

# Bài 5. Lớp và đối tượng

## THCS 4: Lập trình cơ bản với Java

Đỗ Thanh Hà, Nguyễn Thị Minh Huyền

Bộ môn Tin học  
Khoa Toán - Cơ - Tin học  
Trường Đại học Khoa học Tự nhiên

# Vài điều lưu ý khi lập trình

## Ví dụ 1. Tìm chỉ số của phần tử nhỏ nhất mảng

---

```
public static int getMinIndex(int[] values){
    int minValue = Integer.MAX_VALUE;
    int minIndex = -1;
    for(int i=0; i<values.length; i++){
        if (values[i] < minValue){
            minValue = values[i];
            minIndex = i;
        }
    }
    return minIndex;
}
```

---

?

## Ví dụ 2. Tìm chỉ số của phần tử nhỏ thứ 2 trong mảng

```
public static int getSecondMinIndex(int[] values){
    int secondIdx = -1;
    int minIdx= getMinIndex(values);
    for(int i=0; i<values.length; i++){
        if (i == minIdx)
            continue;
        if (secondIdx == -1 || values[i] <
            values[secondIdx])
            secondIdx = i;
    }
    return secondIdx;
}
```

- $values = \{0\}?$ ,  $values = \{0, 0\}?$ ,  $values = \{0, 1\}?$
- Có các vấn đề gì chưa được giải quyết trong đoạn mã trên?

## Ví dụ 2. Tìm chỉ số của phần tử nhỏ thứ 2 trong mảng

```
public static int getSecondMinIndex(int[] values){
    int secondIdx = -1;
    int minIdx= getMinIndex(values);
    for(int i=0; i<values.length; i++){
        if (i == minIdx)
            continue;
        if (secondIdx == -1 || values[i] <
            values[secondIdx])
            secondIdx = i;
    }
    return secondIdx;
}
```

- values = {0}? , values = {0, 0}? , values = {0, 1}?
- Có các vấn đề gì chưa được giải quyết trong đoạn mã trên?

## Ví dụ 2. Tìm chỉ số của phần tử nhỏ thứ 2 trong mảng

```
public static int getSecondMinIndex(int[] values){
    int secondIdx = -1;
    int minIdx= getMinIndex(values);
    for(int i=0; i<values.length; i++){
        if (i == minIdx)
            continue;
        if (secondIdx == -1 || values[i] <
            values[secondIdx])
            secondIdx = i;
    }
    return secondIdx;
}
```

- $values = \{0\}?$ ,  $values = \{0, 0\}?$ ,  $values = \{0, 1\}?$
- Có các vấn đề gì chưa được giải quyết trong đoạn mã trên?

# Vấn đề thường gặp (1)

- Mảng **chỉ số** và mảng **giá trị**

---

```
int[] values = {99, 100, 101};  
System.out.println(values[0] ); // 99
```

---

Values	99	100	101
Indexes	0	1	2

## Vấn đề thường gặp (2)

- Dấu ngoặc sau `if/else`, `for/while`

---

```
for(int i= 0; i< 5; i++)  
    System.out.println("Hello, World");  
    System.out.println("Goodbye");
```

---

- Kết quả in ra màn hình?



## Vấn đề thường gặp (3)

### ■ Giá trị khởi tạo

---

```
int secondIdx = 0;
int minIdx= getMinIndex(values);
for(int i=0; i<values.length; i++){
    if (i == minIdx)
        continue;
    if (values[i] < values[secondIdx])
        secondIdx = i;
}
```

---

### ■ Kết quả trong trường hợp values = {0,1,2}?

## Vấn đề thường gặp (4)

### ■ Cấp phát bộ nhớ cho mảng

---

```
int arrInt[];  
arrInt = new int[4];  
arrInt[0] = 19;  
arrInt[1] = 7;  
arrInt[2] = 3;  
arrInt[3] = 24;
```


```
String arrString[] = {"Tran A", "Nguyen H"};
```

---

## Vấn đề thường gặp (5)

- Định nghĩa phương thức trong phương thức

```
public static void main(String[] arguments) {  
    public static void foobar () {  
    }  
}
```



- Lập trình hướng đối tượng
- Định nghĩa lớp
- Sử dụng lớp
- Tham chiếu và giá trị
- Dữ liệu và phương thức tĩnh

# Lập trình hướng đối tượng (1)

- Mô hình hoá các đối tượng trong thế giới thực
  - Baby
    - Tên
    - Ngày sinh
    - Giới tính
    - Cân nặng
    - Chiều dài
    - ...
- Chương trình thực hiện tương tác giữa các đối tượng

## Lập trình hướng đối tượng (2)

- Chương trình Java thao tác trên các dữ liệu kiểu:
  - Nguyên thủy (int, double, char, v.v.)
  - Đối tượng (String, v.v.)

### **Baby**

```
String name;  
String date;  
boolean isMale;  
double weight;  
double length;
```

- Lớp (class) xác định kiểu đối tượng: dữ liệu của đối tượng, thao tác trên các dữ liệu
- Tại sao sử dụng lớp
  - Có thể hiểu lớp là kiểu dữ liệu có cấu trúc
  - Nếu chỉ sử dụng kiểu cơ bản: quản lí 500 babies ????

# Tại sao lại dùng lớp (ví dụ)

- Ta có thể sử dụng các kiểu cơ bản

---

```
String name[];  
String date[];  
boolean isMale[];  
double weight[];  
double length[];
```

---

**Sắp xếp 500 trẻ nhỏ theo thứ tự tên?**



# Định nghĩa lớp

# Định nghĩa các lớp

---

```
public class Baby{
    String name;
    String date;
    boolean isMale;
    double weight;
    double length;

    public void sayHi(){
        // Say Hi to everybody
    }
}
```

---

```
public class Baby {
```



fields



methods

```
}
```

- Tên lớp: quy ước luôn được viết hoa chữ cái đầu
  - Các tên lớp thư viện đã từng dùng: **System**, **String**, **Math**, **Scanner**
- Mỗi lớp được lưu trong một tệp riêng (**<ClassName>.java**)
- Lớp chứa phương thức **public static void main(String[])** (chương trình chính) là lớp để thực thi.

## Các trường dữ liệu

- Là các biến lưu trữ các thuộc tính của mỗi đối tượng
- Khai báo, khởi gán như các biến

```
public class Baby {
```

```
    TYPE var_name;
```

```
    TYPE var_name = some_value;
```

```
}
```

# Các trường dữ liệu

---

```
public class Baby{  
    String name;  
    String date;  
    boolean isMale;  
    double weight = 3.2;  
    double length;  
    // methods ...  
}
```

---

# Tạo đối tượng của lớp

- `Baby aBaby = new Baby();`
  - `Baby()` là hàm dựng (*constructor*) của lớp `Baby`
- Hàm dựng: được gọi khi tạo đối tượng, thường dùng để khởi tạo giá trị cho các thuộc tính
- Mọi lớp `ClassName` đều có hàm dựng ngầm định `ClassName()`
  - Khi sử dụng hàm dựng ngầm định, các thuộc tính của đối tượng có giá trị như trong khai báo/khởi gán
- Có thể viết nhiều hàm dựng phục vụ việc tạo đối tượng khác nhau tùy mục đích
  - `ClassName(arg)`

---

```
public class ClassName{  
    // Constructor definition: no return value,  
    // no "static" keyword  
    public ClassName(){  
    }  
    public ClassName(arg){  
    }  
}  
  
// Use constructors to create objects in another  
    method  
ClassName obj1 = new ClassName();  
ClassName obj2 = new ClassName(arg);
```

---



# Ví dụ hàm dựng của lớp Baby

---

```
public class Baby{
    String name;
    boolean isMale;
    // other fields/attributes...
    public Baby(String myname, boolean maleBaby){
        name = myname;
        isMale = maleBaby;
    }
    // other methods ...
}
```

---

# Các phương thức (1)

- Hàm dựng
- Các phương thức trong lớp thông thường sẽ không phải phương thức tĩnh (có từ khoá **static**)

---

```
public class Baby{  
    String name = "Alex";  
    //.....  
    public void sayHi(){  
        System.out.println("Hi, my name is" + name);  
    }  
}
```

---

## Các phương thức (2)

---

```
public class Baby{  
    double weight = 3.2;  
    //.....  
    public void eat(double foodWeight){  
        if (foodWeight >=0 && foodWeight < weight){  
            weight = weight + foodWeight;  
        }  
    }  
}
```

---

---

```
public class Baby{
    String name;
    String date;
    boolean isMale;
    double weight = 3.2;
    boolean length;
    //.....

    public void sayHi(){
        //.....
    }
    public void eat(double foodWeight){
        //.....
    }
}
```

---

# Sử dụng lớp

- Định nghĩa lớp
  - `public class Baby{...}`
- Tạo các thực thể (đối tượng) của lớp
  - `Baby bb1 = new Baby("Alex", true);`
  - `Baby bb2 = new Baby("Julien", true);`

## ■ Object.FIELDNAME

---

```
Baby bb1 = new Baby("Alex", true);  
System.out.println(bb1.name);  
System.out.println(bb1.weight);
```

---

## ■ Object.METHODNAME([args])

---

```
Baby bb1 = new Baby("Alex", true);  
bb1.sayHi(); // Hi, my name is Alex  
bb1.eat(1);
```

---

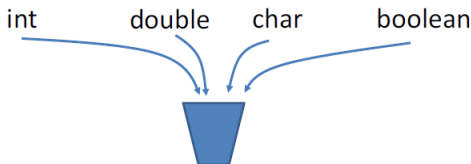


# Tham chiếu và giá trị

# Tham chiếu và giá trị (1)

Kiểu dữ liệu trong Java: Kiểu nguyên thủy (*primitive types*)

- boolean, int, double, char, ...
- Giá trị thực tế được lưu trực tiếp trong biến



## Tham chiếu và giá trị (2)

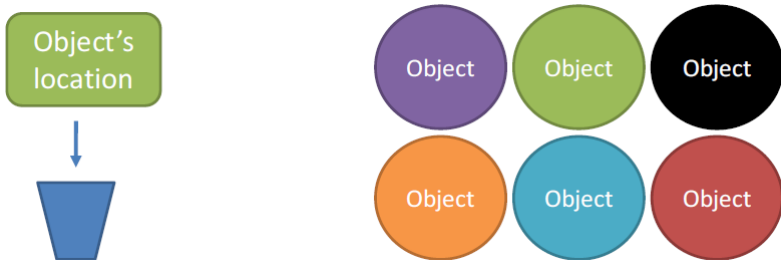
Kiểu dữ liệu trong Java: Kiểu tham chiếu (*reference types*)

- Kiểu mảng và các kiểu đối tượng, int[], String,
- Biến lưu trữ địa chỉ của vùng nhớ lưu trữ đối tượng



## Tham chiếu và giá trị (3)

- Biến lưu trữ địa chỉ của vùng nhớ lưu trữ đối tượng



- Phép toán `==`

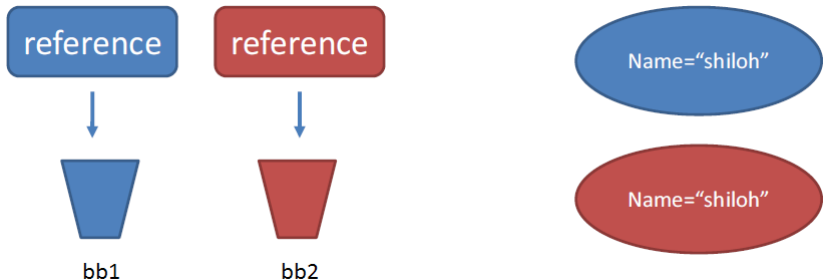
- `Baby bb1 = new Baby("shiloh");`
- `Baby bb2 = new Baby("shiloh");`
- `bb1 == bb2?`

- Biến lưu trữ địa chỉ của vùng nhớ lưu trữ đối tượng
- Phép toán `==`
  - `Baby bb1 = new Baby("shiloh");`
  - `Baby bb2 = new Baby("shiloh");`
  - `bb1 == bb2?`

**FALSE**

## Tham chiếu và giá trị (5)

- `Baby bb1 = new Baby("shiloh");`
- `Baby bb2 = new Baby("shiloh");`



## Tham chiếu và giá trị (6)

```
Baby mybaby = new Baby("davy", true);  
mybaby.name = "david";
```

mybaby's  
location



~~name = 'davy'~~  
ismale = true  
...

## Tham chiếu và giá trị (7)

```
Baby mybaby = new Baby("davy", true);  
mybaby.name = "david";
```

mybaby's  
location



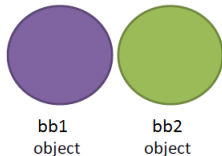
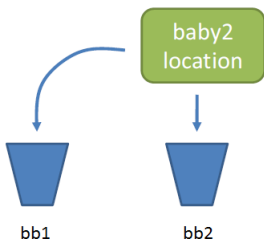
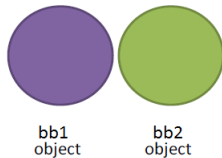
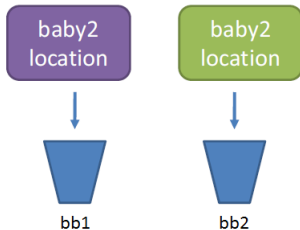
name = 'david'  
ismale = true

...



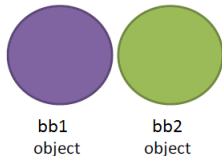
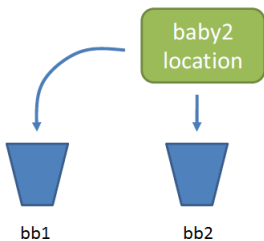
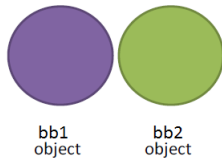
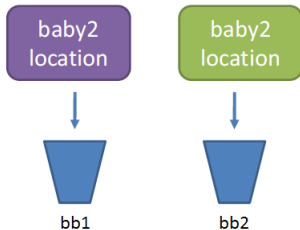
# Tham chiếu và giá trị (8)

- Sử dụng phép gán = để cập nhật tham chiếu
- `bb1 = bb2`



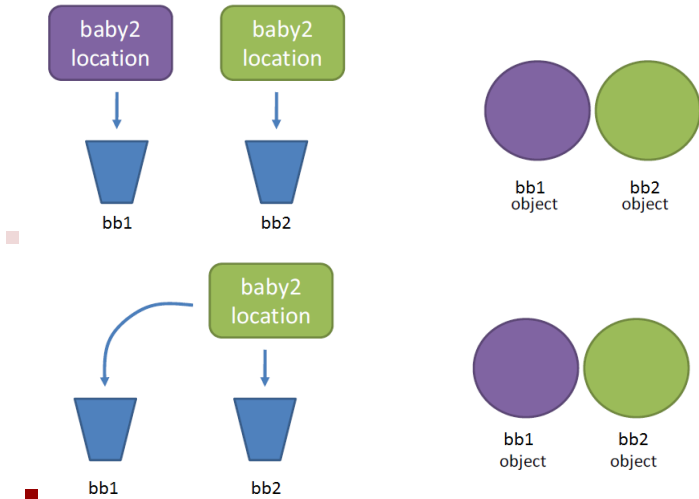
# Tham chiếu và giá trị (8)

- Sử dụng phép gán = để cập nhật tham chiếu
- `bb1 = bb2`



# Tham chiếu và giá trị (8)

- Sử dụng phép gán = để cập nhật tham chiếu
- `bb1 = bb2`



## Tham chiếu và giá trị (10)

---

```
public static void doSomething(int x, int[] ys, Baby b){  
    x = 99;  
    ys[0] = 99;  
    b.name = "99";  
}  
//..... in main  
int i= 0;  
int[] j = {0};  
Baby bb1 = new Baby("50", true);  
doSomething(i, j, bb1);
```

---

**i = ? j = ? bb1 = ?**

# Biến và phương thức tĩnh

- Từ khóa: **static**
- Trường dữ liệu và phương thức tĩnh:
  - Thuộc tính và phương thức của lớp
  - Dùng chung cho tất cả các đối tượng của lớp được tạo ra
  - Có thể truy cập qua tên lớp, không cần tạo đối tượng
    - `ClassName.staticField`
    - `ClassName.staticMethod(args)`

---

```
public class Baby{
    static int numBabiesMade = 0;
}
// .....
Baby.numBabiesMade = 100;
Baby bb1 = new Baby();
Baby bb2 = new Baby();
Baby.numBabiesMade = 2;
System.out.println(bb1.numBabiesMade); // ???
System.out.println(bb2.numBabiesMade); // ???
```

---

## Phương thức tĩnh (2)

- Các phương thức không tĩnh có thể gọi đến phương thức tĩnh trong cùng một lớp
- Ngược lại, các phương thức tĩnh không thể gọi tới phương thức không tĩnh trong cùng một lớp

---

```
public class Baby{  
    String name = "Julien";  
  
    public static void whoAmI(){  
        System.out.println(name);  
    }  
}
```

---

- Phương thức main phải là phương thức tĩnh (tại sao???)
  - public static void main(String args[])
- Các phương thức đã dùng trong lớp *Math* (*Math.sin(double)*, *Math.sqrt(double)*) là các phương thức tĩnh, trường dữ liệu *out* của lớp *System* là biến tĩnh.