NGÔN NGỮ LẬP TRÌNH JAVA

Nội dung

Cơ bản về ngôn ngữ lập trình Java

- Lập trình hướng đối tượng
- Biến, từ khoá, kiểu dữ liệu
- Biểu thức, các cấu trúc điều khiển
- Dữ liệu kiểu mảng

Các khía cạnh nâng cao của lập trình hướng đối tượng

- Thiết kế lớp
- Thiết kế lớp nâng cao
- Xử lý ngoại lệ

Xây dựng ứng dụng Java

- Tạo giao diện đồ hoạ
- Xử lý các sự kiện trên giao diện đồ hoạ
- Xây dựng ứng dụng đồ hoạ

Lập trình Java nâng cao

- Luồng
- Vào / Ra
- Lập trình mạng
- Lập trình với CSDL

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Nội dung

- *Định nghĩa các khái niệm mô hình hoá: abstraction, encapsulation, packages*
- *Tại sao có thể sử dụng lại mã Java trong nhiều ứng dụng*
- *Định nghĩa class, member, attribute, method, constructor, package*
- *Sử dụng các điều khiển truy cập private và public phù hợp với ngữ cảnh*
- *Triệu gọi các phương thức trong một đối tượng*
- *Sử dụng tài liệu lập trình Java*

Các vấn đề

- Bạn hiểu gì về quá trình phân tích và thiết kế phần mềm?
- Bạn hiểu gì về việc thiết kế và sử dụng lại code?
- Các khía cạnh, đặc điểm nào của Java khiến nó trở thành một ngôn ngữ lập trình hướng đối tượng?
- Lập trình hướng đối tượng là gì?

Lịch sử công nghệ phần mềm

Toolkits / Frameworks / Object APIs (1990s–Up)				
Java 2 SDK	AWT / J.F.C./Swing	Jini™	JavaBeans™	JDBC™

Object-Oriented Languages (1980s–Up)					
SELF	Smalltalk	Common Lisp Object System	Eiffel	C++	Java

Libraries / Functional APIs (1960s–Early 1980s)				
NASTRAN	TCP/IP	ISAM	X-Windows	OpenLook

High-Level Languages (1950s–Up)				Operating Systems (1960s–Up)			
Fortran	LISP	C	COBOL	OS/360	UNIX	MacOS	Microsoft Windows

Machine Code (Late 1940s–Up)

2 pha: phân tích - thiết kế

- Quá trình phân tích mô tả xem hệ thống cần phải làm gì?

Trong đó, nó sẽ mô hình hoá thế giới thực, đưa vào đó các actors (người tác động vào hệ thống), các hoạt động, các đối tượng và hành vi của các đối tượng

- Quá trình thiết kế chỉ ra rằng hệ thống sẽ thực hiện công việc đó như thế nào?
  - *Mô hình hoá các mối quan hệ, sự tương tác giữa các đối tượng và người sử dụng hệ thống*
  - *Tìm ra các đặc điểm trừu tượng hoá nhằm giúp cho việc đơn giản hoá bài toán.*

Class - Object

- Trong sản xuất, người ta làm 1 cái phôi để mô tả về 1 thiết bị, sau đó sẽ sản xuất ra các thiết bị thật
- Trong công nghiệp phần mềm, class chính là cái phôi, nó mô tả về các object:
  - *class mô tả dữ liệu chứa trong các object*
  - *class mô tả hành vi mà các object thực hiện*
- Trong Java, class hỗ trợ 3 đặc điểm của lập trình hướng đối tượng
  - *Encapsulation – đóng gói dữ liệu*
  - *Inheritance - sự thừa kế*
  - *Polymorphism – tính đa hình*

Khai báo lớp

- Cú pháp khai báo lớp:

```
<modifier>* class <class_name> {
        <attribute_declaration>*
        <constructor_declaration>*
        <method_declaration>*
}
```

- Ví dụ

```
1 public class Vehicle {
2       private double maxLoad;
3       public void setMaxLoad(double value) {
4            maxLoad = value;
5       }
6 }
```

Khai báo các thuộc tính

- Cú pháp khai báo thuộc tính:

<modifier>* <type> <name> [= <initial_value>];

- Ví dụ

```
1 public class Foo {
2       private int x;
3       private float y = 100.0F;
4       private String name = "Bates Motel";
6 }
```

Khai báo các phương thức

- Cú pháp khai báo phương thức:

<modifier>* <return_type> <name> **(** ***<argument>**** **) {**

<statement>*

}

- Ví dụ

```
1 public class Dog {
2     private int weight;
3     public int getWeight() {
4         return weight;
5     }
6     public void setWeight(int newWeight) {
7         if ( newWeight > 0 ) {
8             weight = newWeight;
9         }
10    }
11 }
```

Truy cập các thành phần của class

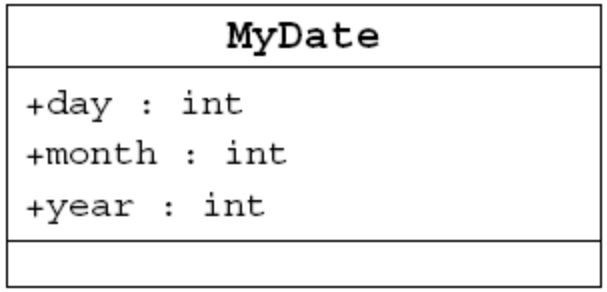
- Cú pháp truy cập: ***<object>.<member>***
- Ví dụ

```
1 d.setWeight(42);
2 d.weight = 42;
```

Che dấu dữ liệu

❑ Vấn đề: client code có thể truy cập trực tiếp vào dữ liệu bên trong của đối tượng (d refers to a MyDate object):

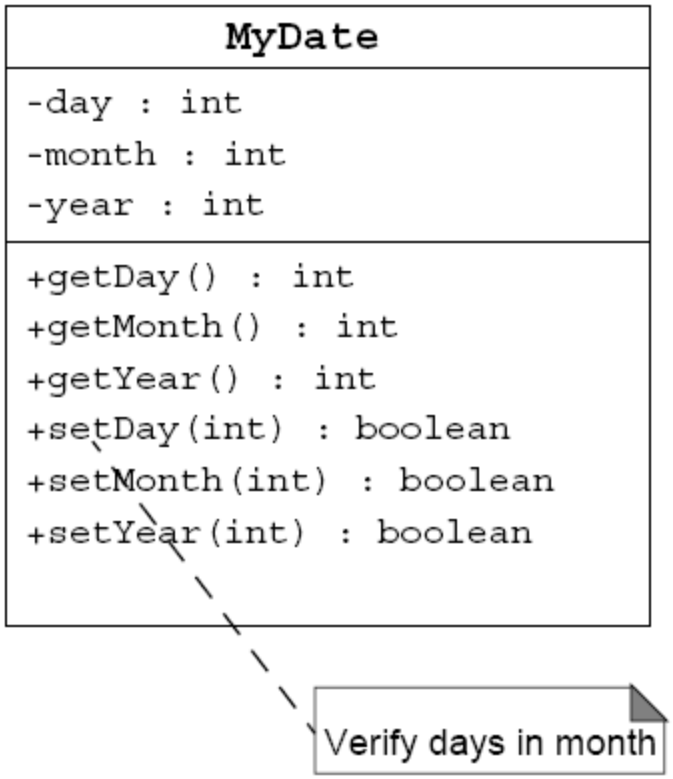
```
d.day = 32;
// invalid day
d.month = 2; d.day = 30;
// plausible but wrong
d.day = d.day + 1;
// no check for wrap around
```



Che dấu dữ liệu

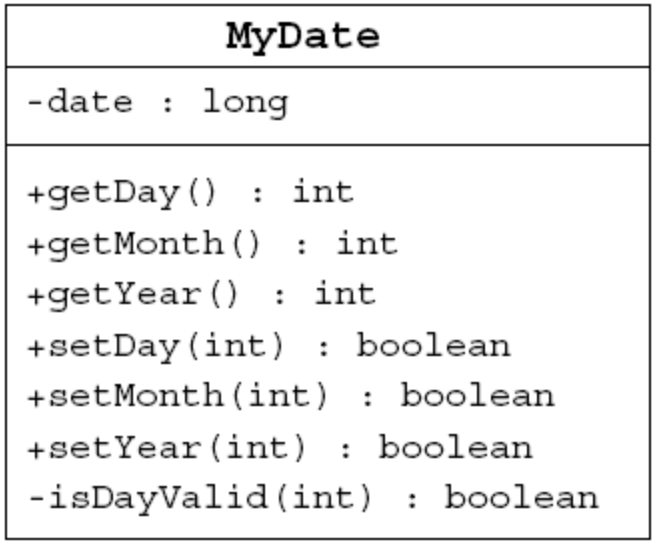
- Giải pháp: client code phải sử dụng các phương thức setters, getters để truy cập vào các dữ liệu bên trong của đối tượng

```
MyDate d = new MyDate();
d.setDay(32);
// invalid day, returns false
d.setMonth(2);
d.setDay(30);
// plausible but wrong,
// setDay returns false
d.setDay(d.getDay() + 1);
// this will return false if wra
// needs to occur
```



Đóng gói dữ liệu

- Che dấu các thành phần cài đặt bên trong của một lớp
- Yêu cầu người sử dụng phải dùng các interface để giao tiếp và truy cập vào các dữ liệu bên trong của lớp
- Giúp cho công việc bảo trì code nhanh chóng hơn



Hàm khởi tạo

- Cú pháp khai báo hàm khởi tạo:

```
[<modifier>] <class_name> ( <argument>* ) {
        <statement>*
}
```

- Ví dụ

```
1 public class Dog {
2       private int weight;
3       public Dog() {
4           weight = 42;
5       }
6 }
```

Hàm khởi tạo mặc định

- Mỗi lớp của Java luôn có ít nhất một hàm khởi tạo
- Nếu người lập trình viên không đưa vào lớp một hàm khởi tạo nào, trình biên dịch sẽ tự động đưa vào lớp một hàm khởi tạo mặc định:
  - Hàm khởi tạo mặc định có các đặc điểm của hàm khởi tạo
  - Nó không có tham số
  - Có modifier trùng với modifier của lớp
  - Lệnh đầu tiên trong thân của default constructor là lời gọi tới hàm khởi tạo không tham số của lớp cha

Câu lệnh package

- Cú pháp:

package <top_pkg_name>[.<sub_pkg_name>]*;

- Ví dụ

```
package shipping.gui.reportscreens;
```

- Phải đặt lời khai báo `package` ở dòng đầu tiên của source file
- Chỉ có 1 khai báo `package` trong 1 file
- Nếu không có lời khai báo `package` nào, class file sẽ được đặt ở thư mục mặc định
- Tên của `package` cần được phân cấp, chia cách bởi dấu (.)

Câu lệnh import

- Cú pháp:

import ***<pkg_name>[.<sub_pkg_name>]*.<class_name>;***

OR

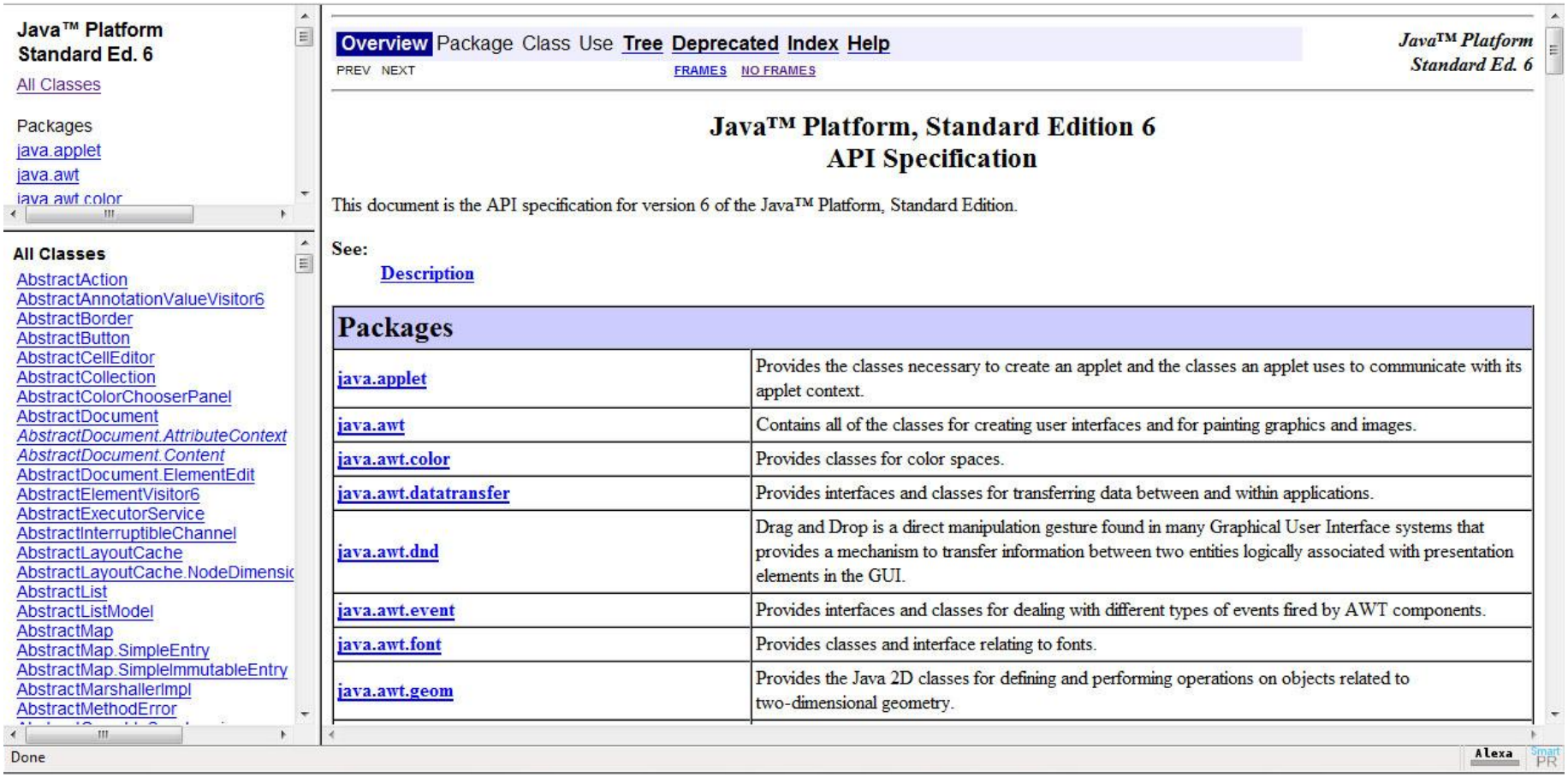
import ***<pkg_name>[.<sub_pkg_name>]*.*;***

- Ví dụ

```
import java.util.List;
import java.io.*;
import shipping.gui.reportscreens.*;
```

- Câu lệnh import phải đứng trước lời khai báo lớp
- Phải chỉ rõ vị trí file cần import

Sử dụng Java API Documentation



BIẾN, TỪ KHOÁ, KIỂU DỮ LIỆU

Nội dung

- *Cách đặt tên định danh trong Java*
- *Bộ từ khoá Java*
- *8 kiểu dữ liệu gốc trong Java*
- *Khoảng giá trị lưu trữ của các kiểu dữ liệu gốc*
- *Định nghĩa biến có kiểu dữ liệu gốc và biến tham chiếu*
- *Khai báo kiểu dữ liệu cho biến*
- *Tạo mới 1 đối tượng bằng từ khoá new*
- *Khởi tạo giá trị mặc định cho biến*

Cách đặt tên định danh

Một số đặc điểm khi đặt tên định danh trong Java:

- Là tên được đặt cho biến, phương thức, lớp
- Tên định danh có thể bắt đầu bằng một ký tự Unicode, dấu gạch dưới (_) hay dấu ($)
- Tên định danh trong Java có phân biệt chữ hoa, chữ thường và không giới hạn về kích thước
- *Ví dụ:*

```
identifier
userName
user_name
_sys_var1
$change
```

Bộ từ khoá Java

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

8 kiểu dữ liệu gốc

- Kiểu logic - `boolean`
- Kiểu ký tự - `char`
- Kiểu số nguyên - `byte`, `short`, `int`, `long`
- Kiểu số thực - `float`, `double`

Kiểu logic

- Chỉ có 2 giá trị: `true`, `false`
- Ví dụ: `boolean truth = true;`
- Giá trị mặc định là kiểu logic là `false`

Kiểu ký tự

- Thể hiện một ký tự Unicode 16-bit
- Ký tự phải được đặt trong dấu nháy đơn
- Ví dụ:

`'a'`	ký tự a
`'\t'`	ký tự tab
`'\u????'`	Biểu diễn ký tự Unicode, ???? là 4 con số ở hệ cơ số 16. Ví dụ: '\u03A6' biểu diễn ký tự phi [ϕ]

Kiểu số nguyên

- Có 3 dạng biểu diễn: hệ cơ số 8, hệ cơ số 10, hệ cơ số 16

`2`	Số 2 ở hệ cơ số 10
`077`	Ký tự đầu tiên là 0 → biểu diễn số ở hệ cơ số 8
`0xBAAC`	0x → biểu diễn số ở hệ cơ số 16

- Kiểu mặc định của kiểu nguyên là `int`
- Hậu tố L (l) ám chỉ rằng đó là kiểu `long`

Kiểu số nguyên – byte, short, int, long

Integer Length	Name or Type	Range
8 bits	`byte`	-2^{7} to $2^{7}-1$
16 bits	`short`	-2^{15} to $2^{15}-1$
32 bits	`int`	-2^{31} to $2^{31}-1$
64 bits	`long`	-2^{63} to $2^{63}-1$

Kiểu số thực – float, double

- Có 3 dạng biểu diễn:

`E or e` biểu diễn số thực lớn

`F or f` biểu diễn số ở dạng float

`D or d` biểu diễn số ở dạng double

- Ví dụ

`3.14` số thực ở dạng double

`6.02E23` số thực lớn ở dạng double

`2.718F` số thực ở dạng float

`123.4E+306D` số thực lớn ở dạng double (thừa hậu tố D)

Kiểu số thực – float, double

- Kiểu mặc định của kiểu số thực là double
- Kích thước của các kiểu dữ liệu:

Float Length	Name or Type
32 bits	`float`
64 bits	`double`

Kiểu dữ liệu tham chiếu

- Trong Java, ngoài 8 kiểu dữ liệu gốc, tất cả các kiểu dữ liệu khác được gọi là kiểu dữ liệu tham chiếu
- Biến có kiểu dữ liệu tham chiếu gọi là biến tham chiếu
- Ví dụ

```
1 public class MyDate {
2    private int day = 1;
3    private int month = 1;
4    private int year = 2000;
5    public MyDate(int day, int month, int year) { ... }
6    public String toString() { ... }
7 }
```

```
1 public class TestMyDate {
2    public static void main(String[] args) {
3       MyDate today = new MyDate(22, 7, 1964);
4    }
5 }
```

Khởi tạo đối tượng

- Sử dụng lời gọi new Xyz() để khởi tạo đối tượng. Khi đó:
  - Bộ nhớ được cấp phát cho đối tượng
  - Thực thi khởi tạo giá trị cho các thuộc tính một cách tường minh
  - Thực thi hàm khởi tạo
  - Tham chiếu tới đối tượng được trả về bởi từ khoá new
- Biến tham chiếu trỏ tới đối tượng và lưu trữ địa chỉ của đối tượng
- Ví dụ

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

Khởi tạo đối tượng

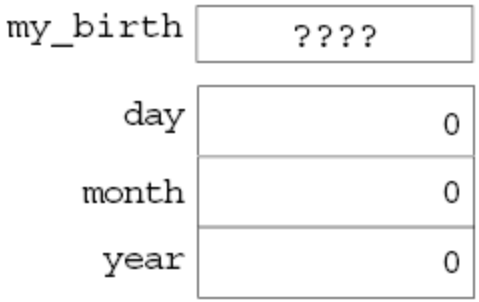
Memory Allocation and Layout

- A declaration allocates storage only for a reference:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

- Use the new operator to allocate space for MyDate:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

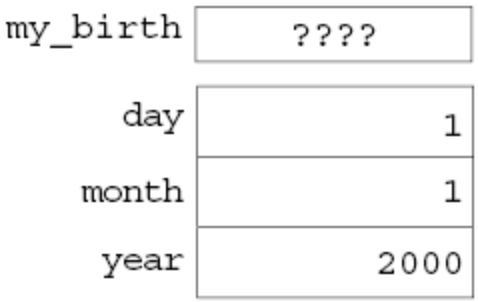


Khởi tạo đối tượng

Explicit Attribute Initialization

- Initialize the attributes as follows:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```



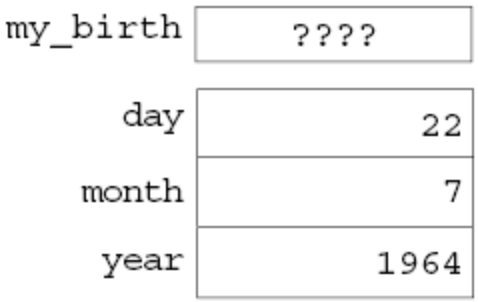
- The default values are taken from the attribute declaration in the class.

Khởi tạo đối tượng

Executing the Constructor

- Execute the matching constructor as follows:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```



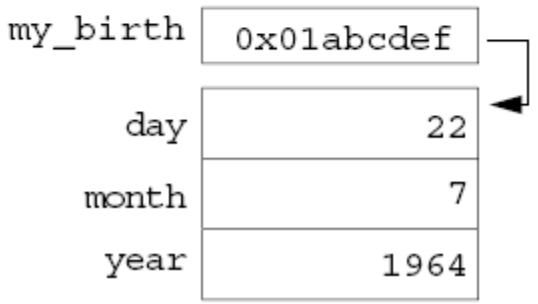
- In the case of an overloaded constructor, the first constructor can call another.

Khởi tạo đối tượng

Assigning a Variable

- Assign the newly created object to the reference variable as follows:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

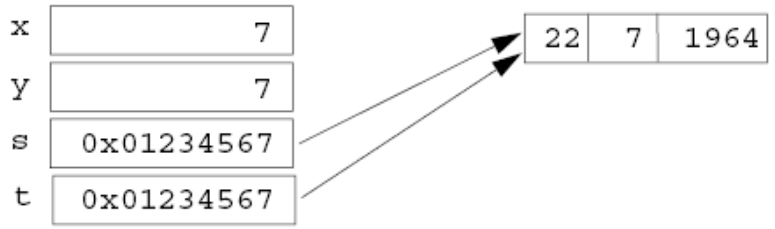


Khởi tạo đối tượng

Assigning References

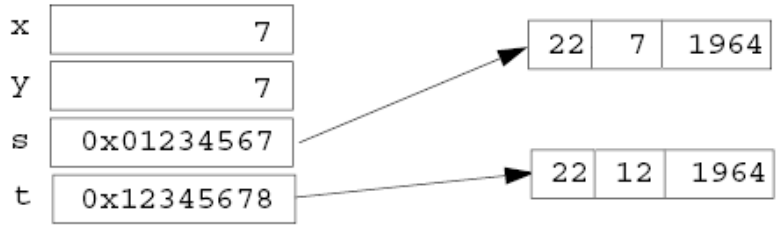
- Two variables refer to a single object:

```
1   int x = 7;
2   int y = x;
3   MyDate s = new MyDate(22, 7, 1964);
4   MyDate t = s;
```



- Reassignment makes two variables point to two objects:

```
5   t = new MyDate(22, 12, 1964);
```



Tham chiếu this

- Tham chiếu this trỏ tới chính lớp hiện tại
- Tham chiếu this giúp phân biệt tham số của phương thức với các thuộc tính
- Tham chiếu this cũng được dùng để truyền tham số

Tham chiếu this

```
1 public class MyDate {
2    private int day = 1;
3    private int month = 1;
4    private int year = 2000;
5
6    public MyDate(int day, int month, int year) {
7       this.day = day;
8       this.month = month;
9       this.year = year;
10   }
11   public MyDate(MyDate date) {
12      this.day = date.day;
13      this.month = date.month;
14      this.year = date.year;
15   }
```

Tham chiếu this

```
17    public MyDate addDays(int moreDays) {
18       MyDate newDate = new MyDate(this);
19       newDate.day = newDate.day + moreDays;
20       // Not Yet Implemented: wrap around code...
21       return newDate;
22    }
23    public String toString() {
24       return "" + day + "-" + month + "-" + year;
25    }
26 }
```

```
1 public class TestMyDate {
2    public static void main(String[] args) {
3       MyDate my_birth = new MyDate(22, 7, 1964);
4       MyDate the_next_week = my_birth.addDays(7);
5       System.out.println(the_next_week);
7    }
8 }
```

Một số quy ước khi lập trình Java

- Packages:

```
com.example.domain;
```

- Classes, interfaces, and enum types:

```
SavingsAccount
```

- Methods:

```
getAccount()
```

- Variables:

```
currentCustomer
```

- Constants:

```
HEAD_COUNT
```