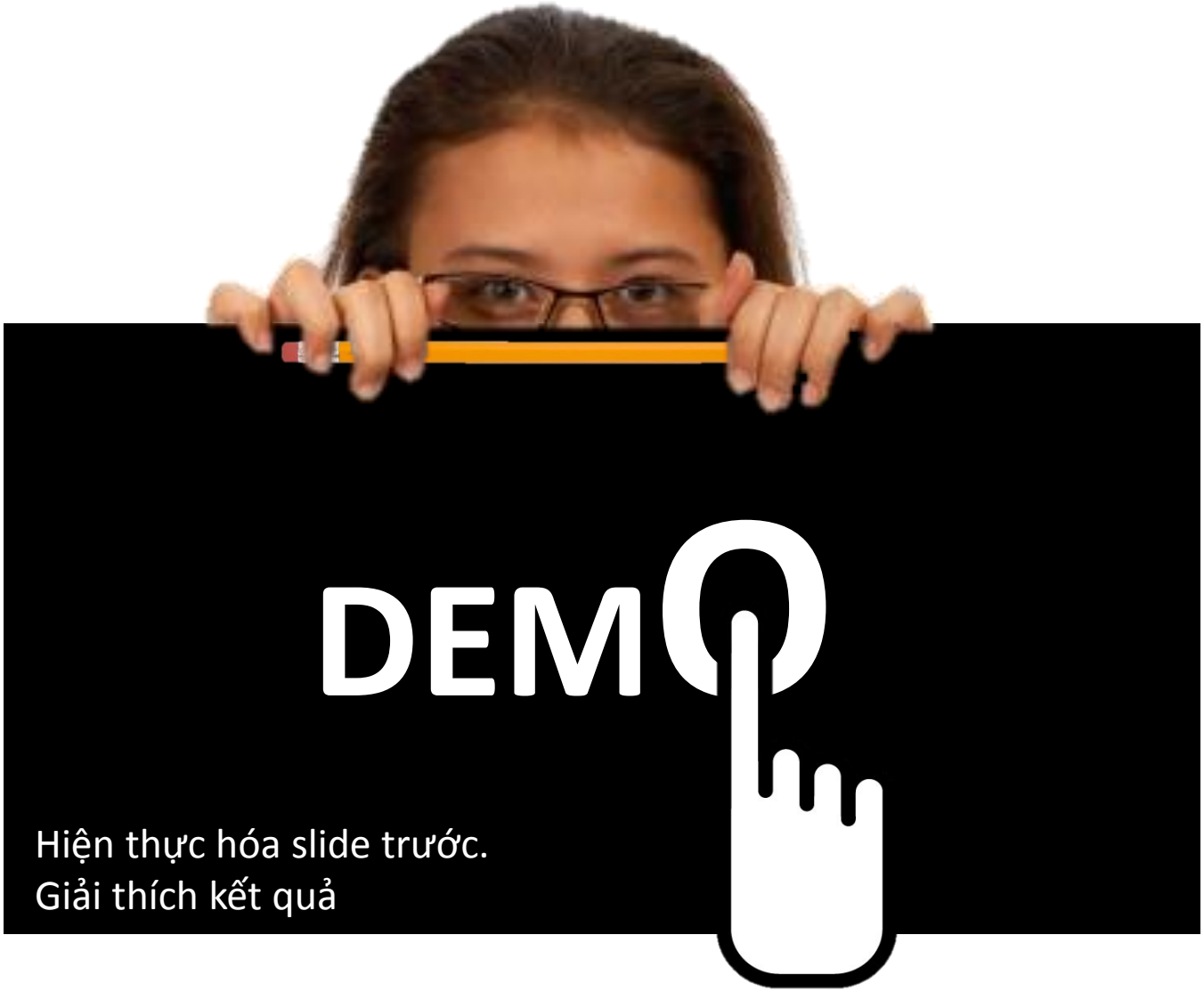
STATIC

- Khối static {} sẽ **chạy trước** khi tạo đối tượng hoặc truy xuất bất kỳ thành viên tĩnh khác
- Thành viên tĩnh của lớp được sử dụng **độc lập** với các đối tượng được tạo ra từ lớp đó.
- Có thể truy cập đến một thành viên tĩnh thông qua **tên lớp** mà không cần tham chiếu đến một đối tượng cụ thể
- Trường static là dữ liệu **dùng chung** cho tất cả các đối tượng được tạo ra từ lớp đó.
- Trong khối và phương thức tĩnh **chỉ được truy cập đến các thành viên tĩnh** khác mà không được phép truy cập đến thành viên thông thường của class

```
public class MyClass{
    static public int X = 100;
    static{
        X+=100;
    }
    static public void method(){
        X+=200;
    }
}
```

```
MyClass o = new MyClass();
o.X += 300;
MyClass.X += 500;
MyClass.method()
```

MyClass.X, o.X có giá trị là bao nhiêu


DEMO
Hiện thực hóa slide trước.
Giải thích kết quả

ĐỊNH NGHĨA HẰNG

- Trong Java có 3 loại hằng
  - Lớp hằng là lớp không cho phép thừa kế
  - Phương thức hằng là phương thức không cho phép ghi đè
  - Biến hằng là biến không cho phép thay đổi giá trị
- Sử dụng từ khóa final để định nghĩa hằng

```
final public class MyFinalClass{...}
```

```
public class MyClass{
    final public double PI = 3.14
    final public void method(){...}
}
```

CHỌN ĐOẠN MÃ ĐÚNG

A

```
final public class Parent{…}
```

```
public class Child extends Parent{
…
}
```

B

```
public class MyClass{
    final int PI = 3.14;
    public void method(){
        PI = 3.1475;
    }
}
```

C

```
public class Parent{
    final public void method(){…}
}
```

```
public class Child extends Parent{
    public void method(){…}
}
```

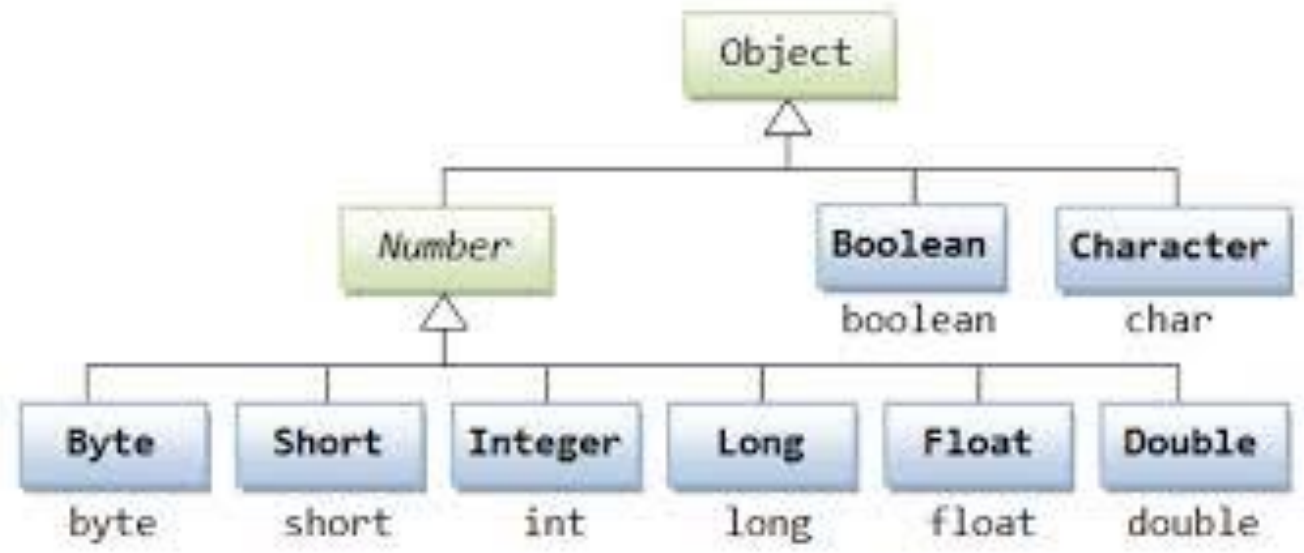
D

```
public class Parent{
    public void method(){…}
}
```

```
public class Child extends Parent{
    public void method(){…}
}
```

LỚP OBJECT

- Khi định nghĩa một lớp mà không kế thừa từ một lớp khác thì mặc định kế thừa lớp Object
- Như vậy mọi lớp đều có lớp cha chỉ duy nhất một lớp không có cha là Object



LỚP NỘI

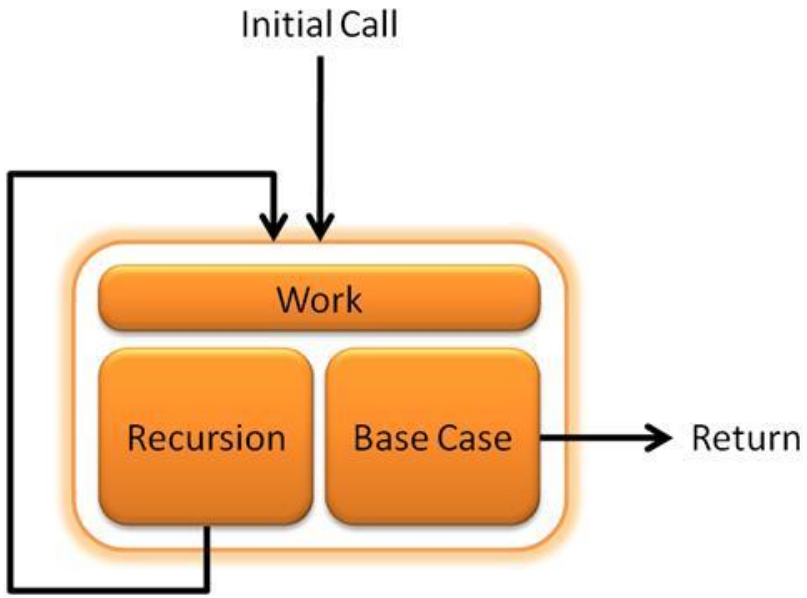
- Lớp nội là lớp được khai báo bên trong một lớp khác
- Có hai loại: lớp nội tĩnh và lớp nội thông thường
- Lớp bên trong chỉ có thể xác định trong phạm vi lớp ngoài cùng và có thể truy cập các thành viên của lớp bao nó

```
public class MyClass{
    static public class MyInnerStaticClass{}
    public class MyInnerClass{}
}
```

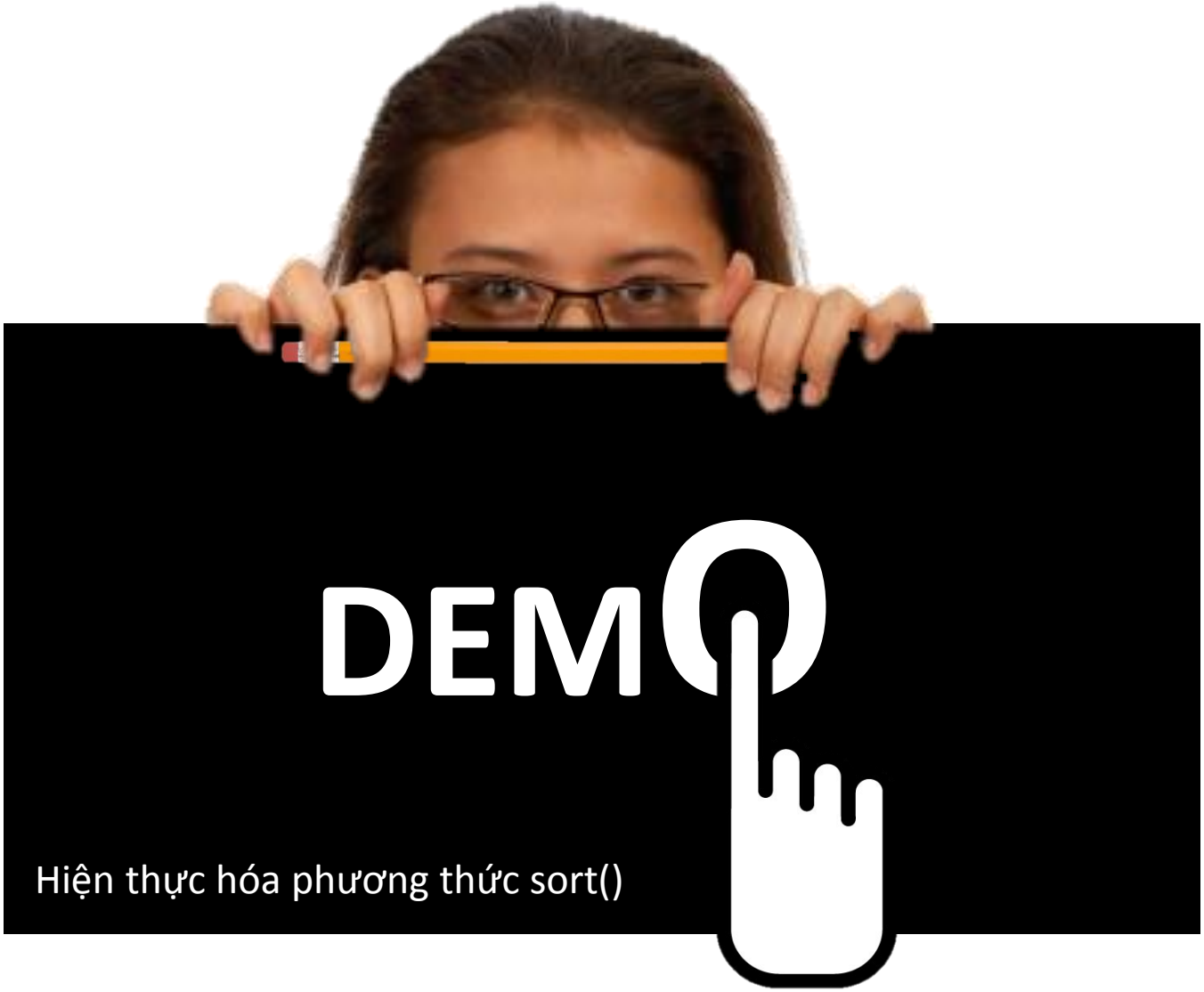
Sử dụng lớp nội

```
MyClass.MyInnerStaticClass x = new MyClass.MyInnerStaticClass();
MyClass.MyInnerClass y = new MyClass().new MyInnerClass();
```

- Một phương thức gọi chính nó
- Phải có lệnh dừng đệ quy trong phương thức để tránh vòng lặp vô hạn
- Đệ qui dễ hiểu nhưng rất tốn tài nguyên




```
public void sort(int[] a, int i){
        if(i >= a.length){
            return;
        }
        for(int j = i + 1; j < a.length; j++){
                if(a[i] < a[j]){
                    int tmp = a[i];
                    a[i] = a[j];
                    a[j] = tmp;
                }
        }
        sort(a, i + 1);
}
```


DEMO
Hiện thực hóa phương thức sort()

TỔNG KẾT NỘI DUNG BÀI HỌC

- Tìm hiểu sâu về constructor
- Phân loại tham số
- Tham số biến đổi
- Sử dụng static
- Định nghĩa hằng
- Lớp nội
- Đệ quy

LAB 8 BUỔI 2

- Lab 8 – bài 3
- Lab 8 – bài 4
- Lab 8 – bài 5 (giảng viên cho thêm)