



Bộ môn Công nghệ phần mềm
VIỆN CÔNG NGHỆ THÔNG TIN TRUYỀN THÔNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI



LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

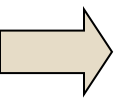
Chương 1. Tổng quan về OOP

Cao Tuấn Dũng
dungct@soict.hut.edu.vn

Nội dung

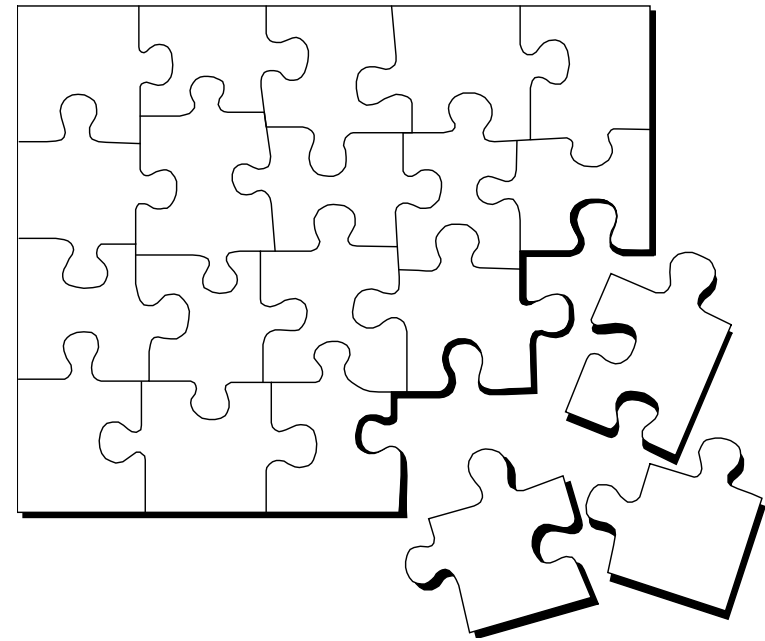
1. Công nghệ hướng đối tượng
2. Đối tượng và lớp
3. Các nguyên lý cơ bản của OO
4. Phân tích thiết kế HĐT
5. Ngôn ngữ lập trình Java/C++
6. Ví dụ và bài tập

Nội dung

- 
1. Công nghệ hướng đối tượng
 2. Đối tượng và lớp
 3. Các nguyên lý cơ bản
 4. Phân tích thiết kế HĐT
 5. Ngôn ngữ lập trình Java
 6. Ví dụ và bài tập

1.1 Công nghệ đối tượng

- Công nghệ đối tượng là một tập các quy tắc (trừu tượng hóa, đóng gói, đa hình), các hướng dẫn để xây dựng phần mềm, cùng với ngôn ngữ, cơ sở dữ liệu và các công cụ khác hỗ trợ các quy tắc này.



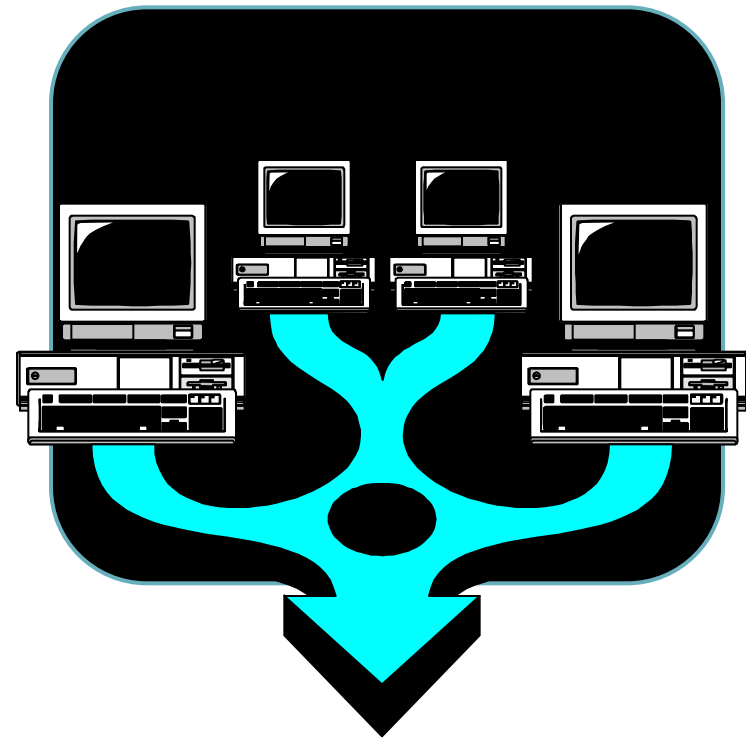
(Object Technology - A Manager's Guide, Taylor, 1997)

Công nghệ đối tượng

- Tạo ra các mô hình phản ánh một lĩnh vực nào đó sử dụng thuật ngữ của lĩnh vực đó.
- Các mô hình được tạo ra cần dễ tạo, dễ thay đổi, mở rộng, thẩm định và kiểm chứng
- Các hệ thống được xây dựng linh hoạt trong thay đổi, có các kiến trúc xác định và có cơ hội để tạo ra và thực thi các thành phần có khả năng tái sử dụng.
- Quy trình phát triển + Ngôn ngữ mô hình hóa + Kỹ thuật công cụ phát triển.

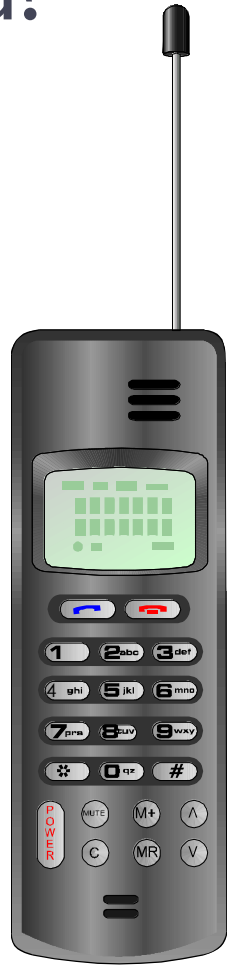
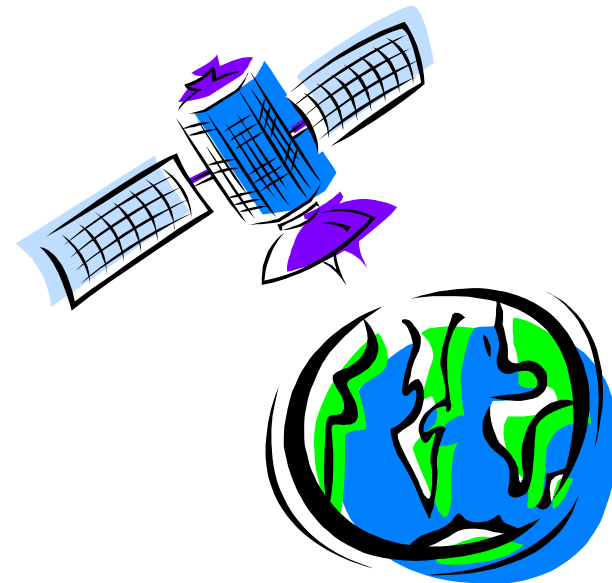
1.2 Công nghệ đối tượng được sử dụng ở đâu?

- Các hệ thống Client/Server và phát triển Web
 - Công nghệ đối tượng cho phép các công ty đóng gói thông tin doanh nghiệp trong các đối tượng và giúp phân phối quá trình xử lý qua mạng Internet hoặc một mạng máy tính.



Công nghệ đối tượng được sử dụng ở đâu?

- Hệ nhúng (embedded system)
- Hệ thống thời gian thực (real-time)
 - Công nghệ đối tượng cho phép các hệ thống thời gian thực có thể phát triển với chất lượng cao hơn và linh hoạt hơn
 - Hệ thống vệ tinh
 - Các hệ thống quốc phòng và hàng không vũ trụ
 - ...



Lịch sử phát triển

- Các mốc chính của công nghệ đối tượng

Simula



1967

C ++



Late 1980s

The UML



1996

1972



Smalltalk

1991



Java

2004



UML 2

1.3 Lịch sử phát triển của các NNLT

- ***Chính là sự phát triển của quá trình trừu tượng hóa***

**Assembly
code**

```

;CLEAR SCREEN USING BIOS
CLR: MOV AX,0600H      ;SCROLL SCREEN
      MOV BH,30         ;COLOUR
      MOV CX,0000       ;FROM
      MOV DX,184FH      ;TO 24,79
      INT 10H           ;CALL BIOS;
;INPUTTING OF A STRING
KEY:  MOV AH,0AH        ;INPUT REQUEST
      LEA DX,BUFFER     ;POINT TO BUFFER WHERE STRING STORED
      INT 21H           ;CALL DOS
      RET              ;RETURN FROM SUBROUTINE TO MAIN PROGRAM;
; DISPLAY STRING TO SCREEN
SCR:  MOV AH,09         ;DISPLAY REQUEST
      LEA DX,STRING     ;POINT TO STRING
      INT 21H           ;CALL DOS
      RET              ;RETURN FROM THIS SUBROUTINE;

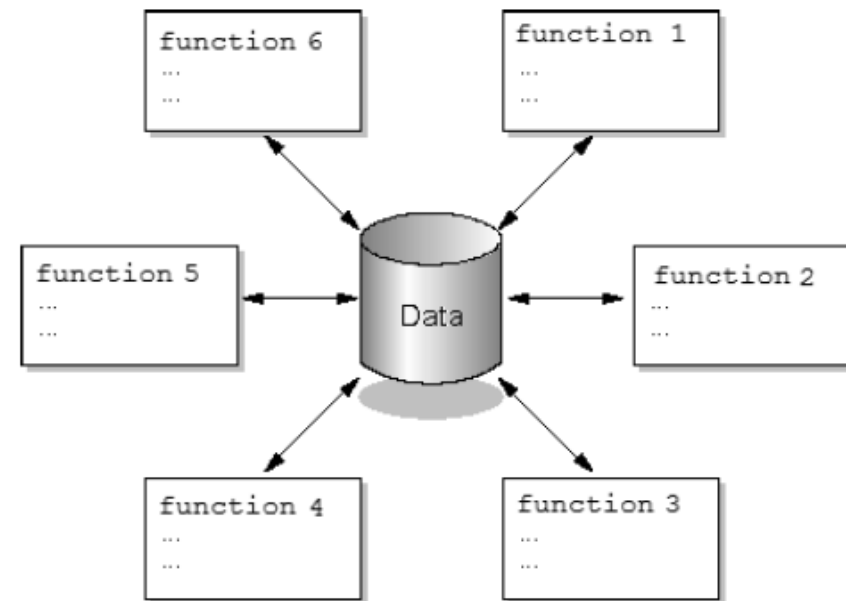
```

- a. Hợp ngữ (Assembly language):
 - Là một ngôn ngữ lập trình tuần tự, gần với tập các lệnh mã máy của CPU.
 - Khó nhớ, khó viết, nhất là với những bài toán phức tạp.
 - Khó sửa lỗi, bảo trì.

Lịch sử phát triển của các NNLT

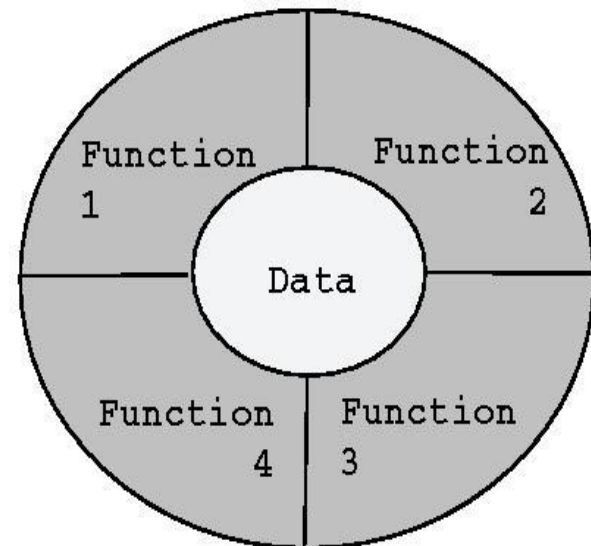
- b. NNLT cấu trúc/thủ tục:

- Xây dựng chương trình dựa trên các hàm/thủ tục/chương trình con
- Dữ liệu và xử lý (hàm) tách rời nhau
- Các hàm không bắt buộc phải tuân theo một cách thức chung truy cập vào dữ liệu



Lịch sử phát triển của các NNLT

- c. NNLT hướng đối tượng:
 - Thể hiện các thành phần của bài toán là các “đối tượng” (object).
 - Hướng đối tượng là một kỹ thuật để mô hình hóa hệ thống thành nhiều đối tượng.



Lịch sử phát triển của các NNLT

- ***Chính là sự phát triển của quá trình trừu tượng hóa***
 - Assembly : Trừu tượng hóa kiểu dữ liệu/chỉ thị cơ bản
 - Ngôn ngữ cấu trúc: Trừu tượng hóa điều khiển (control abstraction) + trừu tượng hóa chức năng (functional abstraction)
 - Ngôn ngữ HĐT: Trừu tượng hóa dữ liệu (Data abstraction)

Bài tập Đọc hiểu

- Đọc và tóm tắt một số điểm khác nhau giữa lập trình cấu trúc (hướng thủ tục) và OOP
- <http://www.desy.de/gna/html/cc/Tutorial/node3.htm>

Nội dung

1. Công nghệ hướng đối tượng
- 2. Đối tượng và lớp
3. Các nguyên lý cơ bản
4. Phân tích thiết kế HĐT
5. Ngôn ngữ lập trình Java/C++
6. Ví dụ và bài tập

Tư tưởng của Alan Kay

1. Tất cả đều là đối tượng.
2. Chương trình phần mềm có thể coi là một tập hợp các đối tượng tương tác với nhau
3. Mỗi đối tượng trong chương trình có các dữ liệu độc lập của mình và chiếm bộ nhớ riêng của mình.
4. Mỗi đối tượng đều có dạng đặc trưng của lớp các đối tượng đó.
5. Tất cả các đối tượng thuộc về cùng một lớp đều có các hành vi giống nhau



Alan Kay

2.1 Đối tượng (object)

- **Đối tượng** là chìa khóa để hiểu được kỹ thuật hướng đối tượng
- Trong hệ thống hướng đối tượng, mọi thứ đều là đối tượng



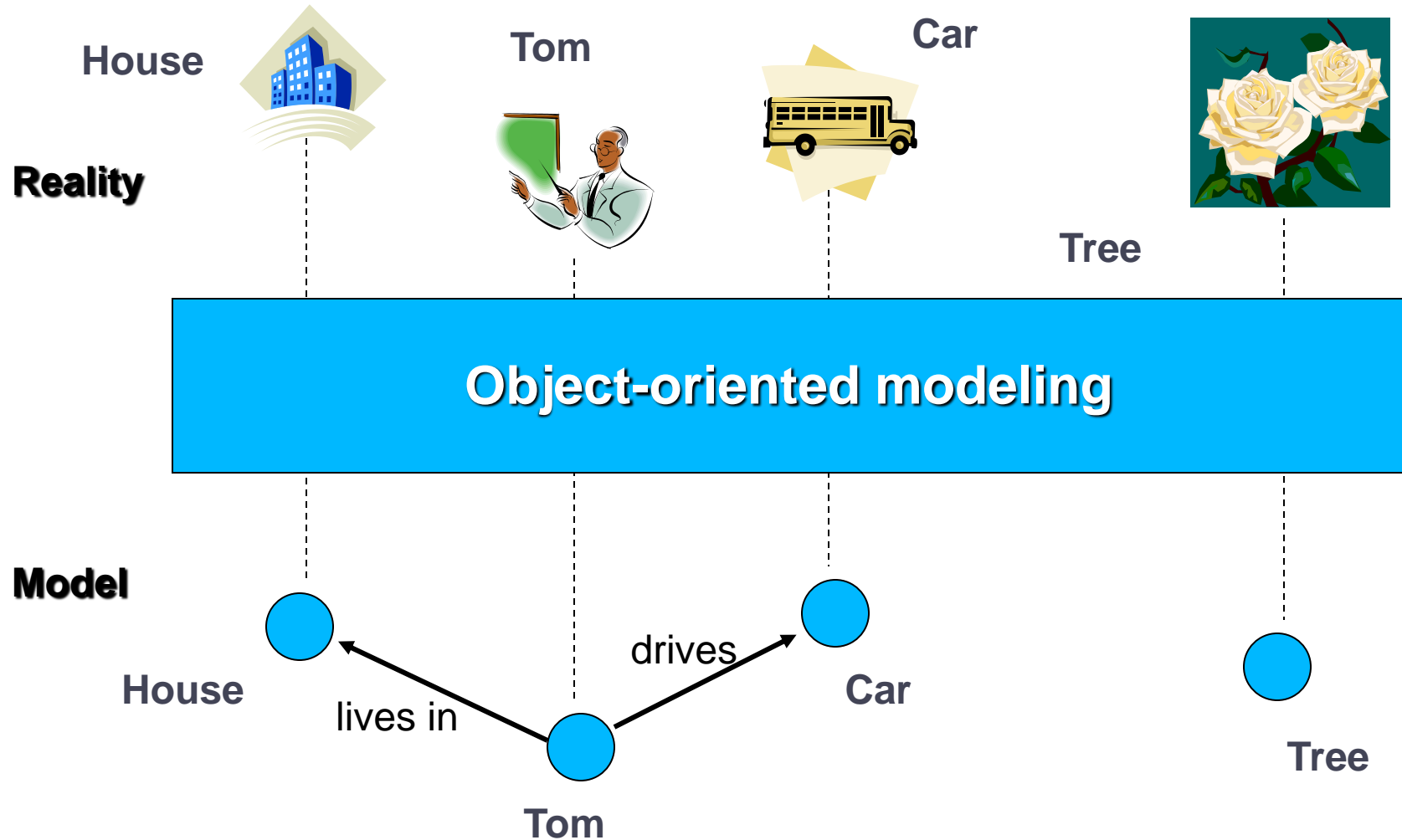
Viết một chương trình hướng đối tượng nghĩa là đang xây dựng một mô hình của một vài bộ phận trong thế giới thực

2.1 Đối tượng là gì?

- Đối tượng trong thế giới thực
 - Ví dụ một chiếc ô tô
- Liên quan đến chiếc ô tô:
 - Các thông tin về chiếc xe như: màu sắc, tốc độ, số km đã đi được,...
 - Các hoạt động của chiếc xe như: tăng tốc khi nhấn ga, giảm tốc khi đạp phanh,...




Đối tượng là gì?



Đối Tượng Thế Giới Thực (Real Object)

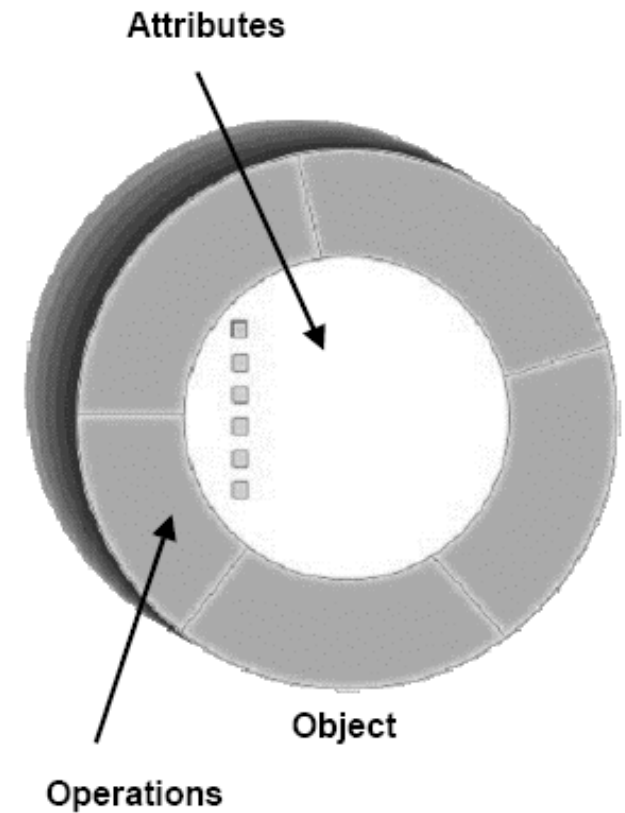
- Một **đối tượng thế giới thực** là một thực thể cụ thể mà thông thường chúng ta có thể *sờ, nhìn thấy* hay *cảm nhận* được.

- Tất cả có trạng thái (state) và hành động (behaviour)

	Trạng thái	Hành động	
Con chó	Tên Màu Giống Vui sướng	Sủa Vẫy tai Chạy Ăn	
Xe đạp	Bánh răng Bàn đạp Dây xích Bánh xe	Tăng tốc Giảm tốc Chuyển bánh răng ...	

Đối tượng là gì?

- Là một thực thể được đóng gói thành trạng thái (*state*) và hành vi (*behavior*).
 - **Trạng thái** được biểu diễn bởi các thuộc tính (attributes) và các mối quan hệ (relationships).
 - **Hành vi** được biểu diễn bởi các thao tác (operations), phương thức (methods).



Trạng thái



Dave
Age: 32
Height: 6' 2"



Brett
Age: 35
Height: 5' 10"



Gary
Age: 61
Height: 5' 8"

Hành vi



Get the mail.
Cook dinner.



Định danh

Okay, which one of you wise guys is the *real* Poppini?

I am the great Poppini!

I'm the great Poppini!

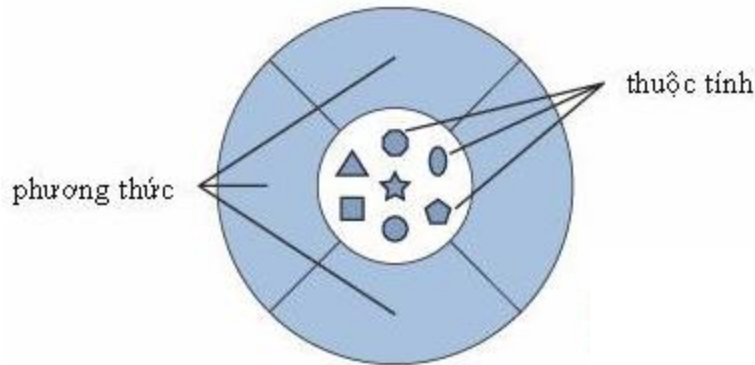
I am the great Poppini.

No, I'm the great Poppini.

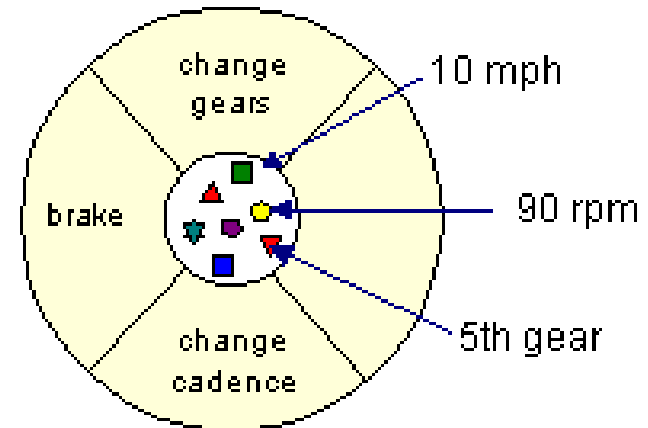
De great Poppini at-a your service.



Đối tượng



Đối tượng phần mềm



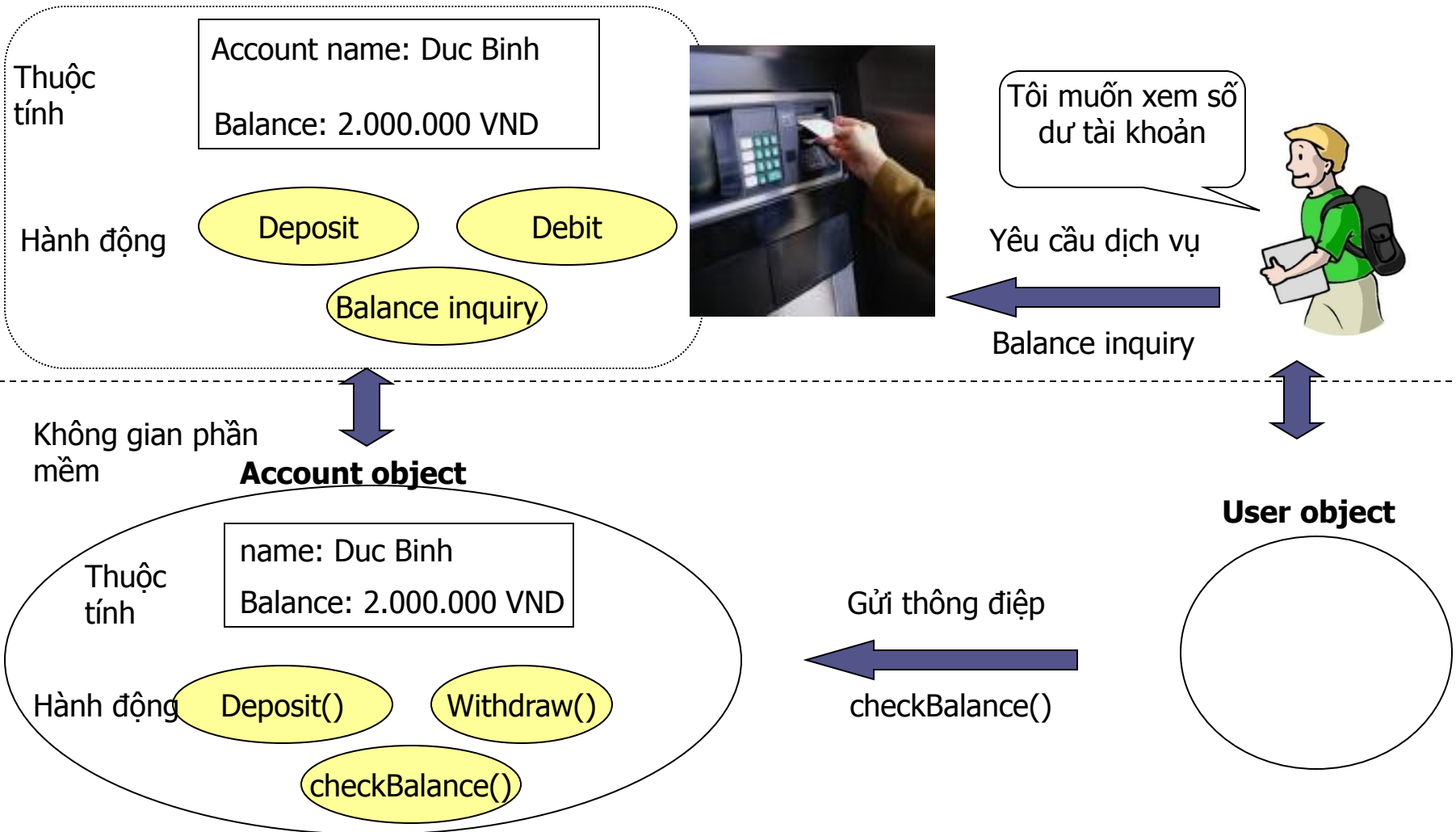
Đối tượng phần mềm **Xe Đạp**

Đối tượng (object) là một thực thể phần mềm bao bọc các **thuộc tính** và các **phương thức** liên quan.

Thuộc tính được xác định bởi giá trị cụ thể gọi là **thuộc tính thể hiện**.
Một đối tượng cụ thể được gọi là một **thể hiện**.

Đối tượng phần mềm và bài toán thực tiễn

Bài toán quản lý tài khoản ngân hàng – thẻ ATM – thanh toán điện tử

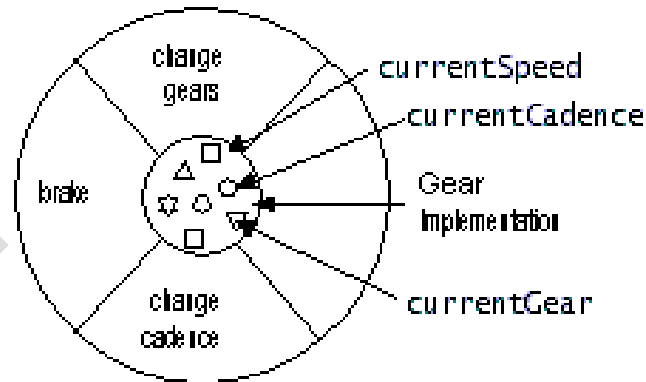


2.2 Lớp

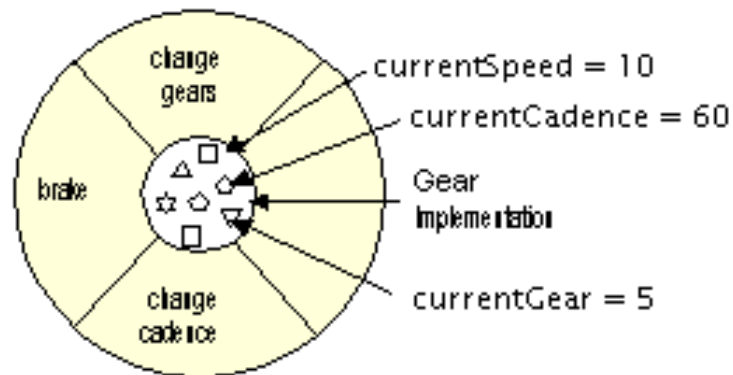
- Một **lớp** là một thiết kế (blueprint) hay mẫu (prototype) cho các đối tượng cùng kiểu
 - Ví dụ: lớp XeDap là một thiết kế chung cho nhiều đối tượng xe đạp được tạo ra
- Lớp định nghĩa các thuộc tính và các phương thức chung cho tất cả các đối tượng của cùng một loại nào đó
- Một đối tượng là một thể hiện cụ thể của một lớp.
 - Ví dụ: mỗi đối tượng xe đạp là một thể hiện của lớp XeDap
- Mỗi thể hiện có thể có những thuộc tính thể hiện khác nhau
 - Ví dụ: một xe đạp có thể đang ở bánh răng thứ 5 trong khi một xe khác có thể là đang ở bánh răng thứ 3.

Ví dụ Lớp Xe đạp

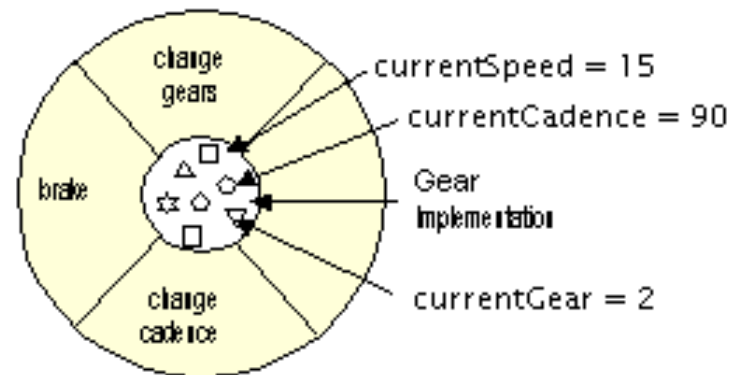
Khai báo cho lớp XeDap



Đối tượng của lớp XeDap

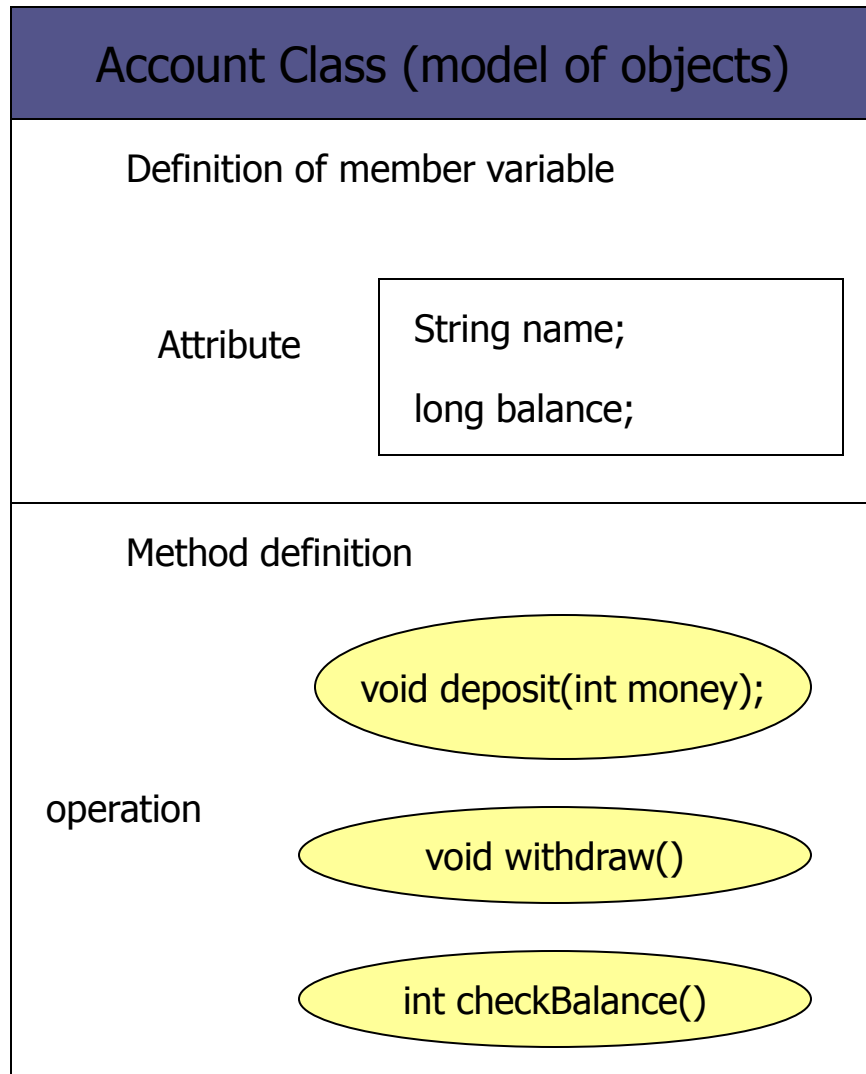


MyBike

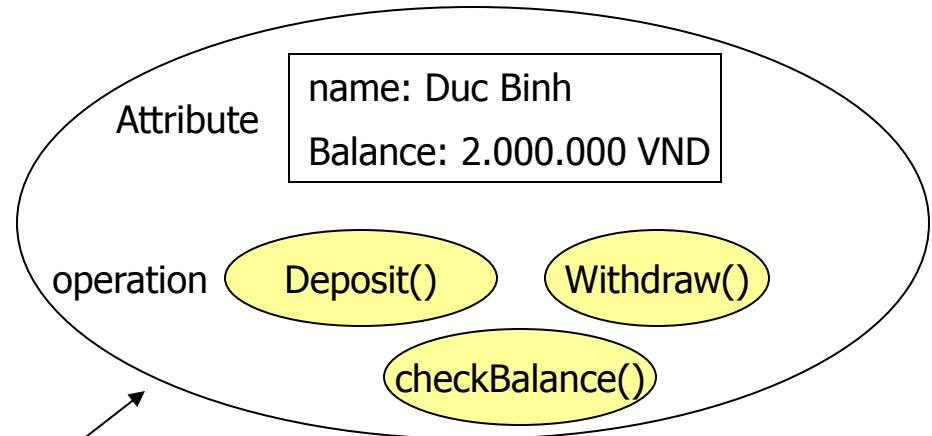


YourBike

Lớp và Đối tượng

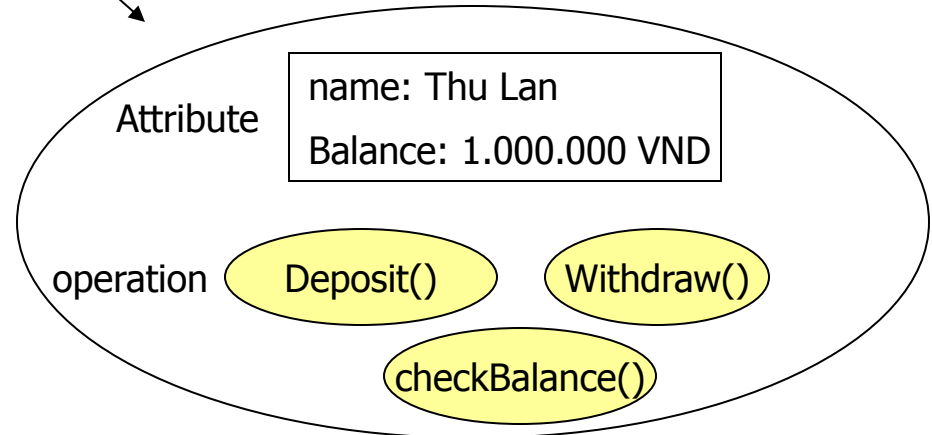


Account object of Mr Duc Binh



INstantiate

Account object of Mrs Thu Lan

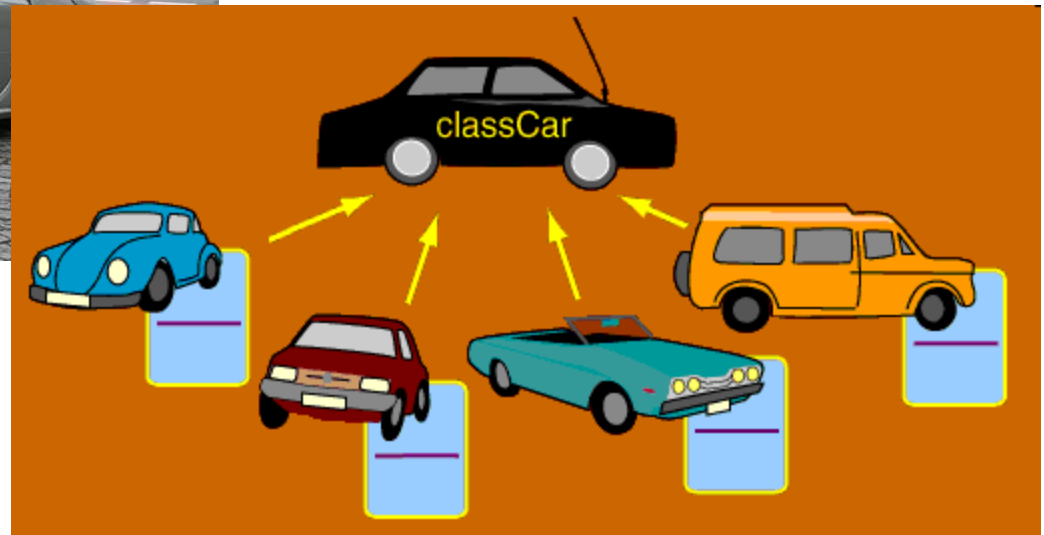


Lớp và đối tượng



Thể hiện

Thiết kế mẫu

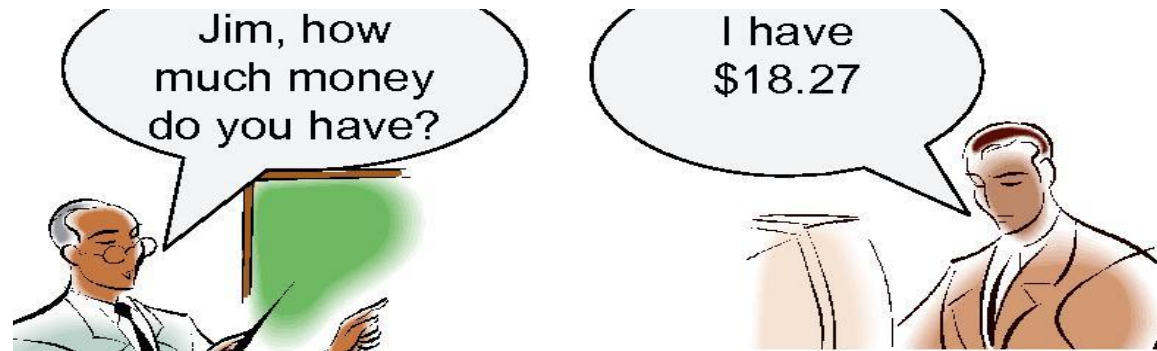


Câu hỏi nhanh

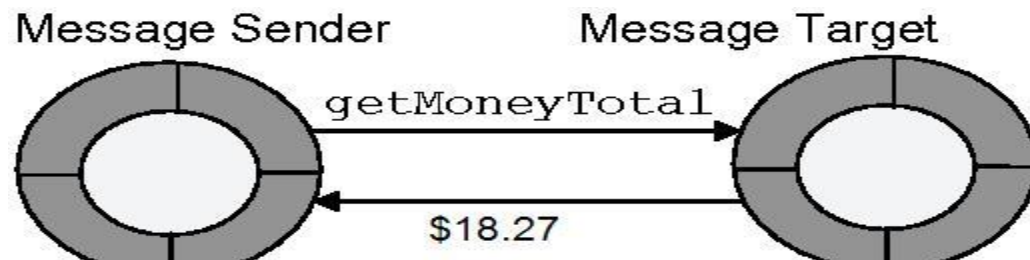
- Xét hệ thống bán hàng trực tuyến Amazon. Đưa ra một số ví dụ về lớp và đối tượng trong hệ thống?
- Cùng câu hỏi cho hệ thống thông tin quản lý đào tạo SIS ĐHBK?

2.3 Tương tác giữa các đối tượng

- Sự giao tiếp giữa các đối tượng trong thế giới thực:



- Các đối tượng và sự tương tác giữa chúng trong lập trình
 - Các đối tượng giao tiếp với nhau bằng cách gửi thông điệp (message)



Trao đổi thông điệp

- Một chương trình (xây dựng theo tiếp cận HĐT) là tập các đối tượng trao đổi thông điệp với nhau

Employee Object



Behaviors

get_SS#()
get_Gender()
get_Date_of_Birth()

Message - get_SS#()

Payroll Object



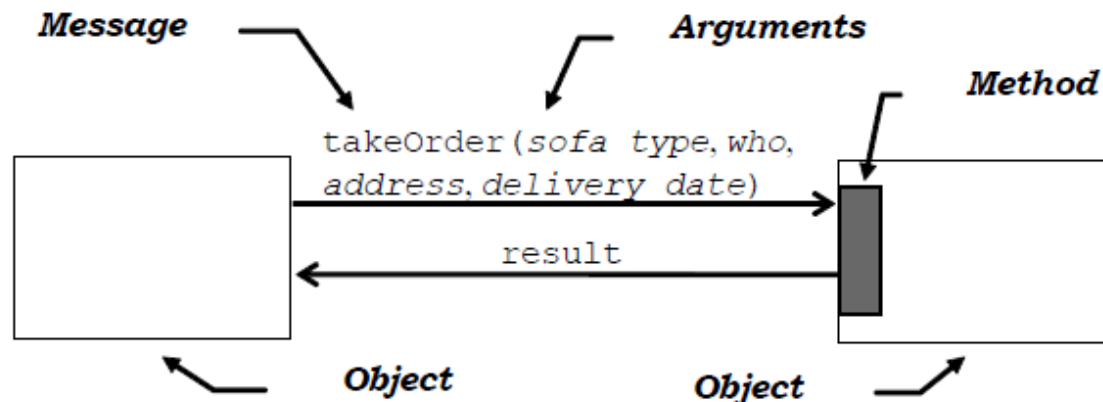
return SS#

Gọi hàm vs. Gửi thông điệp

- Gọi hàm (Call function)
 - Chỉ ra chính xác đoạn mã nào sẽ được thực hiện.
 - Chỉ có duy nhất một sự thực thi của một hàm với một tên nào đó.
 - Không có hai hàm trùng tên
- Gửi thông điệp
 - **Yêu cầu một dịch vụ từ một đối tượng và đối tượng sẽ quyết định cần phải làm gì**
 - **Các đối tượng khác nhau sẽ có các cách thực thi các thông điệp theo cách khác nhau.**

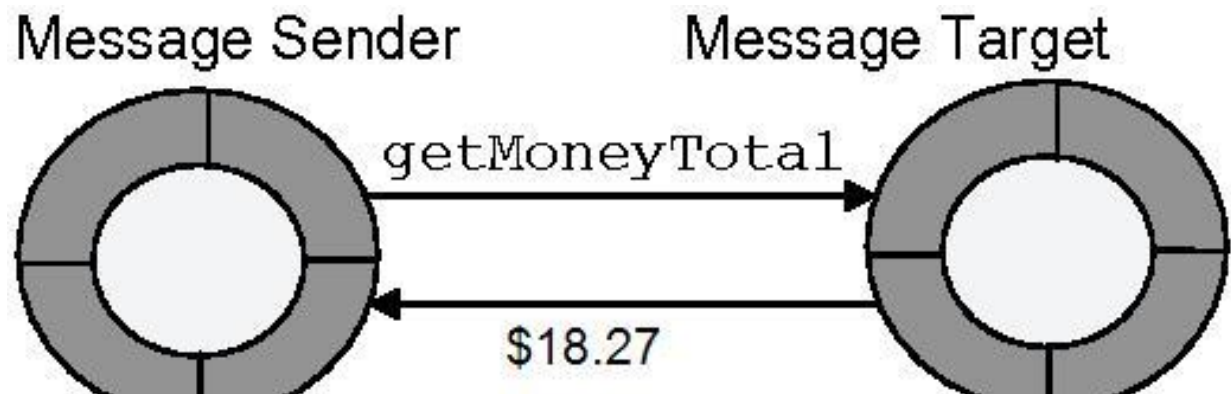
Thông điệp vs. Phương thức

- Thông điệp
 - Được gửi từ đối tượng này đến đối tượng kia, không bao gồm đoạn mã thực sự sẽ được thực thi
- Phương thức
 - Thủ tục/hàm trong ngôn ngữ lập trình cấu trúc
 - Là sự thực thi dịch vụ được yêu cầu bởi thông điệp
 - Là đoạn mã sẽ được thực thi để đáp ứng thông điệp được gửi đến cho đối tượng



2.4 Hướng cấu trúc vs. Hướng ĐT?

- Hướng cấu trúc:
 - data structures + algorithms = Program
 - (cấu trúc dữ liệu + giải thuật = Chương trình)
- Hướng đối tượng:
 - objects + messages = Program
 - (đối tượng + thông điệp = Chương trình)



Hướng cấu trúc - Hướng đối tượng

- Lập trình cấu trúc (procedural Programming):
 - Đơn vị là các thủ tục, hàm
 - Dữ liệu có ranh giới nhất định với thủ tục
- Lập trình Hướng đối tượng
 - Đơn vị là đối tượng
 - Dữ liệu gắn với hàm (phương thức) trong đối tượng
 - Mỗi cấu trúc dữ liệu có các phương thức hoạt động trên nó

2.5 Giao diện của đối tượng

- Thử thách của LTHDT là có thể ánh xạ một phần tử (thực thể) trong không gian bài toán về một đối tượng trong không gian lời giải.
- Một đối tượng có thể được sử dụng khi nó có thể đáp ứng được một số "yêu cầu" nào đó từ bên ngoài. Giao diện của đối tượng định nghĩa các dịch vụ mà đối tượng cung cấp

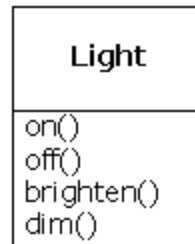
C++

```
Light  
lt;
```

```
lt.on();
```

Type Name

Interface



Java

```
Light lt = new Light();  
lt.on();
```

Giao diện của đối tượng

- Quyết định bằng cách áp dụng nguyên lý Đóng gói.

Ví dụ về lớp và đối tượng trong một số NNLT

Class declaration: **each class** is, by default, an **extension of Object** (can be omitted)

Class constructor: **initialises** the various **fields**

Class method: **retrieves** and/or **modifies** the **state** of the class

```
public class Time extends Object {
    private int hour;
    private int minute;
    private int second;

    public Time () {
        setTime(0, 0, 0);
    }

    public void setTime (int h, int m, int s) {
        hour = ( ( h >= 0 && h < 24 ) ? h : 0 );
        minute = ( ( m >= 0 && m < 60 ) ? m : 0 );
        second = ( ( s >= 0 && s < 60 ) ? s : 0 );
    }
}
```

Class fields: **private** means they **can not be accessed** from **outside** the class

Java: Chương trình và các đối tượng

```
public class Test {  
  
    public static void main (String args[]) {  
        Time time = new Time();  
  
        time.hour = 7;  
        time.minute = 15;  
        time.second = 30;  
    }  
}
```

```
Test.java:6: hour has private access in Time  
            time.hour = 7;  
              ^  
Time.java:7: minute has private access in Time  
            time.minute = 15;  
              ^  
Time.java:8: second has private access in Time  
            time.second = 30;  
              ^  
3 errors
```

Lớp Time trong C++

15

Class definition bắt đầu bằng từ khoá **class**.

Class body bắt đầu bằng ngoặc mở.

Function prototype cho các **public** member function.

Class Time
definition
(1 of 1)

Constructor: thành viên trùng tên với tên class, **Time**, và không có giá trị trả về.

```
1  class Time {  
2  
3  public:  
4      Time(); // constructor  
5      void setTime(int, int, int); // set hour, minute, second  
6      void printUniversal(); // print universal-time format  
7      void printStandard(); // print standard-time format  
8  
9  private:  
10     int hour; // 0 - 23 (24-hour clock format)  
11     int minute; // 0 - 59  
12     int second; // 0 - 59  
13  
14 }; // end class Time
```

Nhận quyền truy nhập

private data member chỉ có thể được truy nhập từ các member function.

Chương trình và các đối tượng: C++

```
1 // Fig. 6.7: fig06_07.cpp
2 // Program to test class Time.
3 // NOTE: This file must be compiled with timel.cpp.
4 #include <iostream>
5
6 using std::cout;
7 using std::endl;
8
9 // include definition of class Time from timel.h
10 #include "timel.h"
11
12 int main()
13 {
14     Time t; // instantiate object t of class Time
15
16     // output Time object t's initial values
17     cout << "The initial universal time is ";
18     t.printUniversal(); // 00:00:00
19     cout << "\nThe initial standard time is ";
20     t.printStandard(); // 12:00:00 AM
21
22     t.setTime( 13, 27, 6 ); // change time
23 }
```

fig06_07.cpp
(1 of 2)

Include **timel.h** để đảm bảo tạo đúng và để tính kích thước đối tượng thuộc lớp **Time**.

Chương trình và các đối tượng: C++

```
24 // output Time object t's new values
25 cout << "\n\nUniversal time after setTime is ";
26 t.printUniversal(); // 13:27:06
27 cout << "\nStandard time after setTime is ";
28 t.printStandard(); // 1:27:06 PM
29
30 t.setTime( 99, 99, 99 ); // attempt invalid settings
31
32 // output t's values after specifying invalid values
33 cout << "\n\nAfter attempting invalid settings:"
34 << "\nUniversal time: ";
35 t.printUniversal(); // 00:00:00
36 cout << "\nStandard time: ";
37 t.printStandard(); // 12:00:00 AM
38 cout << endl;
39
40 return 0;
41
42 } // end main
```

fig06_07.cpp
(2 of 2)

fig06_07.cpp
output (1 of 1)

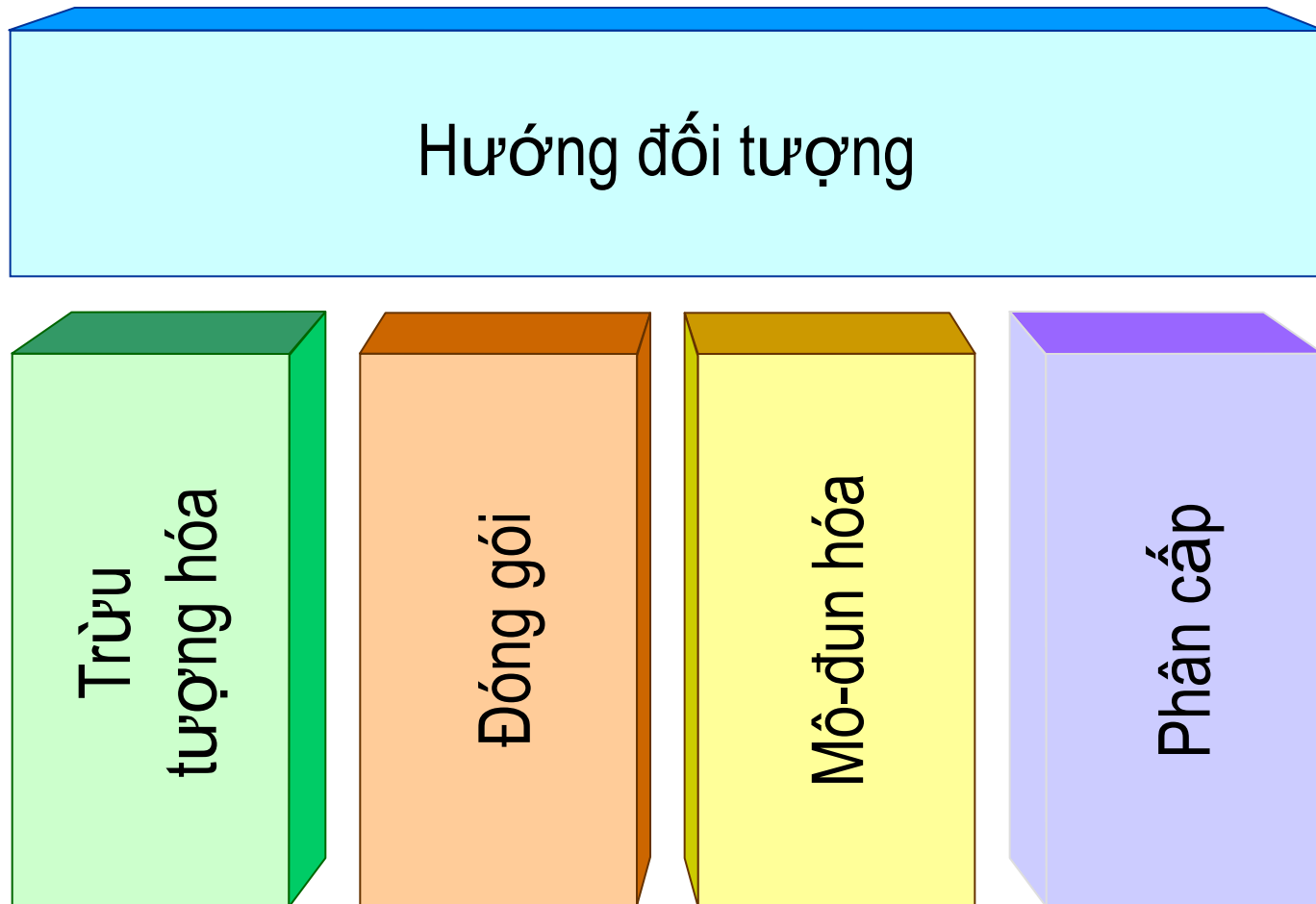
```
The initial universal time is 00:00:00
The initial standard time is 12:00:00 AM

Universal time after setTime is 13:27:06
Standard time after setTime is 1:27:06 PM
```

Nội dung

1. Công nghệ hướng đối tượng
2. Đối tượng và lớp
- 3. Các nguyên lý cơ bản
4. Phân tích thiết kế HĐT
5. Ngôn ngữ lập trình Java/C++
6. Ví dụ và bài tập

2. Các nguyên lý cơ bản của OO

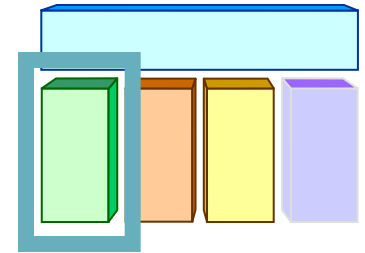


2.1 Trừu tượng hóa (abstraction)

- “Sự bỏ qua có chọn lựa”
 - Quyết định cái gì là quan trọng và không.
 - Tập trung và dựa trên những gì là quan trọng
 - Bỏ qua và không phụ thuộc vào những gì là không quan trọng

2.1. Trừu tượng hóa (Abstraction)

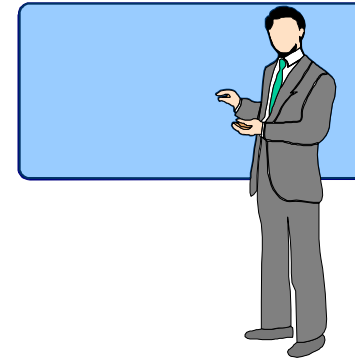
- Là quá trình loại bỏ đi các thông tin cụ thể và giữ lại những thông tin chung.
- Tập trung vào các đặc điểm cơ bản của thực thể, các đặc điểm phân biệt nó với các loại thực thể khác.
- Phụ thuộc vào góc nhìn
 - Quan trọng trong ngữ cảnh này nhưng lại không có ý nghĩa nhiều trong ngữ cảnh khác.



Ví dụ: Trừu tượng hóa



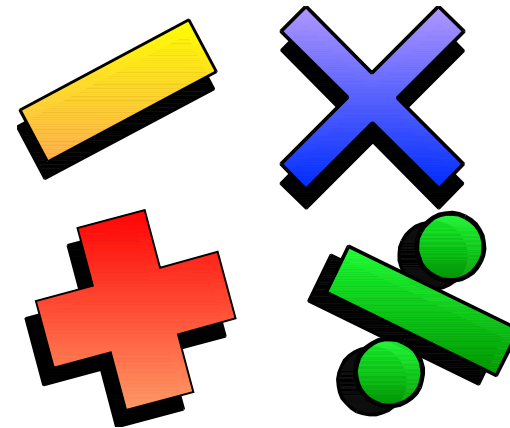
Sinh viên



Giáo viên

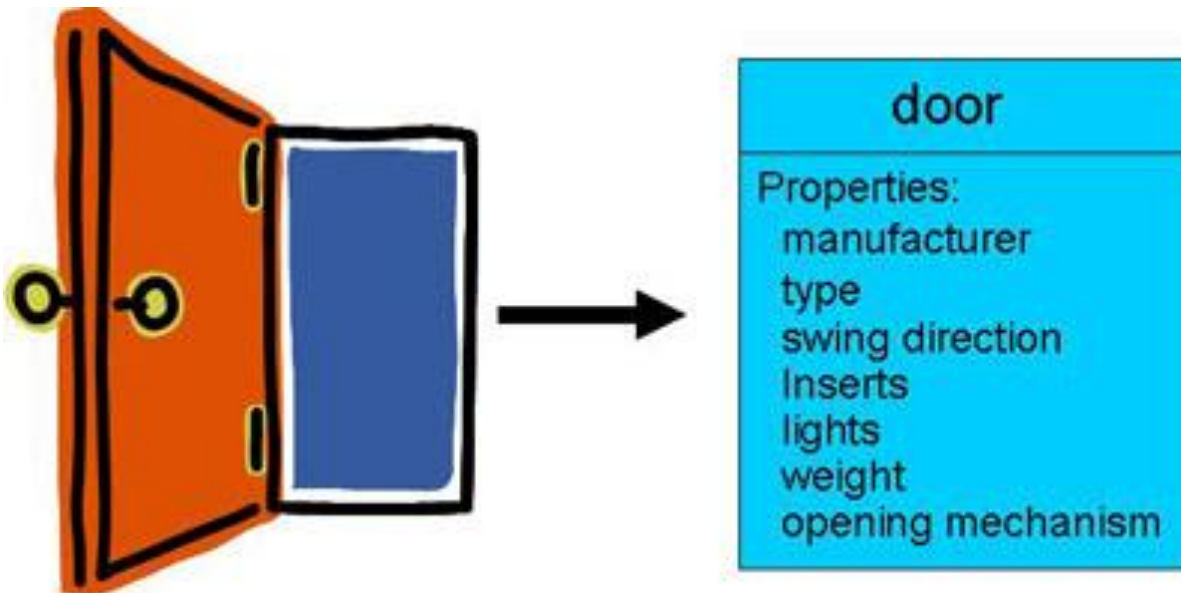


Khóa học diễn ra lúc 9:00 sáng các ngày thứ 3, 5, 7

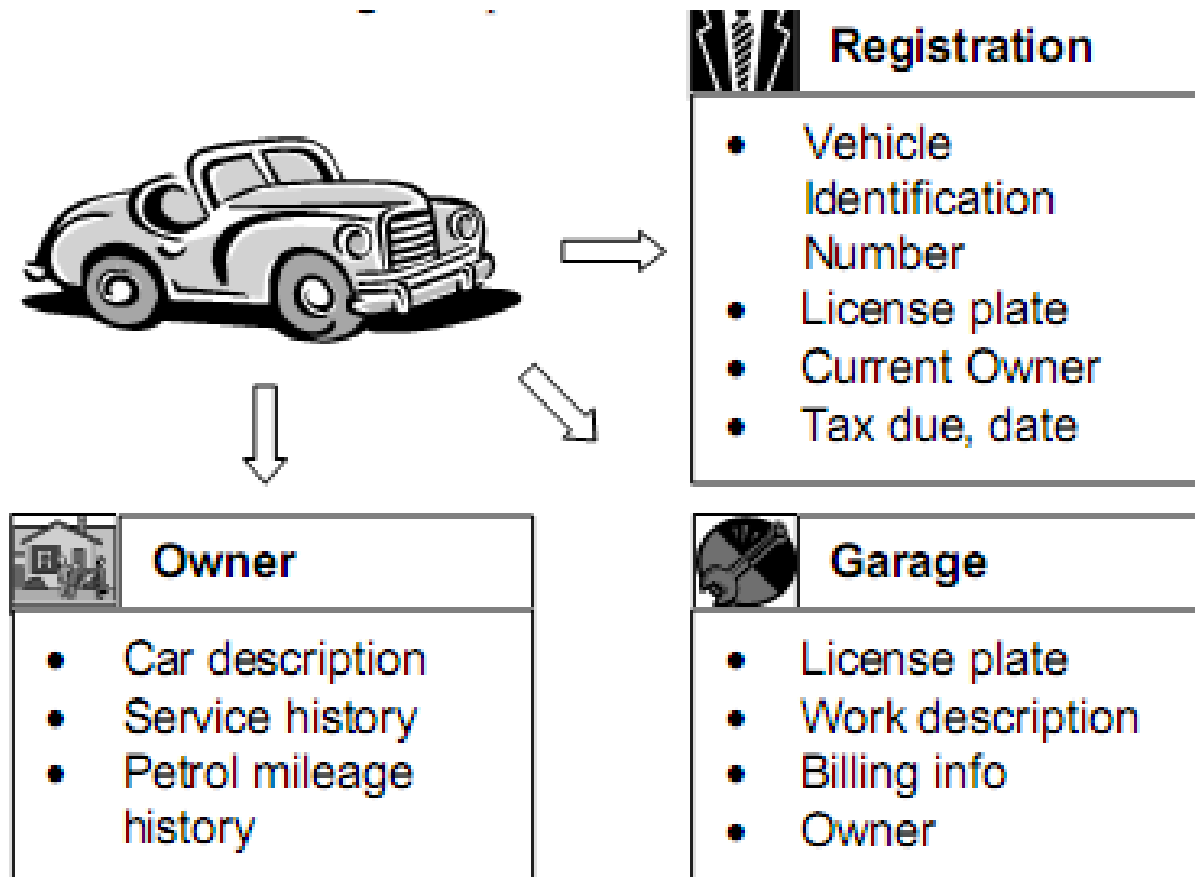


Khóa học (ví dụ đại số)

Ví dụ: Cửa ra vào



Trừu tượng hóa - Góc nhìn

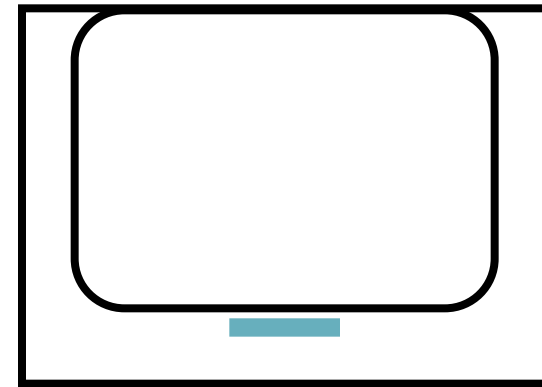
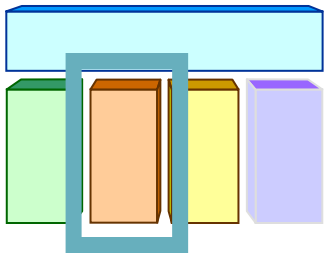


Câu hỏi trên lớp

- Hệ thống làm quen kết bạn qua mạng
 - Góc nhìn người sử dụng
 - Góc nhìn người quản lý

2.2. Đóng gói (Encapsulation)

- Che giấu, ẩn đi chi tiết thực hiện bên trong
 - Cung cấp cho thế giới bên ngoài một giao diện
 - Người dùng không phụ thuộc vào việc sửa đổi sự thực thi bên trong



Tăng cường tính mềm dẻo

Đóng gói (Encapsulation) = package together

- Kết quả của quá trình trừu tượng hóa:
 - Đối tượng = Thuộc tính + phương thức
- Che giấu:
 - Che giấu dữ liệu (thông tin)
 - Che giấu hành động: Thực hiện ẩn

Vai trò của che giấu dữ liệu

- Bảo vệ dữ liệu: Cung cấp các con đường được cấp phép để truy cập dữ liệu
- Thay đổi dữ liệu linh hoạt dễ dàng

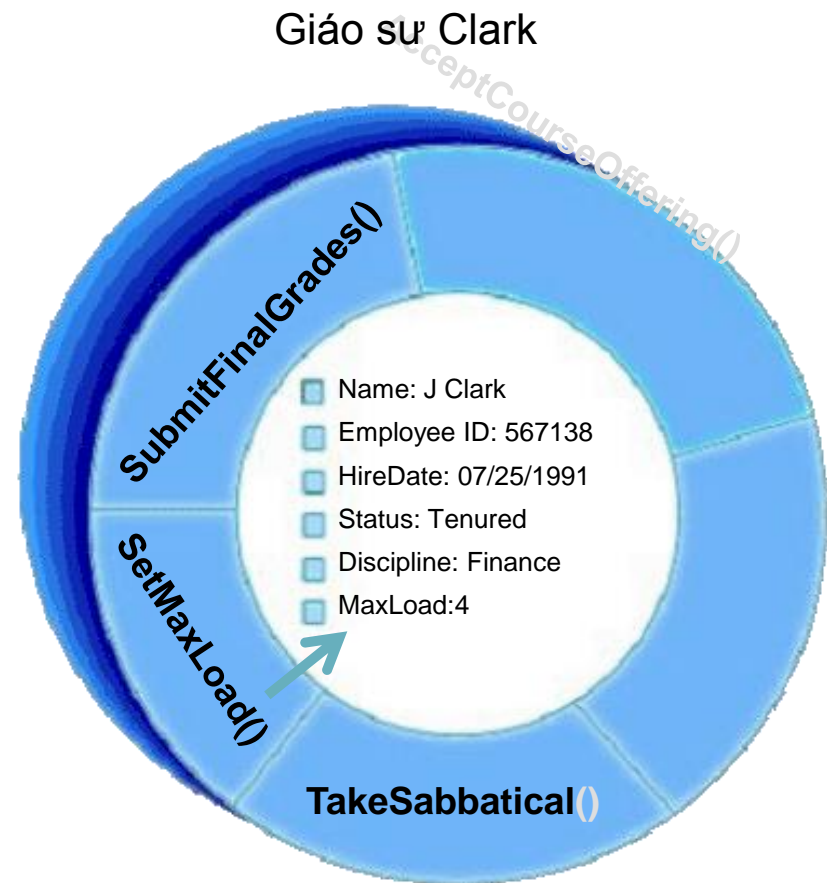

Vai trò của thực hiện ẩn

- Class creator và client programmers.
- Thực hiện ẩn cho phép:
 - Quy định những người sử dụng (client) chỉ được phép truy nhập và sử dụng những gì đã quy định cho họ. Một phần class được che dấu và không cho người sử dụng được quyền truy nhập.
 - Những người thiết kế các class có khả năng thay đổi hay định nghĩa lại class mà vẫn chắc chắn rằng không ảnh hưởng tới chương trình của những người sử dụng class này.

Minh họa việc đóng gói

- Giao diện thông điệp (phương thức) của đối tượng
- Giáo sư Clark được yêu cầu dạy 4 lớp tháng tới

SetMaxLoad(4)

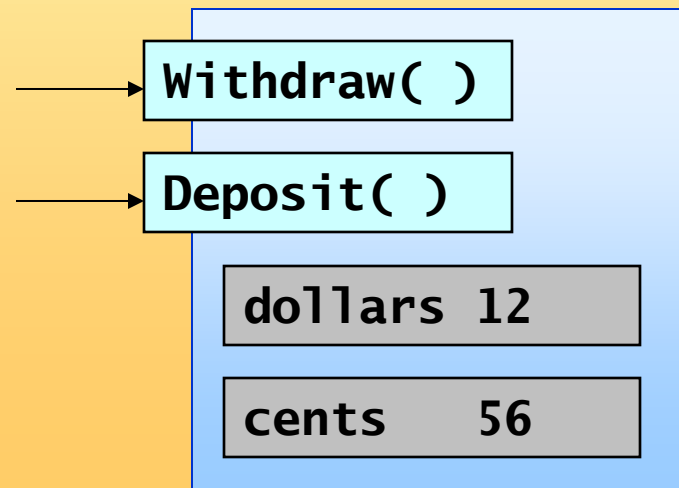
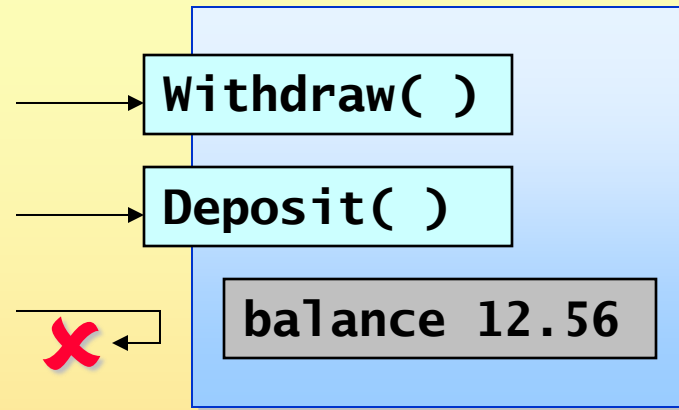


Câu hỏi thảo luận:

- Tính đóng gói đem lại lợi ích gì?

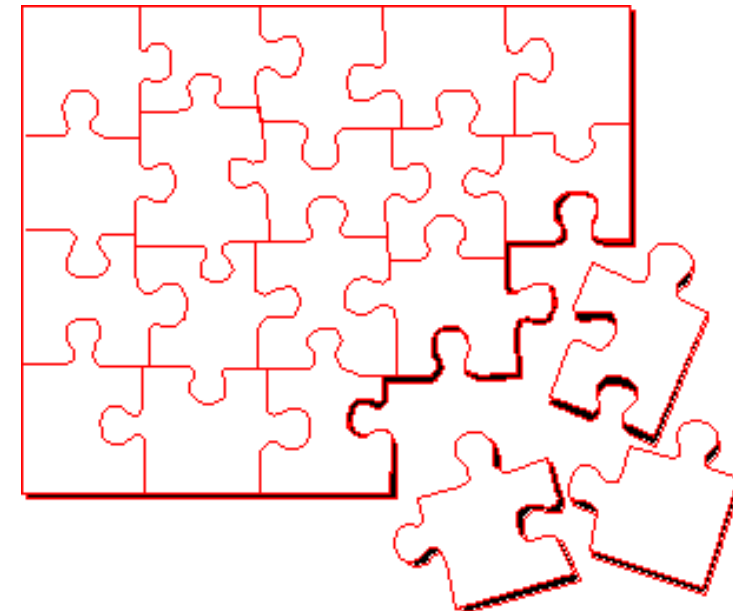
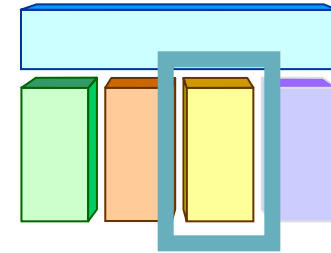
Tính đóng gói

- Cho phép điều khiển
 - Việc sử dụng đối tượng được kiểm soát thông qua các method public
- Hỗ trợ sự thay đổi
 - Việc sử dụng đối tượng không bị ảnh hưởng nếu dữ liệu nội tại (private) bị thay đổi



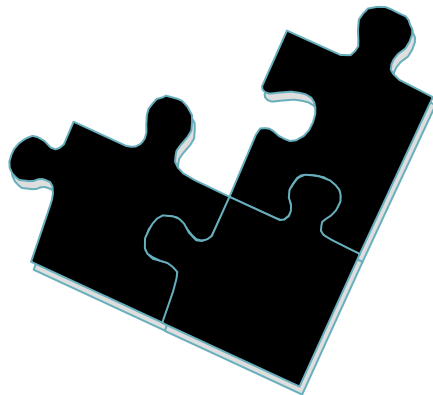
2.3. Mô đun hóa (Modularity)

- Chia nhỏ hệ thống phức tạp thành những thành phần nhỏ có thể quản lý được.
- Cho phép người dùng hiểu biết về hệ thống.

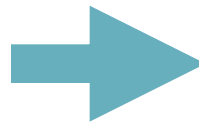


Ví dụ: Mô đun hóa

- Ví dụ, chia nhỏ một hệ thống phức tạp thành các mô đun nhỏ hơn.



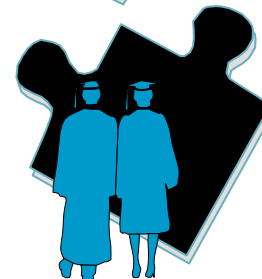
Hệ thống quản lý
siêu thị sách



Hệ thống
quản lý xuất
nhập sách



Hệ thống
kế toán

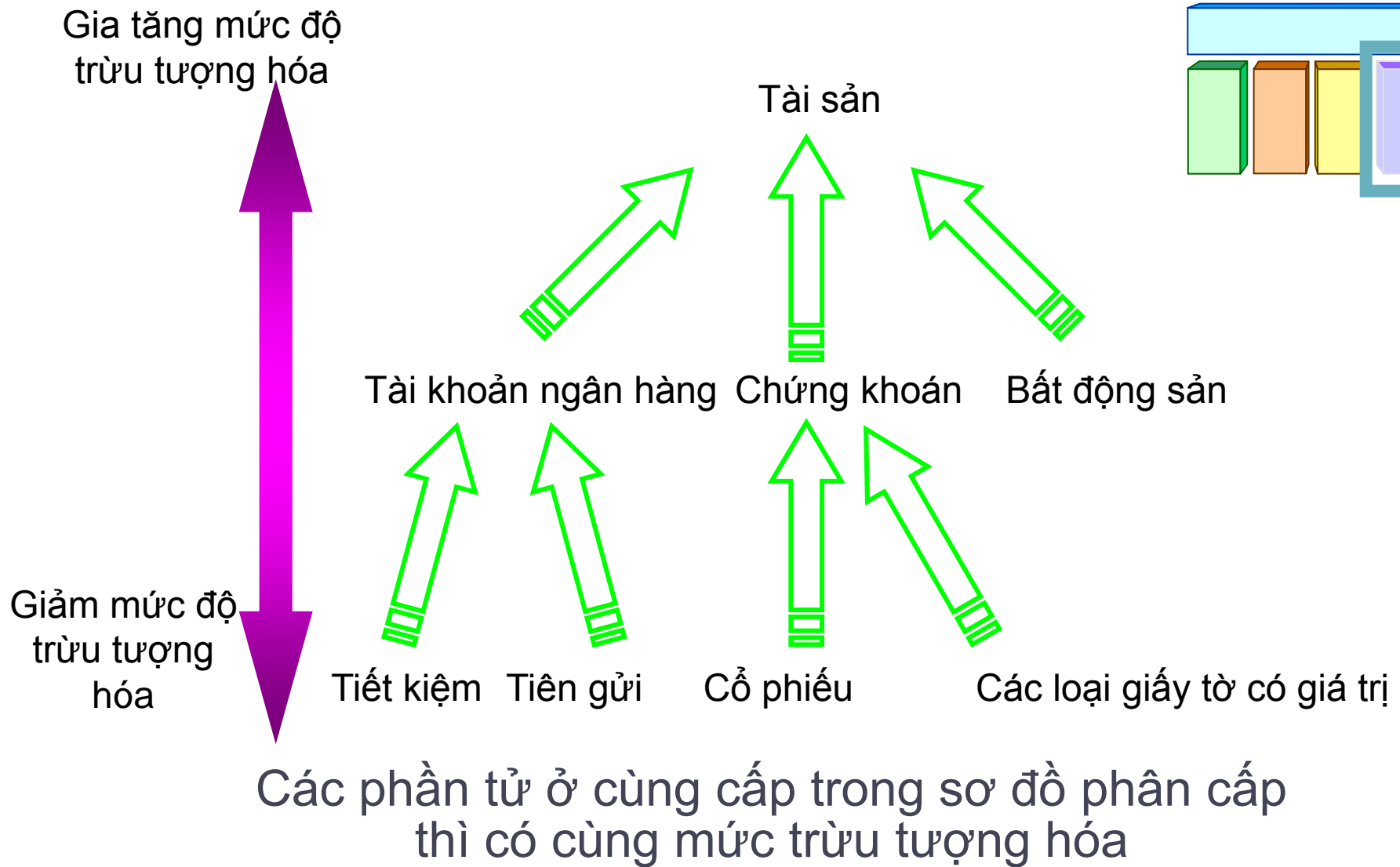


Hệ thống
quản lý
nhân viên

2.4. Phân cấp (Hierarchy)

- **Phân cấp** có thể được định nghĩa là:
 - việc xếp hạng hay xếp thứ tự các mức trừu tượng vào một cấu trúc cây. Các loại: phân cấp kết tập, phân cấp lớp, phân cấp phạm vi, phân cấp mức kế thừa, phân cấp thành phần, phân cấp mức độ chuyên môn hóa, phân cấp loại. (*Dictionary of Object Technology*, Firesmith, Eykholt, 1995.)

2.4. Phân cấp (Hierarchy)



Nội dung

1. Công nghệ hướng đối tượng
2. Đối tượng và lớp
3. Các nguyên lý cơ bản
- 4. Phân tích thiết kế HĐT
5. Ngôn ngữ lập trình Java/C++
6. Ví dụ và bài tập

Phân tích và thiết kế hướng đối tượng

- Phương pháp luận (*methodology*) trong PT&TK phần mềm thông thường được định nghĩa như là một tập các quá trình và thao tác để tìm và khám phá cách có thể giải quyết được bài toán phần mềm.
- Một trong các phương pháp hiệu quả nhất để phát triển phần mềm.

Phân tích và thiết kế hướng đối tượng

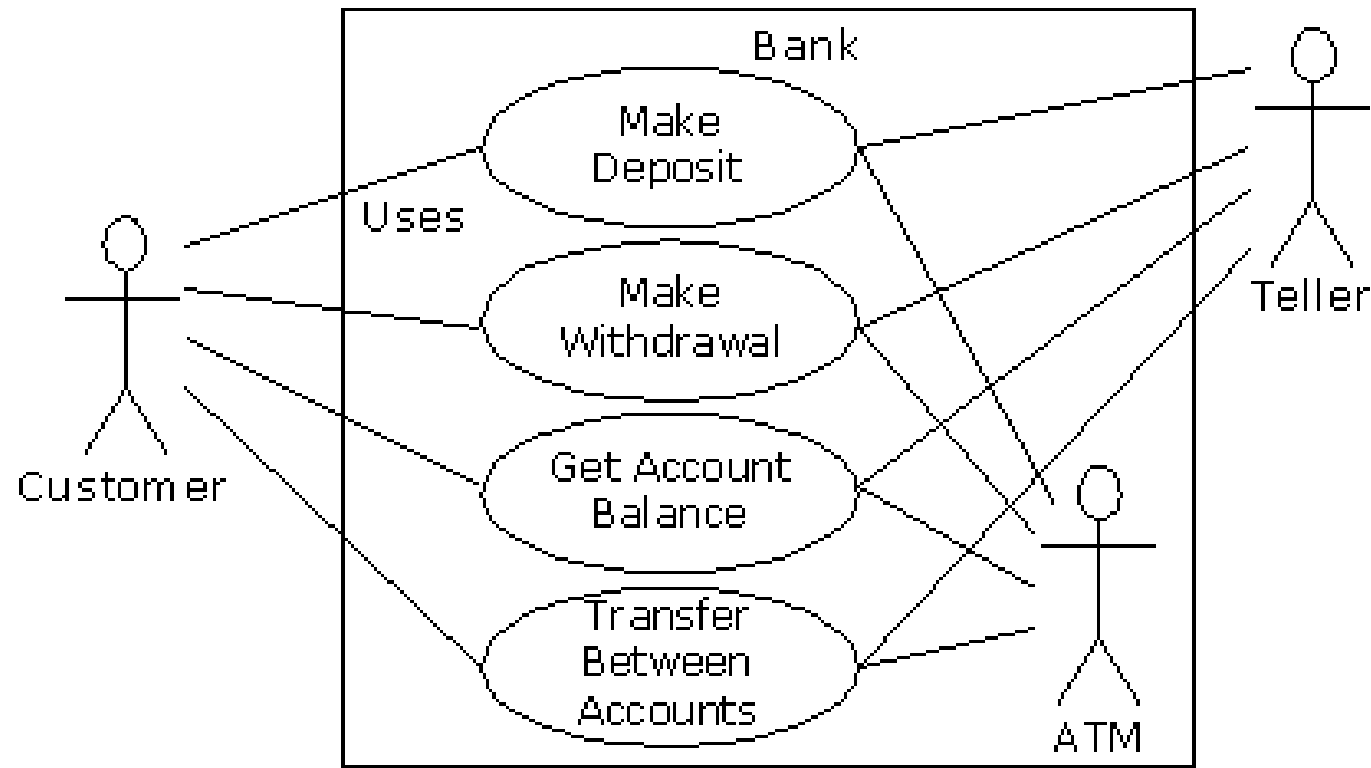
- Sáu giai đoạn
 - Giai đoạn 0: Lập kế hoạch (make a plan)
 - Giai đoạn 1: Xác định mục tiêu - làm gì (what are we making)
 - Giai đoạn 2: Xác định cách làm thế nào (how to build it)
 - Giai đoạn 3: Xây dựng phần lõi - Building the core
 - Giai đoạn 4: Lặp lại (hiệu chỉnh) các trường hợp sử dụng
 - Giai đoạn 5: Phát triển (evolution)

Xác định mục tiêu

Giai đoạn 1: Xác định mục tiêu - làm gì (what are we making)

- Trong giai đoạn này chúng ta có nhiệm vụ xác định cụ thể các mục tiêu, chức năng và nhiệm vụ mà phần mềm chúng ta cần xây dựng phải đáp ứng.
- Trong phương pháp lập trình cổ điển hướng thủ tục người ta gọi giai đoạn này là giai đoạn tạo ra “phân tích yêu cầu và mô tả hệ thống” (*requirements analysis and system specification*).
- Trong PT&TK hướng đối tượng người ta sử dụng các ký pháp và kỹ thuật Use case để mô tả các công việc này.

Biểu đồ Use case



Xác định cách làm: Lớp

- Xác định đặc tính lớp
- Thẻ tương tác - tính chất-lớp *Class-Responsibility-Collaboration* (CRC) card. Mỗi thẻ thể hiện một lớp, trên thẻ chúng ta lưu lại các thông tin sau về các lớp:

[illegible]

Khuôn dạng của thẻ CRC

- 1. Tên của lớp. Thông thường người ta đặt tên lớp liên quan đến vai trò của lớp, chúng ta sẽ sử dụng lớp để làm gì.
- 2. Trách nhiệm của lớp: lớp có thể làm gì. Thông thường các thông tin ở đây bao gồm tên của các hàm thành phần – Tri thức mà lớp đó quản lý
- 3. Tương tác của lớp: lớp này có thể tương tác được với những lớp nào khác. Những lớp cần thiết để lớp đang xét hoàn thành nhiệm vụ

CRC card

BankAccount	
Super Classes :	
Sub Classes : SavingAccount, MarginAccount	
Description : Store the transaction record, customer data, balance, etc.	
Attributes :	
Name	Description
accountNumber	A unique value to identify the accounts
Responsibilities :	
Name	Collaborator
Keep the latest value of the balance	Bank controller, Transaction records

SavingAccount	
Super Classes : BankAccount	
Sub Classes :	
Description : Store the cash information of the customer record.	
Attributes :	
Name	Description
cashBalance	latest value of the cash balance
Responsibilities :	
Name	Collaborator
getBalance	TransactionController, AccountControl

BankController	
Super Classes :	
Sub Classes : AccountController, TransactionController, ATMController	
Description : Control the interactions between the customer and the bank system.	
Attributes :	
Name	Description
status	identify the status of the controller
Responsibilities :	
Name	Collaborator
withDraw	Withdraw money from the bank acco

ATMController	
Super Classes : BankController	
Sub Classes :	
Description : Control the interactions between customer and the ATM terminals.	
Attributes :	
Name	Description
machineType	Identify the type of the ATM terminal
Responsibilities :	
Name	Collaborator
checkingPass-word	

Một số đặc điểm cần lưu ý khi phát triển các lớp

1. Chúng ta cần phân biệt rõ việc tạo ra lớp, sau đó mới nghĩ tới việc phát triển và hoàn thiện lớp trong quá trình giải quyết bài toán
2. Việc phát hiện ra các lớp cần thiết cho chương trình là một trong những nhiệm vụ chính của thiết kế hệ thống, nếu chúng ta đã có những lớp này (trong một thư viện lớp nào đó chẳng hạn) thì công việc sẽ dễ dàng hơn
3. Khi phân tích hay phát triển các lớp không nên tập trung vào để biết tất cả về một lớp, chúng ta sẽ biết tất cả các thuộc tính này khi phát triển hệ thống (learns as you go)

Một số đặc điểm cần lưu ý khi phát triển các lớp

4. Khi tiến hành lập trình cần tuân thủ theo các thiết kế đã làm. Không nên băn khoăn rằng chúng ta sẽ không sử dụng phương pháp lập trình truyền thống và cảm thấy choáng ngợp trước số lượng lớn các đối tượng.
5. Luôn luôn giữ nguyên tắc: mọi vấn đề cần giải quyết theo phương án đơn giản nhất, không nên phức tạp hóa. Sử dụng nguyên lý của Occam Razor: *Lớp đơn giản nhất bao giờ cũng là lớp tốt nhất, hãy bắt đầu bằng những cái đơn giản và chúng ta sẽ kết thúc bằng những hệ thống phức tạp*

Xác định cách làm: Đối tượng

- Trong giai đoạn này , một trong những yêu cầu quan trọng đó là thiết kế các đối tượng (object design). Trong PT&TK hướng đối tượng người ta đã tổng kết 5 bước để thiết kế đối tượng:
- Bước 1. Phát hiện đối tượng (Object discovery). Bước này được thực hiện ở giai đoạn phân tích chương trình. Chúng ta phát hiện các đối tượng nhờ các yếu tố bên ngoài và các đặc điểm bên ngoài
- Bước 2. Lắp ráp đối tượng (Object assembly). Bước tìm kiếm các đặc điểm của đối tượng để thêm vào các thuộc tính, các hàm thành phần cho đối tượng.

Xác định cách làm: Đối tượng

Bước 3. Xây dựng hệ thống (System construction). Trong giai đoạn này chúng ta phát triển các đối tượng, xem xét các tương tác giữa các đối tượng để hình thành hệ thống hoạt động.

Bước 4. Mở rộng hệ thống (System extension). Khi chúng ta thêm vào các tính năng của hệ thống, chúng ta cần thêm các lớp mới, các đối tượng mới và các tương tác giữa các đối tượng này với các đối tượng đã có trong hệ thống.

Bước 5. Tái sử dụng đối tượng (Object reuse). Đây là một trong những thử nghiệm quan trọng của các đối tượng và lớp trong thiết kế phần mềm. Chúng ta cần phải sử dụng lại các lớp và các đối tượng trong phần mềm (thông qua tính kế thừa và tương tác giữa các đối tượng)

Nội dung

1. Công nghệ hướng đối tượng
2. Đối tượng và lớp
3. Các nguyên lý cơ bản
4. Phân tích thiết kế HĐT
- 5. Ngôn ngữ lập trình Java/C++
6. Ví dụ và bài tập

5.1 Ngôn ngữ C++

- Phiên bản cải tiến ngôn ngữ lập trình C thành ngôn ngữ lập trình hướng đối tượng.
- Tất cả các chương trình đã có sẵn của C sẽ chạy được trong C++
- C++ có những đặc tính phát triển tốt hơn ngôn ngữ C:
 - Trong C++ có đặc tính tham chiếu của các biến (*references*) cho phép quản lý địa chỉ của các biến hay lấy ra giá trị của các biến ở các địa chỉ tương ứng.
 - Quản lý tên hàm và biến đã được mở rộng thông qua cơ chế chồng hàm *function overloading*,

5.1 Ngôn ngữ C++

- Tư tưởng phân vùng các biến *namespaces* cho phép quản lý các biến được tốt hơn.
- Tính hiệu quả
- Các phần mềm xây dựng trở nên dễ hiểu hơn
- Hiệu quả sử dụng của các thư viện
- Khả năng sử dụng lại mã thông qua templates
- Quản lý lỗi
- Cho phép xây dựng các phần mềm lớn hơn

5.2.1 Java là gì?



- Java là một ngôn ngữ lập trình HĐT được phát triển bởi Sun Microsystems.
- Java là một ngôn ngữ lập trình khá trẻ
 - Ban đầu được sử dụng để xây dựng ứng dụng điều khiển các bộ xử lý bên trong các thiết bị điện tử dân dụng như máy điện thoại cầm tay, lò vi sóng...
 - Bắt đầu được sử dụng từ năm 1995



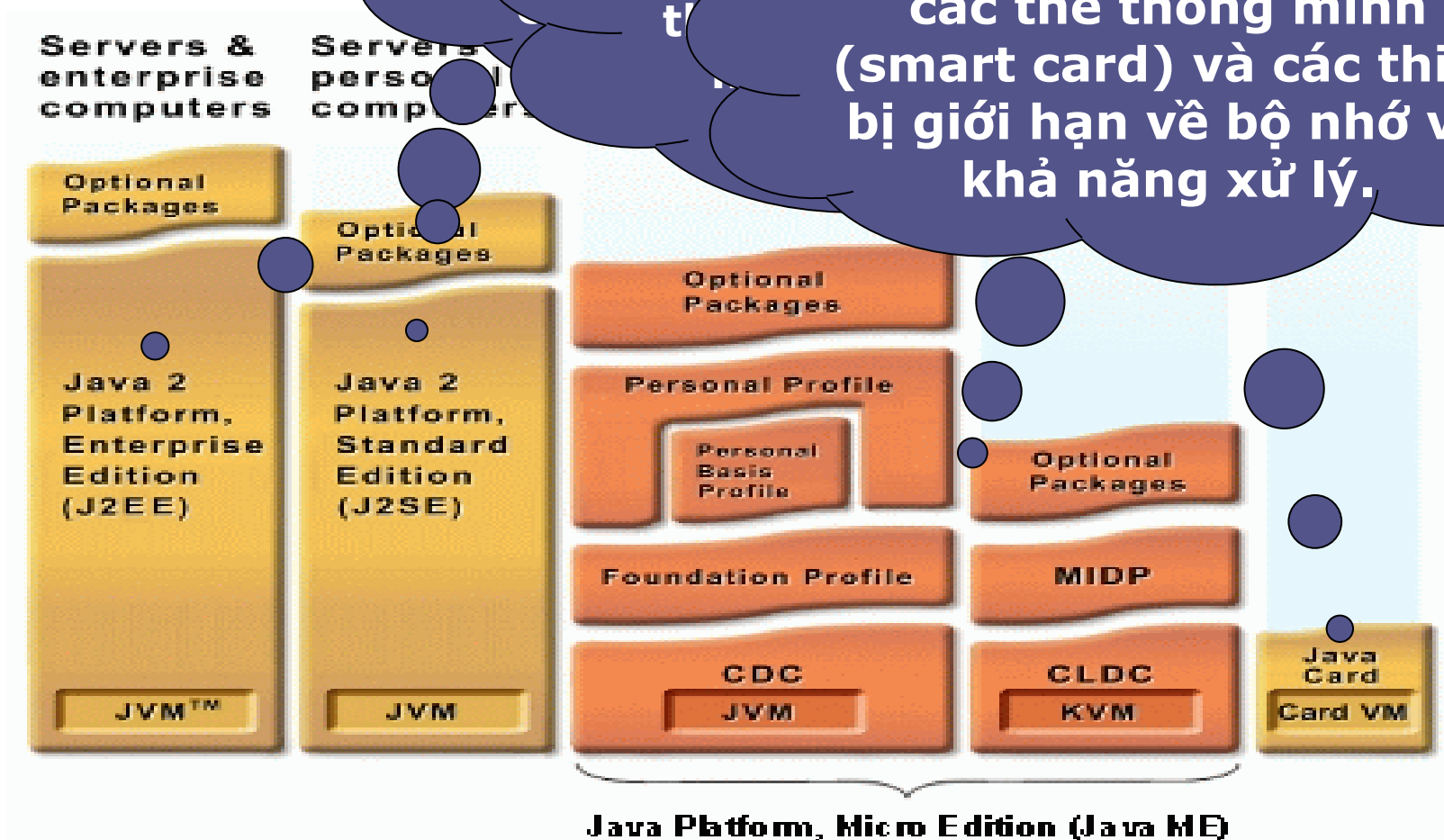
Green Team and James Gosling
(the leader)

Java là gì?

- Ngày nay, nhắc đến Java, không còn nhắc đến như một ngôn ngữ mà còn là một công nghệ, một nền tảng phát triển.
- Java có một cộng đồng phát triển mạnh mẽ
 - Một tập hợp các thư viện với số lượng lớn (từ Sun và các nguồn khác)

J2SE: Cung cấp các thành phần cốt lõi nhất để xây các ứng dụng desktop, server.

Card: Cung cấp môi trường an toàn chạy trên các thẻ thông minh (smart card) và các thiết bị giới hạn về bộ nhớ và khả năng xử lý.



J2SE (Java 2 Platform Standard Edition)

- <http://java.sun.com/j2se>
- Java 2 Runtime Environment, Standard Edition (J2RE):
 - Môi trường thực thi hay JRE cung cấp các Java API, máy ảo Java (JVM) và các thành phần cần thiết khác để chạy các applet và các ứng dụng viết bằng Java.
- Java 2 Software Development Kit, Standard Edition (J2SDK)
 - Tập mẹ của JRE, và chứa mọi thứ nằm trong JRE, bổ sung thêm các công cụ như là trình biên dịch và các trình gỡ lỗi cần để phát triển applet và các ứng dụng.

J2EE (Java 2 Platform Enterprise Edition)

- <http://java.sun.com/j2ee>
- Service-Oriented Architecture (SOA) và Web services
- Các ứng dụng Web
 - Servlet/JSP
 - JSF...
- Các ứng dụng doanh nghiệp
 - EJB
 - JavaMail...
- ...

Lịch sử phát triển của J2SE

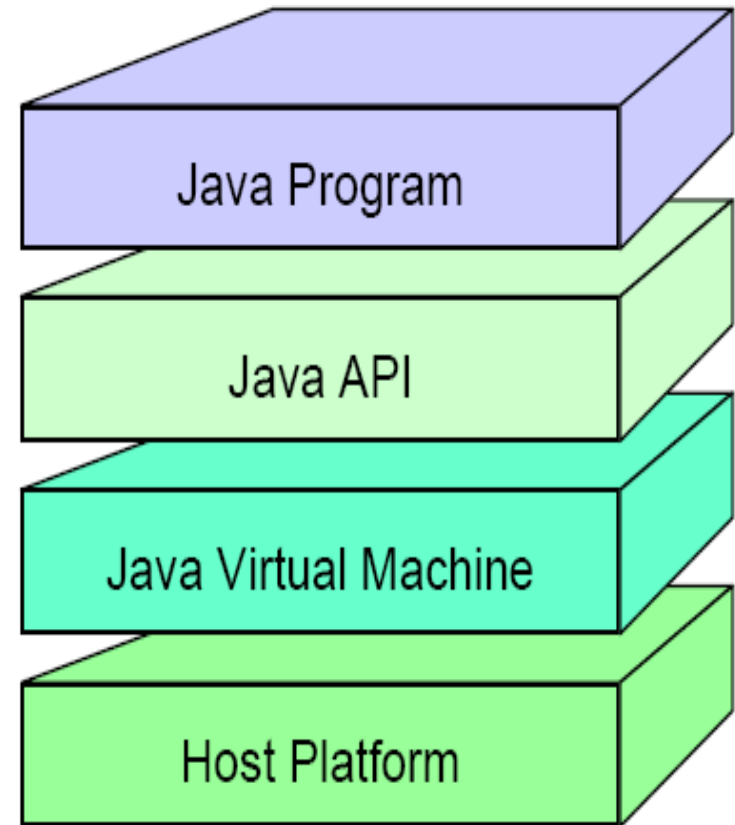
- JDK 1.1.4 (Sparkler): 12 tháng 9, 1997
- JDK 1.1.5 (Pumpkin): 3 tháng 12, 1997
- JDK 1.1.6 (Abigail): 24 tháng 4, 1998
- JDK 1.1.7 (Brutus): 28 tháng 9, 1998
- JDK 1.1.8 (Chelsea): 8 tháng 4, 1999
- J2SE 1.2 (Playground): 4 tháng 12, 1998
- J2SE 1.2.1 (none): 30 tháng 3, 1999
- J2SE 1.2.2 (Cricket): 8 tháng 7, 1999
- J2SE 1.3 (Kestrel): 8 tháng 5, 2000
- J2SE 1.3.1 (Ladybird): 17 tháng 5, 2001

Lịch sử phát triển của J2SE (2)

- J2SE 1.4.0 (Merlin) 13 tháng 2, 2002
- J2SE 1.4.1 (Hopper) 16 tháng 9, 2002
- J2SE 1.4.2 (Mantis) 26 tháng 6, 2003
- J2SE 5 (1.5.0) (Tiger) 29 tháng 9, 2004
- **Java SE 6 (Mustang)**, 11 tháng 12, 2006
 - Các bản cập nhật 2 và 3 được đưa ra vào năm 2007
 - Bản cập nhật 4 đưa ra tháng 1 năm 2008.
- **Java SE 7 (Dolphin)**, 4/2008.

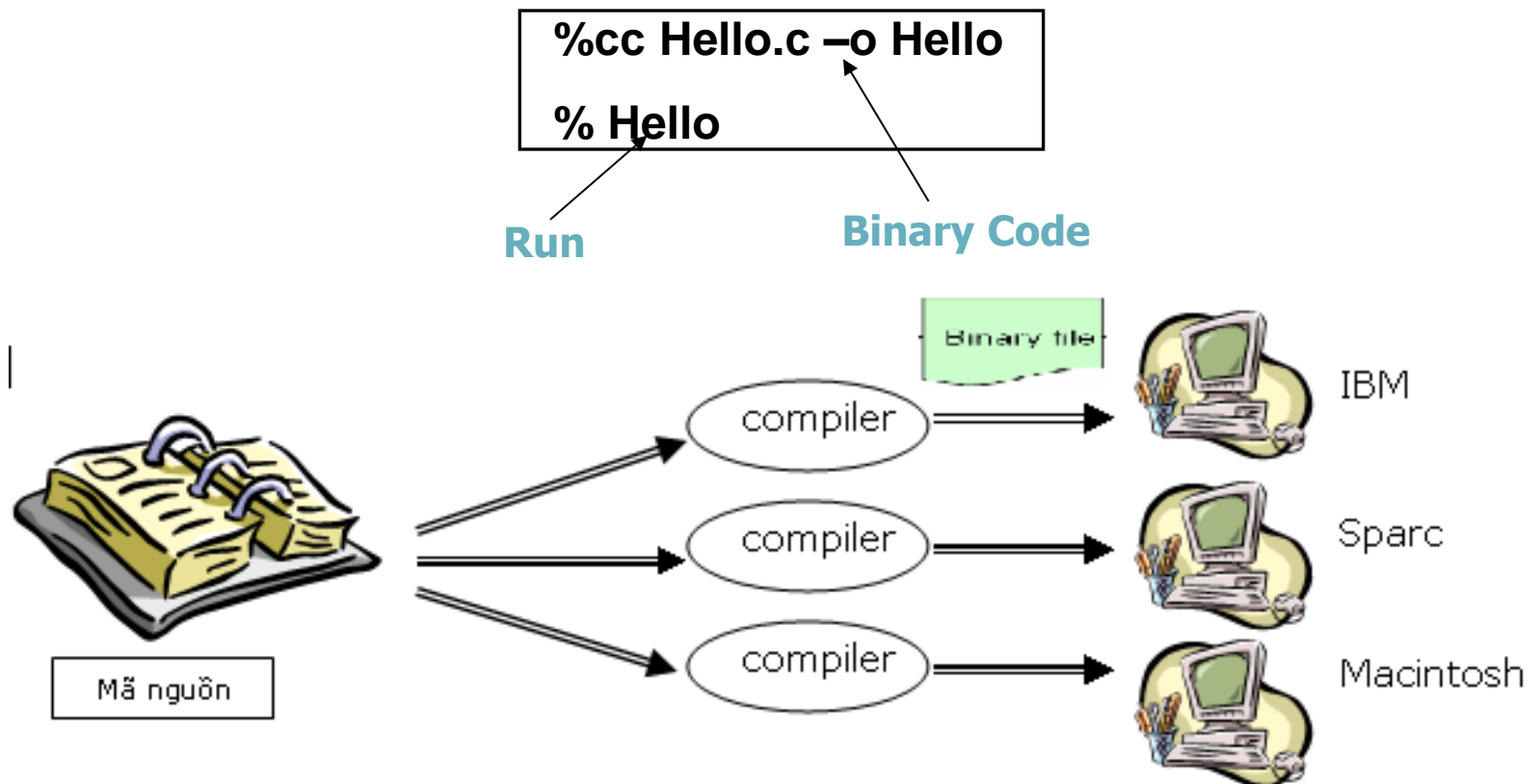
5.2.3 Nền tảng của Java (Java platform)

- Platform là môi trường phát triển hoặc triển khai.
- Java platform có thể chạy trên mọi hệ điều hành
 - Các platform khác phụ thuộc vào phần cứng
 - Java platform cung cấp:
 - Máy ảo Java - Java Virtual Machine (JVM).
 - Giao diện lập trình ứng dụng - Application Programming Interface (API).



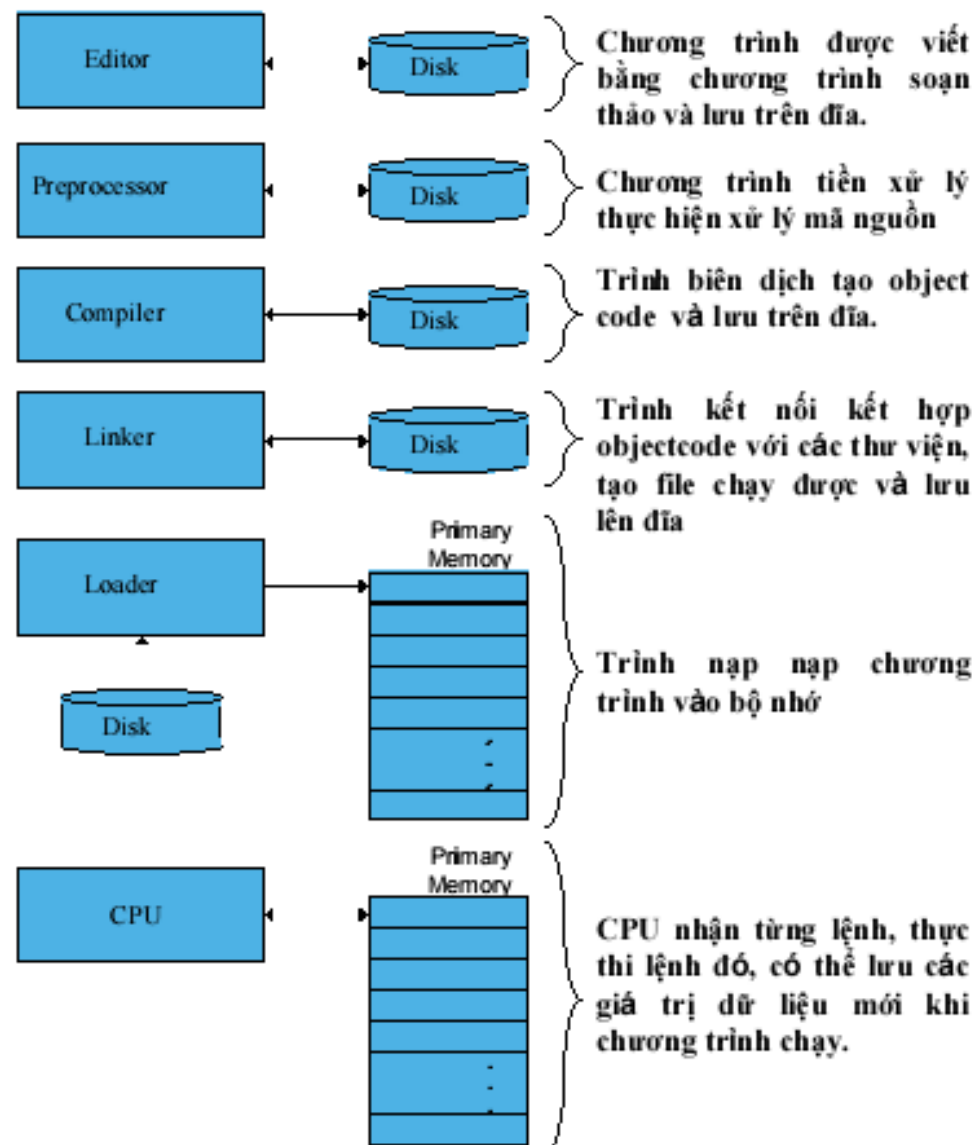
5.3. Mô hình dịch của Java

- a. Mô hình biên dịch truyền thống:
 - Mã nguồn được biên dịch thành mã nhị phân.



Các giai đoạn của chương trình C++:

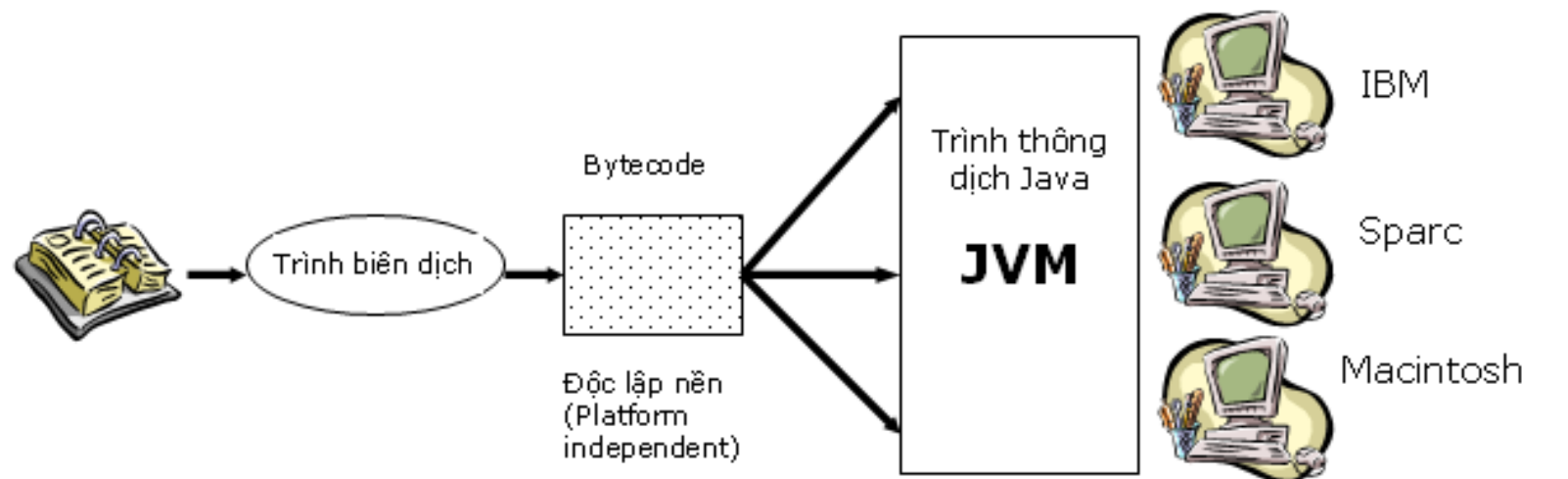
1. Soạn thảo - Edit
2. Tiền xử lý - Preprocess
3. Biên dịch - Compile
4. Liên kết - Link
5. Nạp - Load
6. Chạy - Execute



5.3. Mô hình dịch của Java (2)

- b. Mô hình dịch của Java:

- Mã nguồn được biên dịch thành bytecode rồi được thông dịch bởi JVM.



5.3. Mô hình dịch của Java (3)

- Máy ảo Java (Java Virtual Machine):
 - Máy ảo Java là trái tim của ngôn ngữ Java
 - Đem đến cho các chương trình Java khả năng viết một lần nhưng chạy được ở mọi nơi
 - Tạo ra môi trường bên trong để thực thi lệnh:
 - Nạp các file .class
 - Quản lý bộ nhớ
 - Dọn “rác”
 - Trình thông dịch “**Just In Time - JIT**”
 - Chuyển tập lệnh bytecode thành mã máy cụ thể cho từng loại CPU.

5.4. Các tính năng của Java

- Java được thiết kế:
 - Ngôn ngữ lập trình mạnh, đầy đủ tính năng và thuần hướng đối tượng.
 - Dễ học, cú pháp tương tự như C++
 - Độc lập nền tảng
 - Hỗ trợ phát triển các ứng dụng trong môi trường mạng
 - Lý tưởng cho các ứng dụng Web

5.4. Các tính năng của Java (2)

- **Mạnh mẽ**
 - Thư viện lớp: Hàng trăm lớp được viết trước với nhiều các phương thức tiện ích.
 - Java sử dụng mô hình con trỏ không cho phép truy cập trực tiếp vào bộ nhớ; bộ nhớ không thể ghi đè.
- **Hướng đối tượng**
 - Java hỗ trợ phát triển phần mềm bằng cách sử dụng khái niệm “đối tượng”
 - Phần mềm được phát triển sử dụng Java bao gồm các lớp và các đối tượng

5.4. Các tính năng của Java (3)

- Đơn giản
 - Từ khóa
 - Java có 50 từ khóa
 - So với Cobol hay VB có tới hàng trăm từ khóa
 - Có ý nghĩa đặc biệt trong ngôn ngữ
 - Được sử dụng để viết các câu lệnh
- Network capable
 - Java hỗ trợ phát triển các ứng dụng phân tán
 - Một số loại ứng dụng của Java được thiết kế để được truy cập thông qua trình duyệt Web.

5.4. Các tính năng của Java (3)

- Java có 50 từ khóa

<code>abstract</code>	<code>boolean</code>	<code>break</code>	<code>byte</code>
<code>case</code>	<code>catch</code>	<code>char</code>	<code>class</code>
<code>const</code>	<code>continue</code>	<code>default</code>	<code>do</code>
<code>double</code>	<code>else</code>	<code>extends</code>	<code>final</code>
<code>finally</code>	<code>float</code>	<code>For</code>	<code>goto</code>
<code>If</code>	<code>implements</code>	<code>import</code>	<code>instanceof</code>
<code>int</code>	<code>interface</code>	<code>long</code>	<code>native</code>
<code>new</code>	<code>package</code>	<code>private</code>	<code>protected</code>
<code>public</code>	<code>return</code>	<code>short</code>	<code>static</code>
<code>strictfp</code>	<code>super</code>	<code>switch</code>	<code>synchronized</code>
<code>this</code>	<code>throw</code>	<code>throws</code>	<code>transient</code>
<code>try</code>	<code>void</code>	<code>volatile</code>	<code>while</code>

5.4. Các tính năng của Java (5)

- Đa luồng (Multi-threaded)
 - Cho phép chương trình của bạn chạy nhiều hơn một tác vụ tại cùng một thời điểm.
- Khả chuyển (Portable)
 - Các chương trình có thể viết và biên dịch một lần, rồi chạy trên các nền tảng khác
 - Nhờ mô hình biên dịch/thông dịch (WORE – Write Once, Run Everywhere)

5.4. Các tính năng của Java (6)

- Các môi trường phát triển
 - **Java Development Kit**
 - Miễn phí trên Sun Website: java.sun.com
 - Bao gồm: Trình biên dịch, JVM và các lớp đã có
 - **Integrated Development Environments (IDEs):**
Cung cấp:
 - Các trình soạn thảo phức tạp
 - Các công cụ gỡ lỗi
 - Các công cụ phát triển đồ họa

5.5. Các kiểu chương trình Java

- Ứng dụng (Application)
 - Không cần chạy trên các trình duyệt
 - Có thể gọi các chức năng thông qua dòng lệnh hoặc menu lựa chọn (đồ họa)
 - Phương thức main() là điểm bắt đầu thực hiện ứng dụng
- Applet
 - Chương trình đồ họa chạy trên trình duyệt tại máy trạm (client).
 - Có thể được xem bằng appletviewer hoặc nhúng trong trình duyệt Web có cài JVM.

5.5. Các kiểu chương trình Java (2)

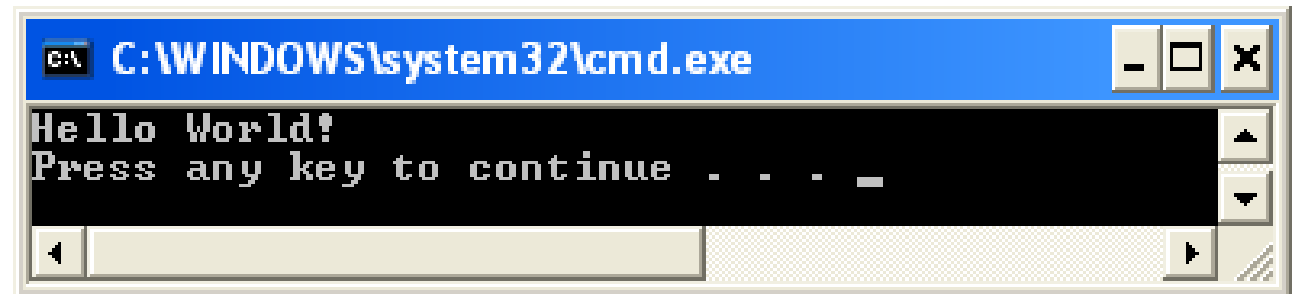
- Ứng dụng Web (Web application)
 - Tạo ra các nội dung động trên server thay cho trên trình duyệt.
 - Chạy trong các ứng dụng server
 - Servlet: Kiểm soát các yêu cầu từ trình duyệt và trả lại các phản hồi
 - JavaServer Page (JSP): Các trang HTML được nhúng với mã Java.

Nội dung

1. Công nghệ hướng đối tượng
2. Các nguyên lý cơ bản của OO
3. Ngôn ngữ lập trình Java
- 4. Ví dụ và bài tập

Ví dụ 1 - HelloWorld

```
// HelloWorld.java
// Chương trình hiển thị dòng chữ "Hello World"
public class HelloWorld {
    /* Phương thức main sẽ được gọi đầu tiên
       trong bất kỳ ứng dụng Java nào */
    public static void main(String args[]) {
        System.out.println( "Hello World!" );
    } // kết thúc phương thức main
} // kết thúc lớp HelloWorld
```



Ví dụ 1 (tiếp)

- Chú thích (Comment)
 - Trên 1 dòng: Bắt đầu bằng: //
 - Nhiều dòng: /* ... */
- Java phân biệt chữ hoa chữ thường
- Từ khóa có sẵn của Java:
 - class: Khai báo lớp
 - public: Quy định phạm vi truy cập
- Tên lớp chứa hàm main phải trùng với tên file .java.

Cài đặt và chạy thử chương trình Java

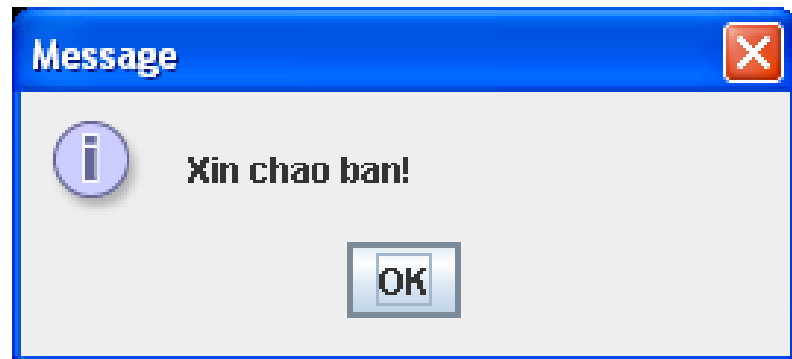
- Bước 1: Cài đặt j2sdk1.5, cài đặt các biến môi trường (nếu dùng cmd)
- Bước 2: Cài trình soạn thảo TextPad/JCreator
- Bước 3: Lập trình/Viết mã nguồn
- Bước 4: Dịch
 - cmd: `javac HelloWorld.java`
 - Textpad: `Ctrl + 1`
 - JCreator: `F7` hoặc `Build` → `Build Project/File`
- Bước 5: Chạy chương trình
 - cmd: `java HelloWorld.class`
 - Textpad: `Ctrl + 2`
 - JCreator: `F5` hoặc `Run` → `Run Project/File`

Biến môi trường

- `PATH = ...;C:\Program Files\Java\jdk1.6\bin`
- `CLASSPATH = C:\Program Files\Java\jdk1.6\lib;.;C:\Program Files\Java\jdk1.6\include`

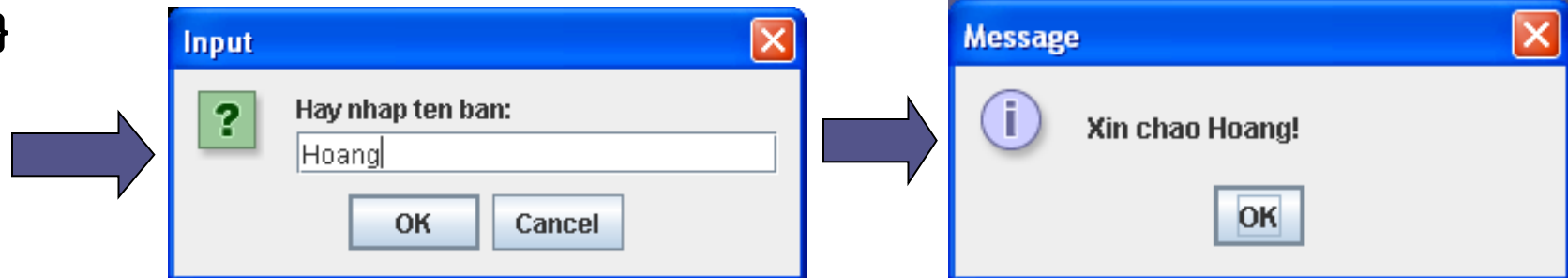
Ví dụ 2 - GUI

```
import javax.swing.JOptionPane;  
public class FirstDialog{  
    public static void main(String[] args){  
        JOptionPane.showMessageDialog(null,  
            "Xin chao ban!");  
        System.exit(0);  
    }  
}
```



Ví dụ 3 - Nhập, xuất dữ liệu

```
import javax.swing.JOptionPane;  
public class HelloNameDialog{  
    public static void main(String[] args){  
        String result;  
        result = JOptionPane.showInputDialog("Hay nhap  
                                                ten ban:");  
        JOptionPane.showMessageDialog(null,  
                                      "Xin chao "+ result + "!");  
        System.exit(0);  
    }  
}
```



Ví dụ về lớp và đối tượng trong Java

Class declaration: **each class** is, by default, an **extension of Object** (can be omitted)

Class constructor: **initialises** the various **fields**

Class method: **retrieves** and/or **modifies** the **state** of the class

```
public class Time extends Object {
    private int hour;
    private int minute;
    private int second;

    public Time () {
        setTime(0, 0, 0);
    }

    public void setTime (int h, int m, int s) {
        hour = ( ( h >= 0 && h < 24 ) ? h : 0 );
        minute = ( ( m >= 0 && m < 60 ) ? m : 0 );
        second = ( ( s >= 0 && s < 60 ) ? s : 0 );
    }
}
```

Class fields: **private** means they **can not be accessed** from **outside** the class

Java: Chương trình và các đối tượng

```
public class Test {  
  
    public static void main (String args[]) {  
        Time time = new Time();  
  
        time.hour = 7;  
        time.minute = 15;  
        time.second = 30;  
    }  
}
```

```
Test.java:6: hour has private access in Time  
        time.hour = 7;  
            ^
```

```
Time.java:7: minute has private access in Time  
        time.minute = 15;  
            ^
```

```
Time.java:8: second has private access in Time  
        time.second = 30;  
            ^
```

3 errors

Câu hỏi nhanh:

- Sự khác nhau cơ bản giữa lập trình cấu trúc và lập trình hướng đối tượng?

Lập trình cấu trúc - Lập trình hướng đối tượng

- Lập trình cấu trúc (procedural Programming):
 - Đơn vị là các thủ tục, hàm
 - Dữ liệu có ranh giới nhất định với thủ tục
- Lập trình Hướng đối tượng
 - Đơn vị là đối tượng
 - Dữ liệu gắn với hàm (phương thức) trong đối tượng
 - Mỗi cấu trúc dữ liệu có các phương thức hoạt động trên nó

Thảo luận

- Lập trình hướng đối tượng giải quyết tốt hơn những vấn đề nào của lập trình cấu trúc?
- Đọc tài liệu, đưa ra ví dụ minh họa.

Bài tập

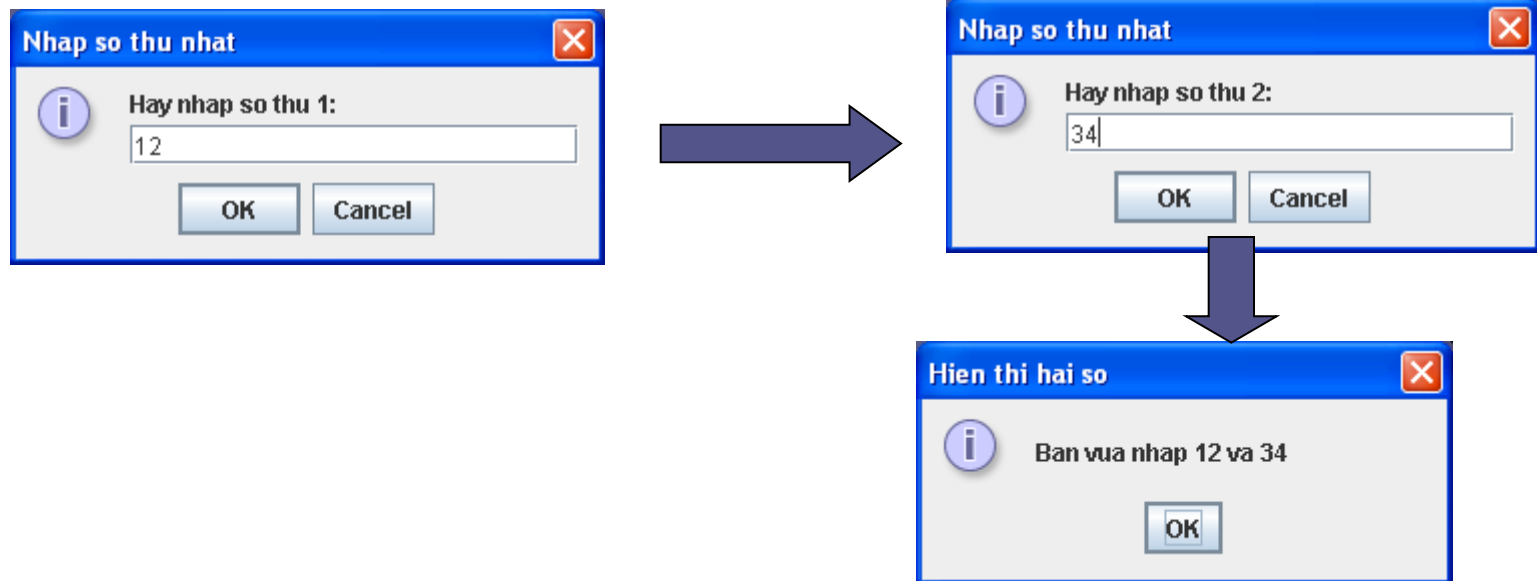
1. Chạy thử các ví dụ trong bài học trên Textpad và Jcreator/Netbean/Eclipse.
2. Viết chương trình in ra màn hình 2 chữ cái đầu của tên bạn. Ví dụ tên Thủy → In ra 2 chữ TH:

TTTTTTTTT	H	H
T	H	H
T	HHHHHHHH	
T	H	H
T	H	H

3. Tìm hiểu một câu chuyện ngắn liên quan đến lịch sử ra đời của Java (tên người sáng lập ra Java,...)

Bài tập (2)

4. Viết chương trình nhập hai số nguyên và hiển thị 2 số nguyên vừa nhập.



5. Sửa chương trình trên, hiển thị tổng, hiệu, tích thương của 2 số vừa nhập

Links download tài liệu

- <http://fit.hut.edu.vn/~dungct/Courses/OOP/K52/Huongdan-BTL-LTHDT-K52.doc>
- <http://fit.hut.edu.vn/~dungct/Courses/OOP/K52/Danh%20sach%20BTL-LTHDT-K52.doc>