# THCS 4: LẬP TRÌNH CƠ BẢN VỚI JAVA

## KIỂM SOÁT QUYỀN TRUY CẬP, PHẠM VI BIẾN LỚP, TỔ CHỨC GÓI

Nguyễn Thị Tâm
nguyenthitam.hus@gmail.com

Ngày 14 tháng 11 năm 2024

# Ôn tập

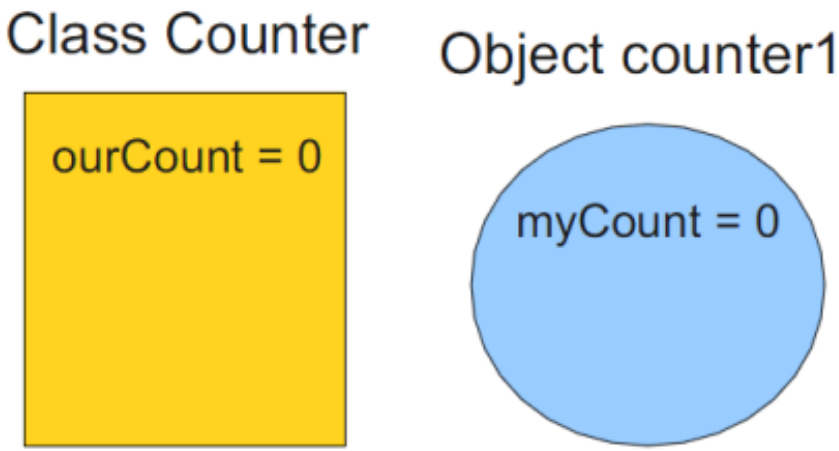
```java
public class TestMatrix{
    public static void printMatrix (int a[][])
    {
        for (int i = 0;i<a.length;i++){
            for (int j = 0;j < a[0].length; j++){
                System.out.print(a[i][j] + " ");
            }
            System.out.println();
        }

    }
    public static void main(String arg[]){
        int a[][] = {{1, 4, 5}, {4, 5, 7}};
        int b[][] = new int[a.length][a[0].length];
        b = a;
        b[0][0] = 8;
        printMatrix(a);
    }
}
```

# Ôn tập (1)

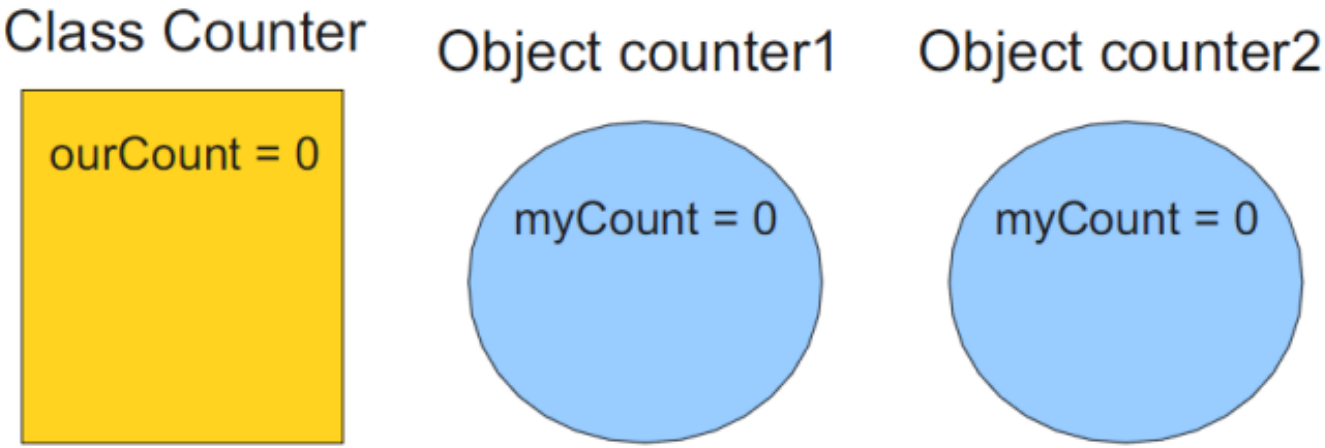
```
public class Counter{
    int myCount = 0;
    static int ourCount = 0;
    void increment(){
        myCount++;
        ourCount++;
    }
    public static void main(String arg[]){
        Counter counter1 = new Counter();
        Counter counter2 = new Counter();
        counter1.increment();
        counter1.increment();
        counter2.increment();
        System.out.println("counter 1: " +
        counter1.myCount + " " + counter1.ourCount);
        System.out.println("counter 2: " +
        counter2.myCount + " " + counter2.ourCount);
    }
}
```

# Ôn tập (2)



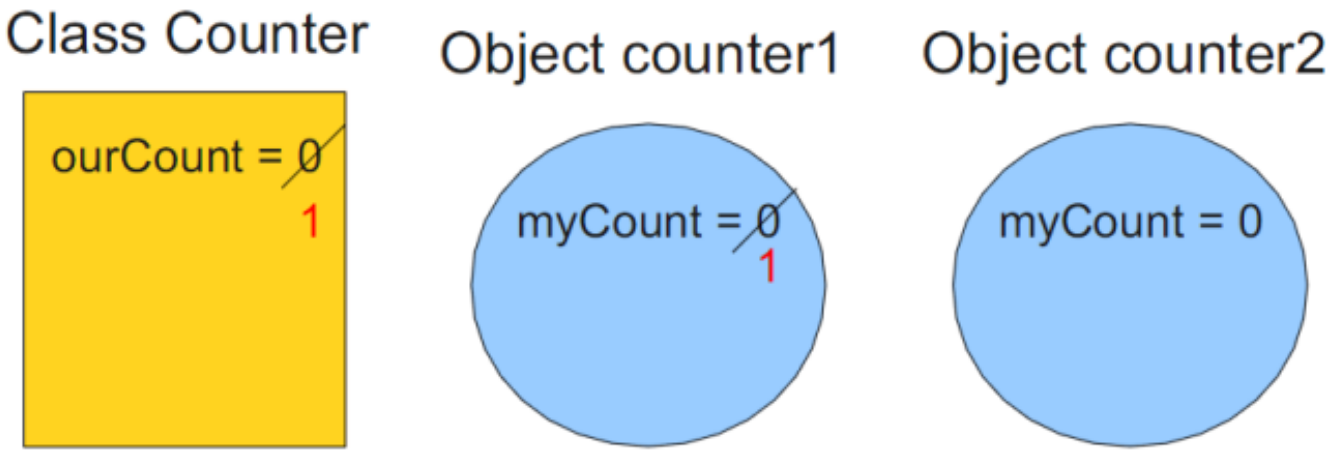

```
Counter counter1 = new Counter();
```

# Ôn tập (2)



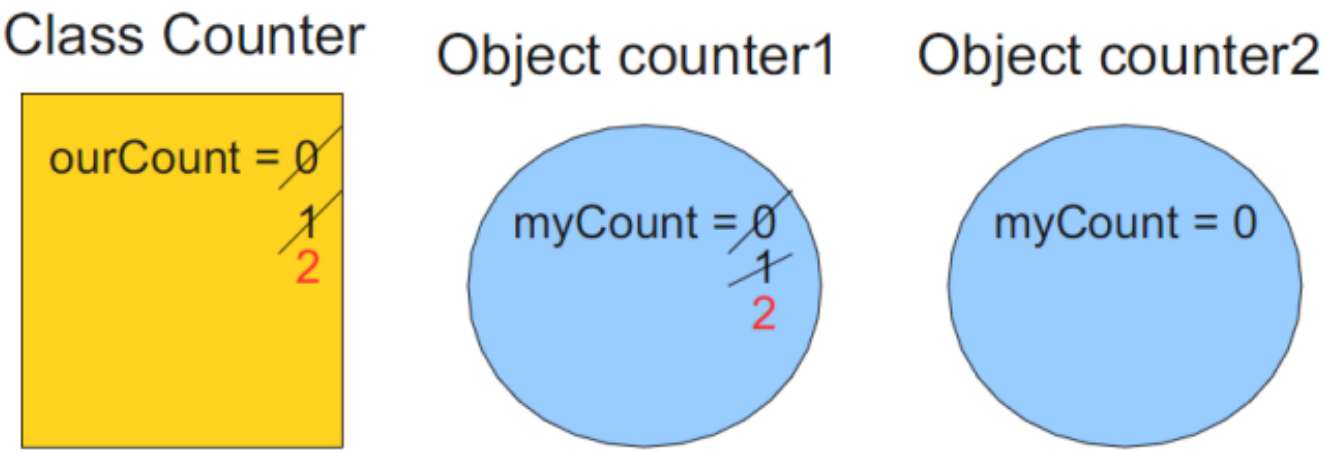

```
Counter counter1 = new Counter();
Counter counter2 = new Counter();
```

# Ôn tập (2)



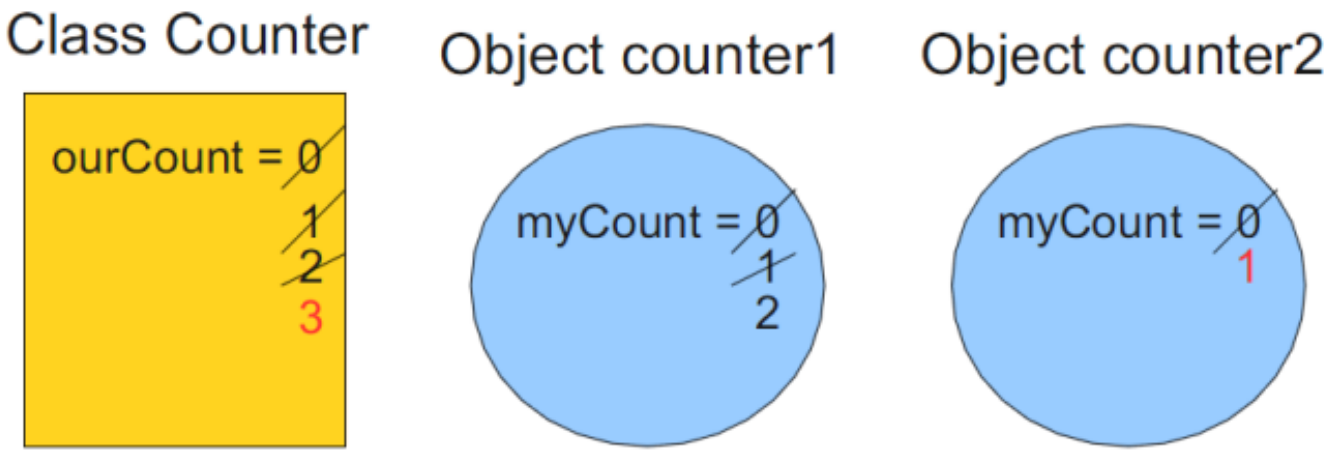

```
Counter counter1 = new Counter();
Counter counter2 = new Counter();
counter1.increment();
```

# Ôn tập (2)




```
Counter counter1 = new Counter();
Counter counter2 = new Counter();
counter1.increment();
counter1.increment();
```

# Ôn tập (2)




```
Counter counter1 = new Counter();
Counter counter2 = new Counter();
counter1.increment();
counter1.increment();
counter2.increment();
```

# Kiểm soát quyền truy cập (1)

```
    public class CreditCard{
        public String cardNumber;
        public double expenses;
        public void charge(double amount){
            expenses += amount;
        }
        public String getCardNumber(String password){
            if(password.equals("SECRET!3*!")){
            return cardNumber;
        }
        return "robber";
    }
}
```

# Kiểm soát quyền truy cập (2)

```
public class Malicious{
    public static void main(String args[]){
        maliciousMethod(new CreditCard());
    }
    public static void maliciousMethod(CreditCard card){
        card.expenses = 0;
        System.out.println(card.cardNumber);
    }
}
```

# Kiểm soát quyền truy cập (3)

## public và private

- **public**: Tất cả các lớp khác có thể truy cập trực tiếp tới thuộc tính và phương thức của đối tượng
- **private**: Chỉ có thể truy cập thuộc tính hay phương thức từ trong chính lớp đó

Kiểm soát truy cập áp dụng cho các **thuộc tính** và **phương thức**

# Kiểm soát quyền truy cập (4)

```
public class CreditCard{
    private String cardNumber;
    private double expenses;
    public void charge(double amount){
        expenses +=amount;
    }
    public String getCardNumber(String password){
        if(password.equals("SECRET!3*!")){
            return cardNumber;
        }
        return "robber";
    }
}
```

# Kiểm soát quyền truy cập (5)

## Tại sao phải kiểm soát quyền truy cập

- Bảo vệ các thông tin riêng
- Làm rõ cách sử dụng lớp
- Tách biệt cài đặt và giao diện lớp

# Phạm vi biến trong lớp

- Phạm vi biến trong phương thức (nhắc lại): từ vị trí khai báo cho tới dấu đóng ngoặc } tương ứng với dấu mở ngoặc { gần nhất xuất hiện trước nó
- Phạm vi biến ở mức lớp: trong toàn lớp
- Ví dụ:

```
public class Example{
    private int memberVariable;
    public void setVariable(int value){
        memberVariable = value;
    }
}
```

# Từ khoá this

- Tên biến trong phương thức có thể trùng với tên biến ở mức lớp. Khi đó từ khóa **this** dùng để phân biệt biến lớp với biến của phương thức
- Ví dụ:

```
public class Example{
    private int memberVariable;
    public void setVariable(int memberVariable){
        this.memberVariable = memberVariable;
    }
}
```

# Tổ chức gói

- Mục đích: tổ chức các lớp sao cho việc tra cứu, sử dụng được thuận tiện
  - Mỗi lớp thuộc vào một gói (package) nào đó
  - Các lớp trong cùng một gói phục vụ mục đích tương tự nhau
  - Cho phép phân biệt các lớp khác nhau nhưng trùng tên
- Các package đơn giản là các thư mục
- Trong chương trình nếu sử dụng lớp thuộc các gói khác (thư mục khác) thì cần phải khai báo (import)

# Định nghĩa gói

- Định nghĩa gói ở đầu tệp

```
package path.foo;
public class Foo{
// ..... do somethings
}
```

- Tệp Foo.java chứa lớp này phải nằm trong thư mục con path\foo của thư mục làm việc hiện thời
- Khai báo sử dụng một lớp cụ thể trong gói hay tất cả các lớp trong gói

```
import path.foo.Foo;
import path.foo.*;
```

# Tổ chức gói

```java
package parenttools;
public class BabyFood{
}
```

```java
package parenttools;
public class Baby{
}
```

```java
package adult;
import parenttools.Baby;
import parenttools.BabyFoods;
public class Parent{
    public static void main(String []args){
        Baby baby = new Baby();
        baby.feed(new BabyFood());
    }
}
```

# Lưu ý khi sử dụng gói

- Tất cả các lớp "nhìn thấy" các lớp trong cùng gói (cùng thư mục) không cần khai báo ("import") các lớp này khi sử dụng
- Tất cả các lớp "nhìn thấy" các lớp trong gói java.lang
  - java.lang.System
  - java.lang.String
  - ....
- Mỗi lớp chỉ sử dụng được các lớp trong gói khác nếu các lớp này đã được khai báo **public**

# Kiểm soát quyền truy cập

| | class | package | subclass | world |
|---|---|---|---|---|
| public | Y | Y | Y | Y |
| protected | Y | Y | Y | N |
| *no modifier* | Y | Y | N | N |
| private | Y | N | N | N |

# Kiểu dữ liệu ArrayList

- Mảng: kích thước cố định sau khi khởi tạo => bất tiện nếu số phần tử không được biết trước
- ArrayList: mảng có kich thước thay đổi
- Sử dụng ArrayList
  - Lấy kích thước mảng (phương thức size)
  - Thêm phần tử (phương thức add)
  - Đọc/thay đổi giá trị phần tử (phương thức get/set)
  - Xóa phần tử (phương thức remove)
  - Lặp trên các phần tử