



BỘ MÔN CÔNG NGHỆ PHẦN MỀM
VIỆN CNTT & TT
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI



IT3100

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Bài 03. Trừu tượng hóa và Đóng gói

Mục tiêu bài học

- Tìm hiểu bản chất, vai trò của trừu tượng hóa
 - Khái niệm, các góc nhìn, so sánh lớp và đối tượng
- Tìm hiểu về Đóng gói
 - Khái niệm đóng gói, che giấu dữ liệu
 - Chỉ định truy cập
 - Phương thức getter/setter
- Tìm hiểu cách xây dựng lớp, gói
 - Xây dựng lớp trong Java
 - Quản lý lớp với package
 - Biểu diễn đối tượng, lớp, gói trong UML

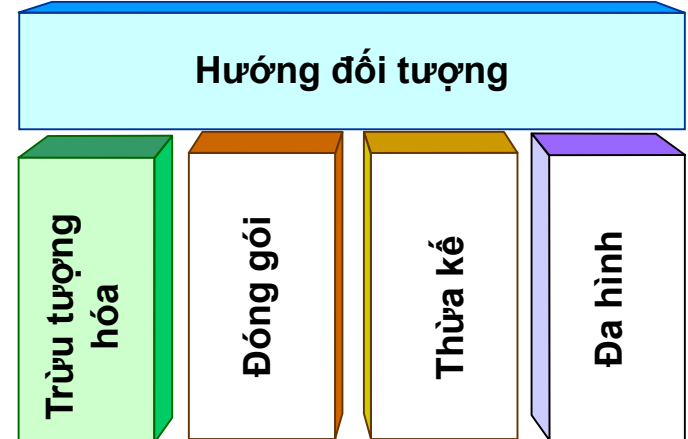
Nội dung

1. Trừu tượng hóa
2. Đóng gói
3. Xây dựng lớp và gói

1/ Trừu tượng hóa

1.1 Trừu tượng hóa

- Là một trong 4 nguyên lý cơ bản của lập trình HĐT.
- Là quá trình loại bỏ đi các thông tin ít quan trọng và giữ lại những thông tin quan trọng, có ý nghĩa với bài toán.
- 2 loại trừu tượng hóa
 - Trừu tượng hóa điều khiển
 - Trừu tượng hóa dữ liệu



1.1. Trừu tượng hóa (2)

- Trừu tượng hóa điều khiển:
 - Bằng cách sử dụng các chương trình con (subprogram) và các luồng điều khiển (control flow)
 - Ví dụ: $a := (1 + 2) * 5$
 - Nếu không có trừu tượng hóa điều khiển, LTV phải chỉ ra tất cả các thanh ghi, các bước tính toán mức nhị phân...
- Trừu tượng hóa dữ liệu:
 - Xử lý dữ liệu theo các cách khác nhau tùy bài toán

1.2. Trừu tượng hóa dữ liệu – Góc nhìn

- Trừu tượng hóa dữ liệu là một cách nhìn hoặc cách biểu diễn một thực thể chỉ bao gồm các thuộc tính liên quan trong một ngữ cảnh nào đó.
- Dựa vào các đặc điểm, thuộc tính đó để phân biệt các thực thể khác nhau trong ngữ cảnh đó.
- Góc nhìn khác nhau (bài toán khác nhau) thì đặc điểm, thuộc tính dùng để trừu tượng hóa sẽ khác nhau.

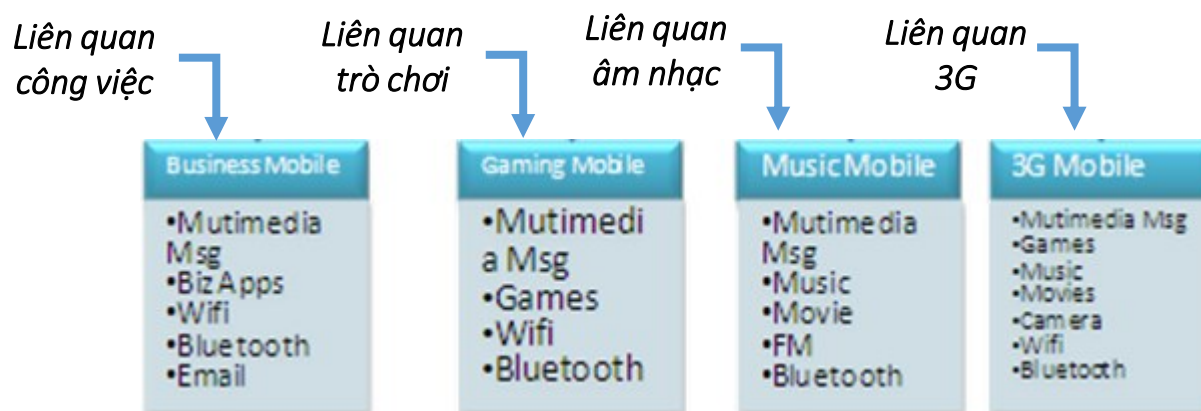
Ví dụ: Điện thoại Nokia



- Những thông tin về các "đối tượng" này?
 - Tất cả là điện thoại Nokia
 - Có loại nắp trượt, có loại nắp gập, có loại dạng bar
 - Một số điện thoại là dòng doanh nhân, một số dòng âm nhạc, 3G...
 - Bàn phím loại tiêu chuẩn, QWERTY hoặc không có bàn phím
 - Màu sắc, chất liệu, kích cỡ... khác nhau
 - v.v...
- Tùy bài toán, chỉ “trích rút” lấy những thông tin quan trọng, phù hợp

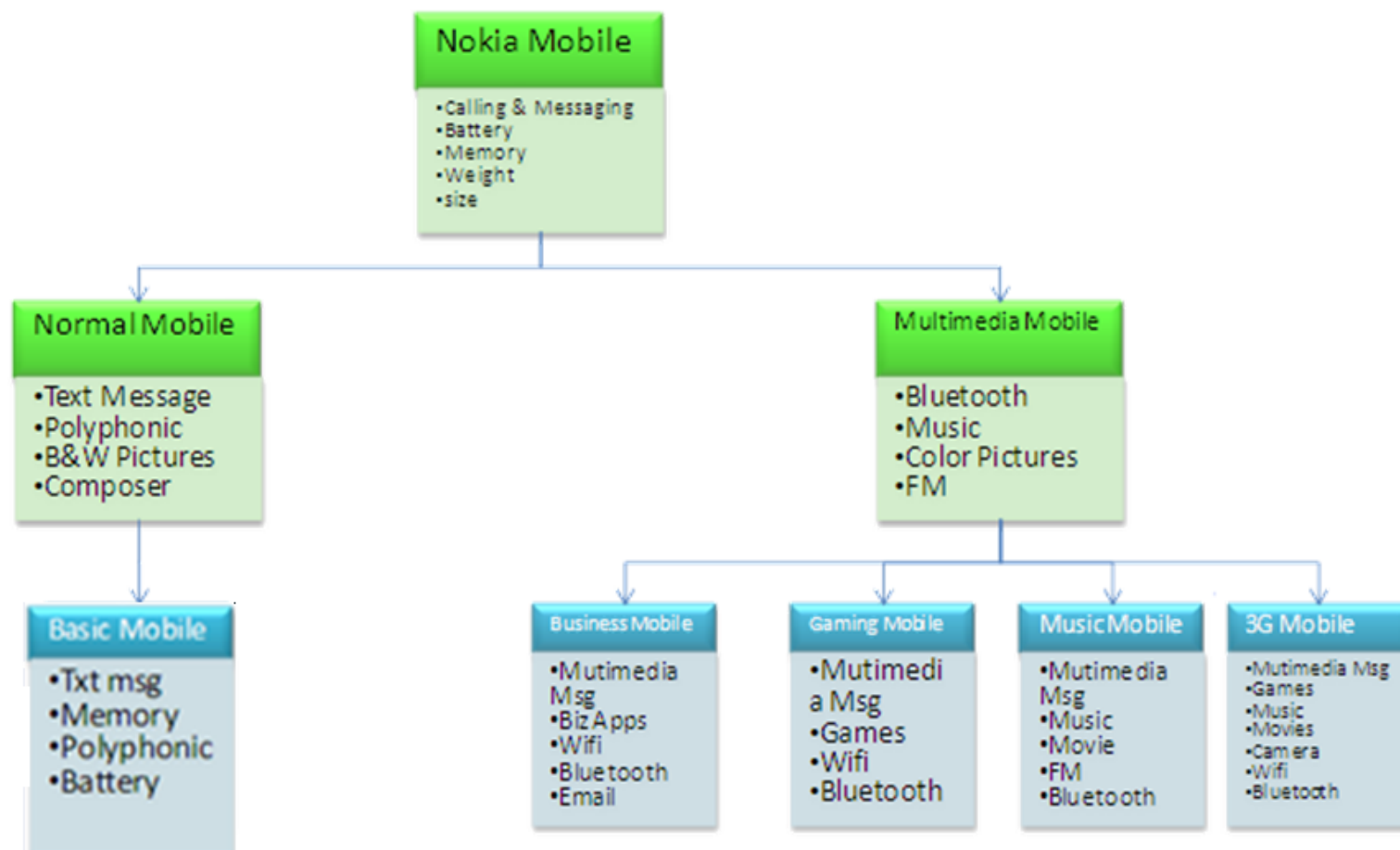
Ví dụ: Điện thoại Nokia (2)

- Các bài toán khác nhau, yêu cầu mô tả các tính chất khác nhau về chiếc điện thoại.



Ví dụ: Điện thoại Nokia (3)

- Có thể trừu tượng hóa nhiều mức.



1.3 Lớp

- Lớp (Class) là cách phân loại các đối tượng dựa trên đặc điểm chung của các đối tượng đó.
- Lớp chính là kết quả của quá trình *trừu tượng hóa dữ liệu*.
 - Lớp định nghĩa một *kiểu dữ liệu* mới, trừu tượng hóa một tập các đối tượng
 - Một đối tượng gọi là một *thể hiện* của lớp
- Lớp gồm các *phương thức* và *thuộc tính* chung của các đối tượng cùng một loại.

1.3.1 Thuộc tính

- Thuộc tính
 - Một thuộc tính của một lớp là một trạng thái chung được đặt tên của *tất cả* các thể hiện của lớp đó có thể có.
 - Ví dụ: Lớp Ô tô có các thuộc tính
 - Màu sắc
 - Vận tốc
- Các thuộc tính của cũng là các giá trị trừu tượng.
- Mỗi đối tượng có bản sao các thuộc tính của riêng nó
 - Ví dụ: một chiếc Ô tô đang đi có thể có màu đen, vận tốc 60 km/h

1.3.2 Phương thức

- Phương thức:
 - Xác định các hoạt động chung mà *tất cả* các thể hiện của lớp có thể thực hiện được.
 - Xác định cách một đối tượng đáp ứng lại một thông điệp
- Thông thường các phương thức sẽ hoạt động trên các thuộc tính và thường làm thay đổi các trạng thái của đối tượng.
 - Bất kỳ phương thức nào cũng phải thuộc về một lớp nào đó.
 - Ví dụ: Lớp Ô tô có các phương thức
 - Tăng tốc
 - Giảm tốc

1.4. Lớp vs. Đối tượng

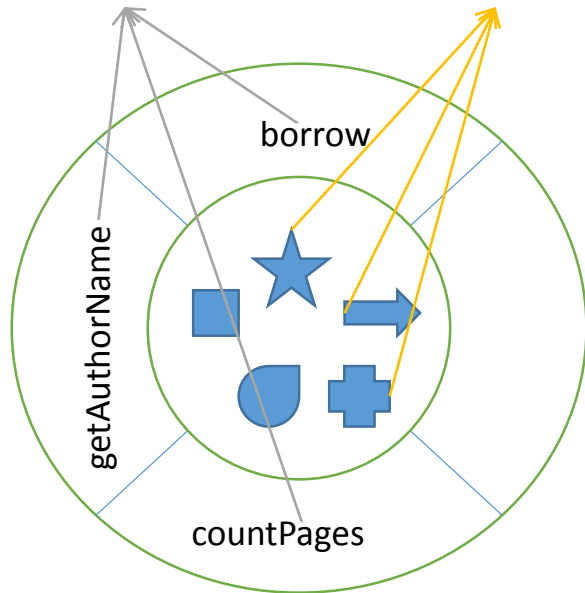
Lớp	Đối tượng
Lớp là mô hình khái niệm, mô tả các thực thể	Đối tượng là sự vật thật, là thực thể thực sự
Lớp như một bản mẫu, định nghĩa các <i>thuộc tính</i> và <i>phương thức</i> chung của các đối tượng	Mỗi đối tượng có một lớp xác định dữ liệu (<i>thuộc tính</i>) và hành vi (<i>phương thức</i>) của nó. Dữ liệu của các đối tượng khác nhau là khác nhau
Một lớp là sự trừu tượng hóa của một tập các đối tượng	Đối tượng là một thể hiện (instance) của một lớp

1.4. Lớp vs. Đối tượng (2)

Lớp BOOK

Phương thức: các hành vi đối tượng có thể thực hiện

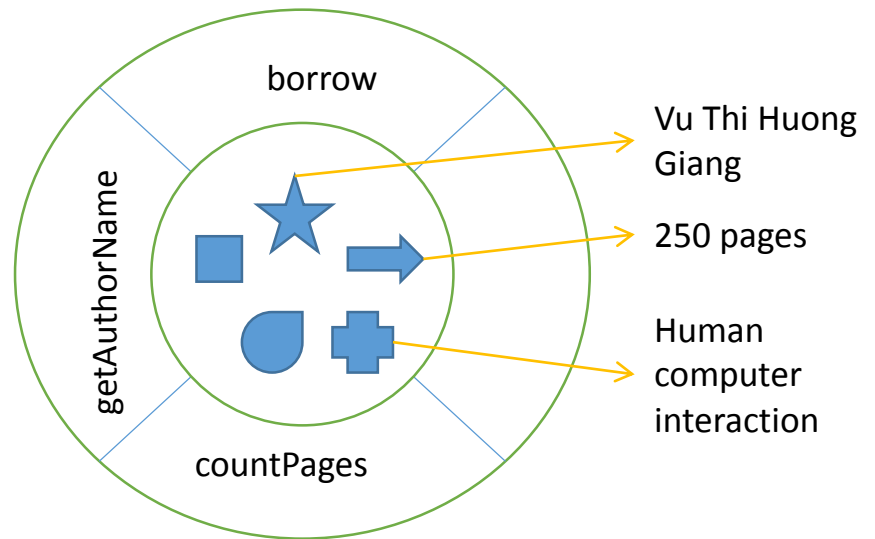
Thuộc tính: các thông tin liên quan đến trạng thái



Đối tượng MyBook

Thể hiện: Một đối tượng cụ thể

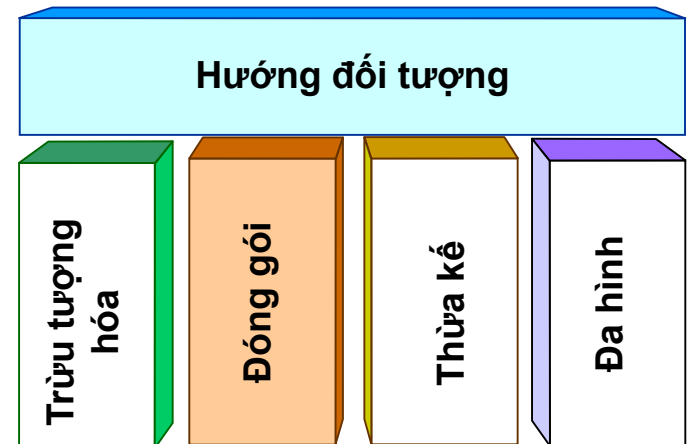
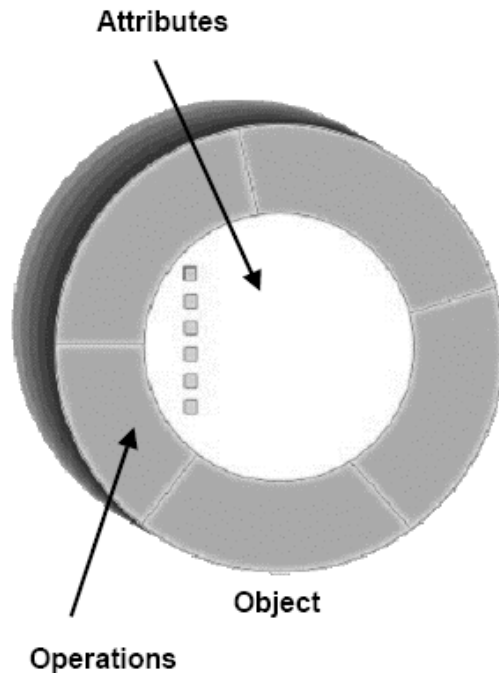
Thuộc tính thể hiện: gán giá trị cho các thuộc tính của một đối tượng cụ thể



2/ Đóng gói

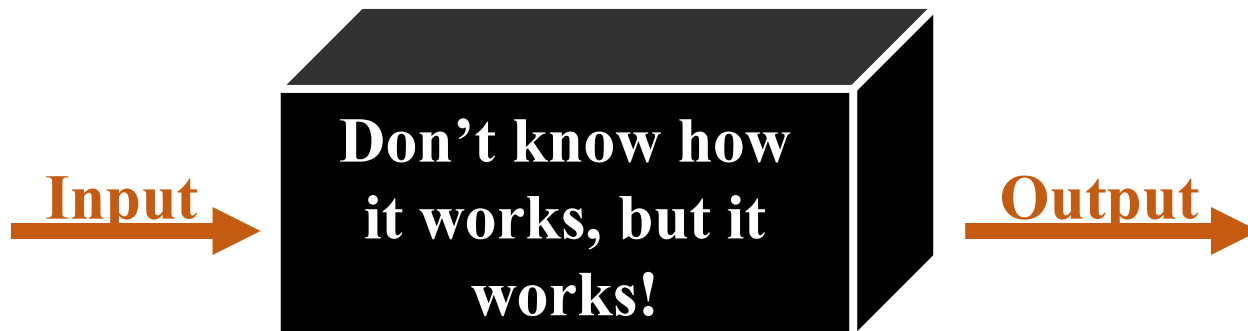
2.1. Đóng gói (Encapsulation)

- Là một trong 4 nguyên lý cơ bản của lập trình HĐT.
- Dữ liệu/thuộc tính và hành vi/phương thức được đóng gói trong một lớp.



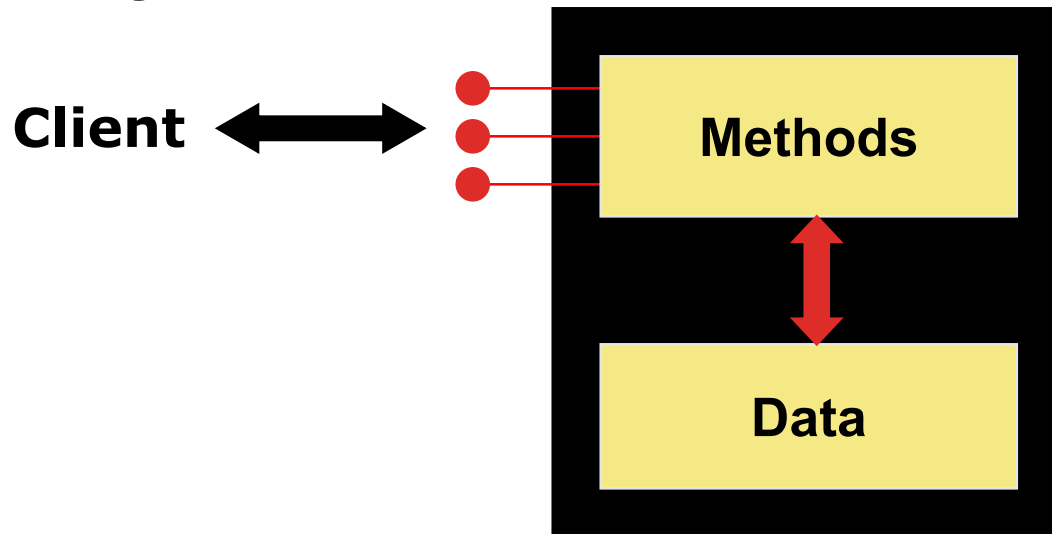
2.1. Đóng gói (2)

- Một đối tượng là một thực thể được đóng gói với mục đích:
 - Cung cấp tập các dịch vụ nhất định
 - Đối tượng được đóng gói có thể được xem như một hộp đen – các công việc bên trong là ẩn so với client
 - Dù thay đổi thiết kế/mã nguồn bên trong nhưng giao diện bên ngoài không bị thay đổi theo



2.1. Đóng gói (3)

- Sau khi đóng gói, một đối tượng có hai khung nhìn:
 - Bên trong: Chi tiết về các thuộc tính và các phương thức của lớp tương ứng với đối tượng
 - Bên ngoài: Các dịch vụ mà một đối tượng có thể cung cấp và cách đối tượng đó tương tác với phần còn lại của hệ thống

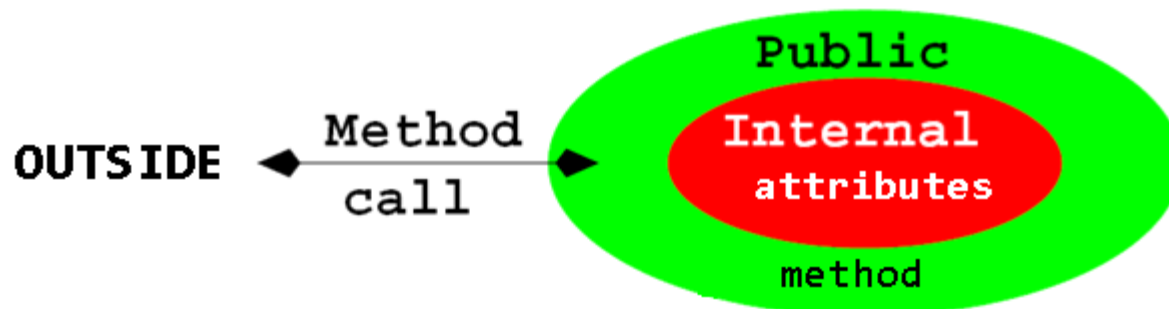


2.2 Phạm vi truy cập

- Phạm vi truy cập xác định khả năng nhìn thấy được của một thành phần của chương trình với các thành phần khác của chương trình
- Đối với lớp
 - Phạm vi truy cập có thể được áp dụng cho các thành phần của lớp
 - **private**: chỉ truy cập được từ bên trong lớp đó
 - **public**: có thể truy cập được tại mọi nơi, từ trong và ngoài lớp
 - Xem thêm phần 3.1.4

2.3 Che giấu dữ liệu

- Sử dụng phạm vi truy cập để che giấu dữ liệu: tránh thay đổi trái phép hoặc làm sai lệch dữ liệu
- Dữ liệu được che giấu ở bên trong lớp bằng cách gán phạm vi truy cập *private*.
 - chỉ có thể truy cập từ các phương thức bên trong lớp
- Các đối tượng khác muốn truy nhập vào dữ liệu riêng tư này phải thông qua các phương thức của lớp có phạm vi truy cập *public*.



2.3 Che giấu dữ liệu (2)

- Để truy cập và chỉnh sửa các giá trị của dữ liệu, lớp cần phải cung cấp các dịch vụ
 - Accessor (getter): Trả về giá trị hiện tại của một thuộc tính (dữ liệu)
 - Mutator (setter): Thay đổi giá trị của một thuộc tính
 - Thường là getX và setX, trong đó X là tên thuộc tính

2.4 Phương thức Get

- Các phương thức truy vấn Get là các phương thức dùng để hỏi giá trị của các thành viên dữ liệu của một đối tượng
- Có nhiều loại câu hỏi truy vấn có thể:
 - truy vấn đơn giản (“giá trị của x là bao nhiêu?”)
 - truy vấn điều kiện (“thành viên x có lớn hơn 10 không?”)
 - truy vấn dẫn xuất (“tổng giá trị của các thành viên x và y là bao nhiêu?”)
- Đặc điểm quan trọng của phương thức truy vấn là nó không nên thay đổi trạng thái hiện tại của đối tượng
 - không thay đổi giá trị của thành viên dữ liệu nào.

2.5 Phương thức Set

- Các phương thức thiết lập Set là các phương thức dùng để thay đổi giá trị các thành viên dữ liệu
- Ưu điểm của việc sử dụng các phương thức setter là kiểm soát tính hợp lệ của các thành phần dữ liệu
 - Kiểm tra giá trị đầu vào trước khi gán vào các thuộc tính

Ví dụ: phương thức get, set

```
class Student{  
    private String name;  
    public String getName() {  
        return this.name;  
    }  
    public void setName(String name)  
    {  
        this.name = name;  
    }  
}
```

Ví dụ: phương thức get, set (2)

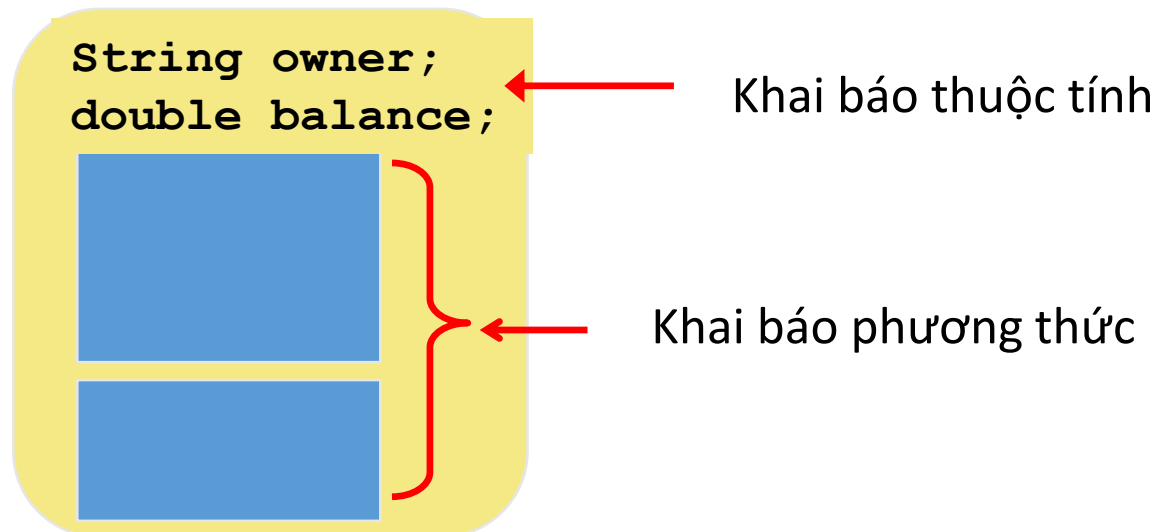
```
class Student{
    private String name;
    public String getName() {
        return this.name;
    }
    public void setName(String name)
    {
        this.name = name;
    }
}
```

```
class Manager{
    private Student[] students;
    public initialize()
    {
        students = new Student[10];
        students[0] = new Student();
        //students[0].name = "Hung"; error
        students[0].setName("Hung");
    }
}
```

3/ Xây dựng lớp và gói

3.1 Xây dựng lớp

- Thông tin cần thiết để định nghĩa một lớp
 - *Tên (Name)*
 - Tên lớp nên mô tả đối tượng trong thế giới thật
 - Tên lớp nên là số ít, ngắn gọn, và xác định rõ ràng cho sự trừu tượng hóa.
 - Danh sách các thuộc tính
 - Danh sách các phương thức



3.1.1 Khai báo lớp

- Cú pháp: sử dụng từ khóa `class`

```
class Ten_Lop {  
    // Nội dung lớp  
    // Khai báo các thuộc tính  
    // Khai báo và cài đặt các phương thức  
}
```

Ví dụ

```
class Student {  
    // Nội dung lớp  
}
```

3.1.2 Khai báo Thuộc tính

- Cú pháp khai báo thuộc tính: Tương tự khai báo biến

`phamViTruyCap kieu tenThuocTinh;`

- Thuộc tính có thể được khởi tạo khi khai báo
 - Các giá trị mặc định sẽ được sử dụng nếu không được khởi tạo.
- Ví dụ

The diagram shows a Java code snippet for a class `BankAccount` with two private attributes. Annotations with arrows point to specific parts of the code:

- access modifier**: Points to the `private` keyword in the first attribute declaration.
- type**: Points to the `String` data type in the first attribute declaration.
- name**: Points to the `owner` variable name in the first attribute declaration.

```
package com.megabank.models;

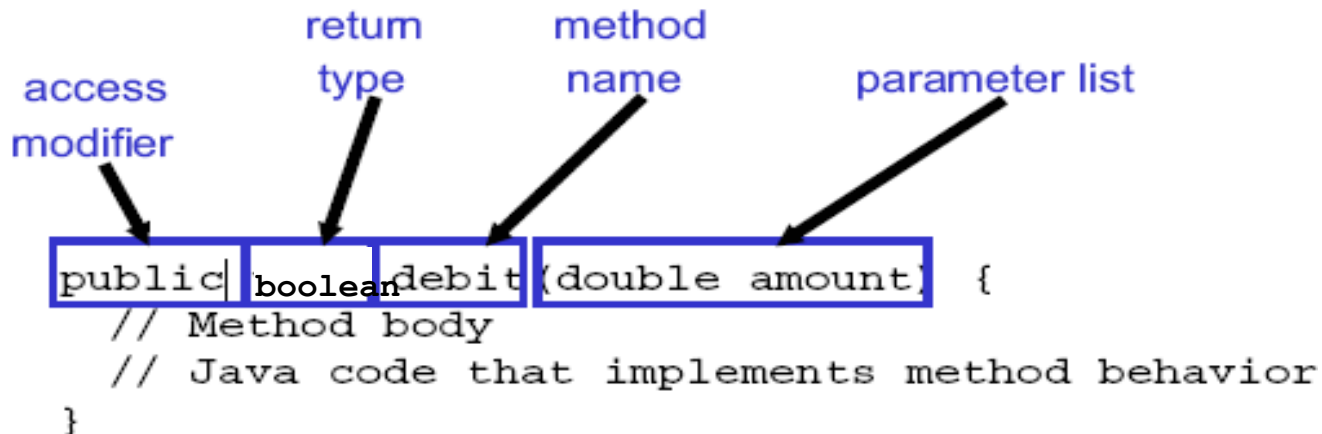
public class BankAccount {
    private String owner;
    private double balance = 0.0;
}
```

3.1.3 Khai báo Phương thức

- Khai báo: tương tự khai báo hàm
- Cú pháp

```
phamViTruyCap kiểuTrảVề tênPhươngThức (ds tham số) {  
    // Nội dung phương thức  
}
```

- Ví dụ

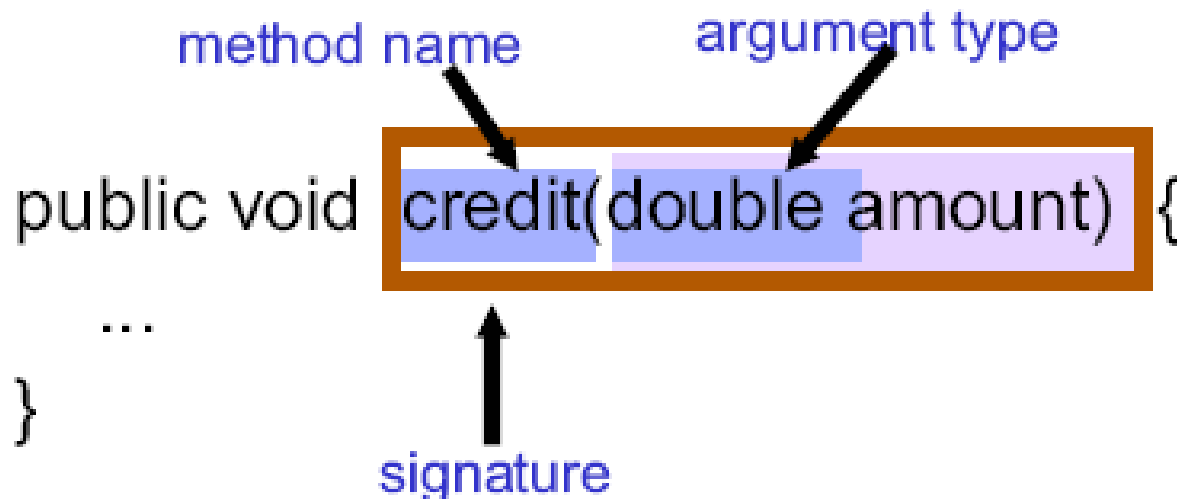


The diagram shows a Java method declaration: `public boolean debit(double amount) {`. Four labels with arrows point to specific parts of the declaration: 'access modifier' points to 'public', 'return type' points to 'boolean', 'method name' points to 'debit', and 'parameter list' points to '(double amount)'. The rest of the code block is shown below the opening brace.

```
public boolean debit(double amount) {  
    // Method body  
    // Java code that implements method behavior  
}
```

3.1.3 Khai báo Phương thức (2)

- Mỗi phương thức phải có một chữ ký riêng, phân biệt các phương thức, gồm:
 - Tên phương thức
 - Số lượng các tham số và kiểu của chúng



The diagram illustrates the components of a Java method signature. It shows a code snippet: `public void credit(double amount) {`. The text is enclosed in a brown rectangular box. Three blue labels with arrows point to specific parts of the signature: 'method name' points to 'credit', 'argument type' points to 'double', and 'signature' points to the entire 'credit(double amount)' portion. The 'public void' part is to the left of the box, and '...' and '}' are below it.

```
public void credit(double amount) {  
    ...  
}
```


3.1.3 Khai báo Phương thức (3)

- Khi phương thức trả về ít nhất một giá trị hoặc một đối tượng thì bắt buộc phải có câu lệnh `return` để trả điều khiển cho đối tượng gọi phương thức.
- Nếu phương thức không trả về 1 giá trị nào (`void`) không cần câu lệnh `return`
- Có thể có nhiều lệnh `return` trong một phương thức; câu lệnh đầu tiên mà chương trình gặp sẽ được thực thi.

Ví dụ - Khai báo phương thức

```
Public boolean checkOdd(int i)
{
    if (i %2 ==0)
        return true;
    else
        return false;
}
```

```
Public boolean checkOdd(int i)
{
    return true;
    return false; //error
}
```

Ví dụ - Khai báo lớp

```
class BankAccount {  
    private String owner;  
    private double balance;  
    public boolean debit(double amount) {  
        if (amount > balance)  
            return false;  
        else {  
            balance -= amount;  
            return true;  
        }  
    }  
    public void credit(double amount) {  
        balance += amount;  
    }  
}
```

3.1.4 Phạm vi truy cập thuộc tính/phương thức

- Phạm vi truy cập thuộc tính/phương thức
 - **public**: Thuộc tính hoặc phương thức có thể được truy cập từ bất cứ đâu, kể cả bên ngoài lớp, ngoài gói chứa lớp đó.
 - **không có chỉ định**: Thuộc tính hoặc phương thức chỉ có thể được truy cập từ bên trong package chứa lớp đó.
 - **private**: Thuộc tính hoặc phương thức chỉ có thể được truy cập trong phạm vi lớp đó
 - **protected**: Thuộc tính hoặc phương thức chỉ có thể được truy cập trong phạm vi lớp đó và từ lớp con kế thừa của lớp đó.

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
<i>no modifier</i>	Y	Y	N	N
private	Y	N	N	N

3.2. Xây dựng gói

- Các lớp được nhóm lại thành package
 - Package bao gồm một tập hợp các lớp có quan hệ logic với nhau
- Gói (package) giống như thư mục giúp:
 - Tổ chức và xác định vị trí lớp dễ dàng và sử dụng các lớp một cách phù hợp.
 - Tránh cho việc đặt tên lớp bị xung đột (trùng tên)
 - Các package khác nhau có thể chứa các lớp có cùng tên
 - Bảo vệ các lớp, dữ liệu và phương thức ở mức rộng hơn so với mối quan hệ giữa các lớp.
- Còn được gọi là *không gian tên (namespace)* trong một số ngôn ngữ lập trình (C++...)

3.2. Xây dựng gói

- Một số package có sẵn của Java: `java.lang`, `javax.swing`, `java.io`...
- Package có thể do ta tự đặt
 - Cách nhau bằng dấu “.”
 - Quy ước sử dụng ký tự thường để đặt tên package
 - Tên gói phải được viết trên cùng của file mã nguồn
- Chỉ được phép có 1 câu khai báo gói trong mỗi file mã nguồn, và khai báo này sẽ được áp dụng cho tất cả các dữ liệu trong file đó.
- Một gói có thể được đặt trong một gói khác
 - Phân cách bằng dấu .
 - Ví dụ `package trungtt.oop.k59;`

3.2.1. Khai báo gói

- Cú pháp khai báo:

```
package tenpackage;  
chi_dinh_truy_cap class TenLop {  
    // Than lop  
}
```

- **chi_dinh_truy_cap:**

- public: Lớp có thể được truy cập từ bất cứ đâu, kể cả bên ngoài package chứa lớp đó.
- Không chỉ định: Lớp chỉ có thể được truy cập từ bên trong package chứa lớp đó.

```
package oop.k52.cnpm;  
    public class Student {  
        ...  
    }
```

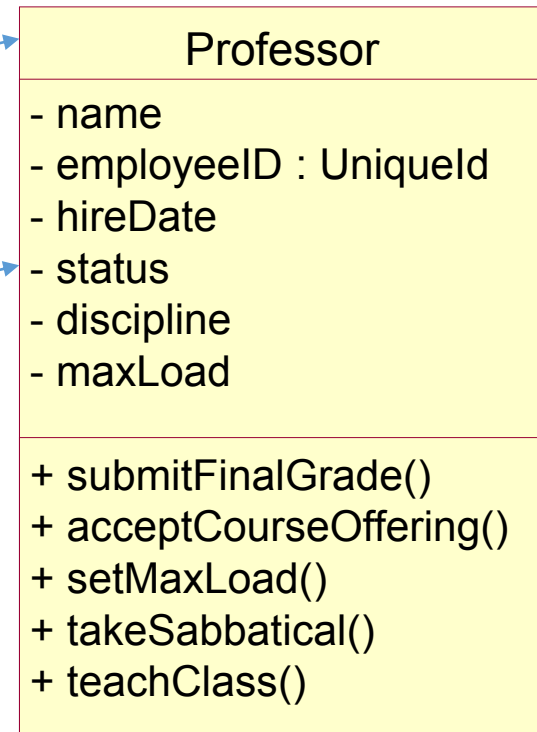
3.3 Biểu diễn trong UML

- Lớp (class) được biểu diễn bằng 1 hình chữ nhật với 3 thành phần:

- Tên lớp

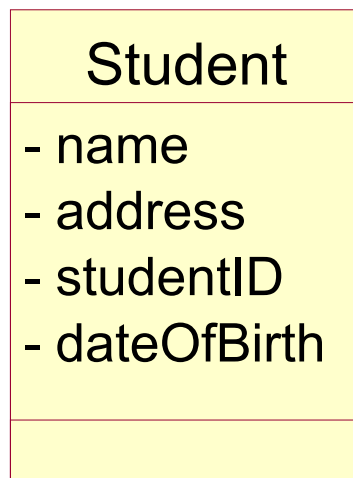
- Thuộc tính

- Phương thức

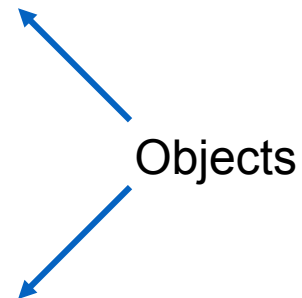
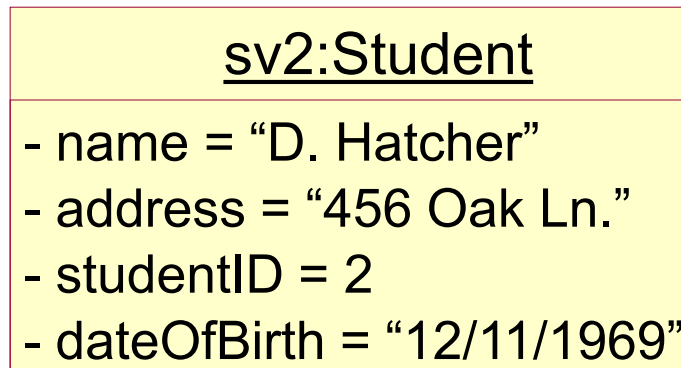
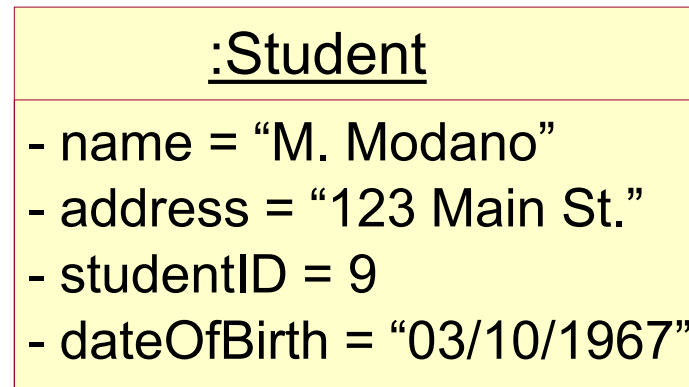


3.3 Biểu diễn trong UML (2)

- Đối tượng: biểu diễn bằng *tên đối tượng:tên lớp*, và các giá trị của thuộc tính.



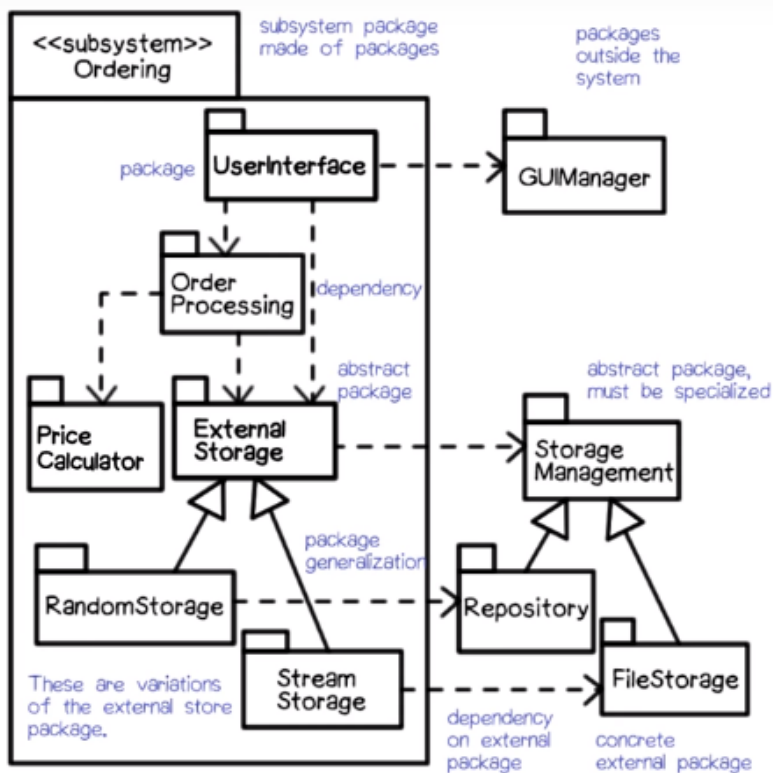
Class



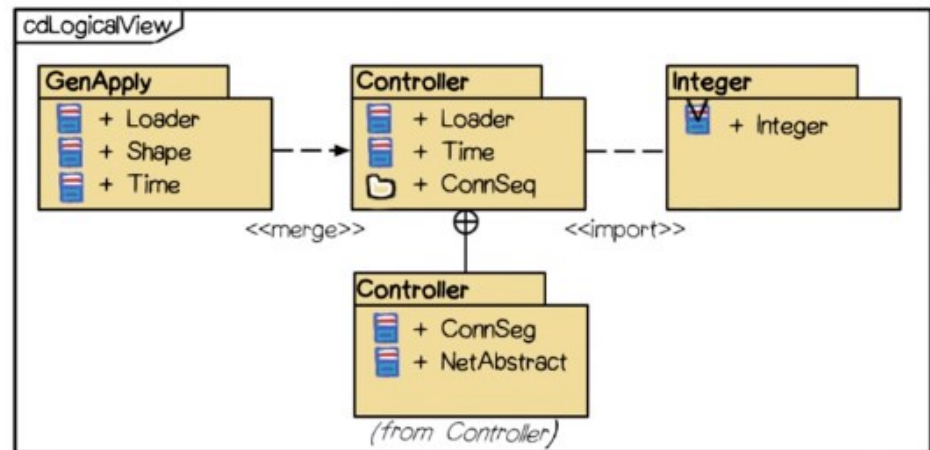
3.3 Biểu diễn trong UML (3)

- Biểu diễn gói trong UML

UML 1.5



UML 2.0



Bài tập

- **Bài 1**: Viết khai báo một gói chứa hai lớp: lớp hình vuông, lớp hình tròn. Viết khai báo lớp hình vuông, lớp hình tròn cùng các thuộc tính thích hợp, các phương thức get/set thích hợp.
- **Bài 2**: Viết khai báo một lớp Vector gồm 3 thành phần với những phương thức cộng/trừ vector, nhân với 1 hằng số, nhân vô hướng 2 vector.

Bài tập

- **Bài 3**. Viết mã nguồn cho lớp NhanVien như trong hình bên biết:

- Lương = Lương cơ bản * Hệ số lương
- Phương thức inTTin() hiển thị thông tin của đối tượng NhanVien tương ứng.

NhanVien
-tenNhanVien: String
-luongCoBan: double
-heSoLuong: double
+LUONG_MAX: double
+tangLuong(double): boolean
+tinhLuong(): double
+inTTin()

- Phương thức tangLuong(double) tăng hệ số lương hiện tại lên một lượng bằng giá trị tham số double truyền vào. Nếu điều này làm cho lương của nhân viên > lương tối đa cho phép thì không cho phép thay đổi, in ra thông báo và trả về false, ngược lại trả về true.
- Viết các phương thức get và set cho các thuộc tính của lớp NhanVien

