

Hiểu thêm về Java

(Dựa trên slide của PGS. TS. Nguyễn Việt Hà)

Nội dung

- ▶ Dữ liệu kiểu nguyên thủy và đối tượng
- ▶ Tham chiếu
- ▶ Giải phóng bộ nhớ
- ▶ Gói và kiểm soát truy cập
- ▶ Kiểu hợp thành (composition)
- ▶ Vào ra với luồng dữ liệu chuẩn

Tài liệu tham khảo

- ▶ *Giáo trình Lập trình HĐT*, chương 2, 3, 4

Kiểu dữ liệu nguyên thủy

- ▶ Java cung cấp các kiểu nguyên thủy
 - ▶ số: byte, short, int, long, float, double
 - ▶ không có khái niệm unsigned
 - ▶ kích thước cố định trên mọi platform
 - ▶ logic: boolean
 - ▶ ký tự: char
- ▶ Dữ liệu kiểu nguyên thủy ***không phải là đối tượng***
 - ▶ `int a = 5;`
 - ▶ `int b = 5;`
 - ▶ `if (a==b)...`
- ▶ Tồn tại lớp đối tượng tương ứng(Wrapper class): Integer, Float,..
 - ▶ `Integer count = new Integer(0);`
 - ▶ Xem thêm: <http://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html>

Wrapper class và Autoboxing

▶ Wrapper class

- ▶ byte -> **Byte**
- ▶ short -> **Short**,
- ▶ int -> **Integer**,
- ▶ long -> **Long**,
- ▶ float -> **Float**,
- ▶ double -> **Double**
- ▶ boolean -> **Boolean**
- ▶ char -> **Character**

▶ Autoboxing

- ▶ Boxing: tự chuyển từ kiểu dữ liệu nguyên thủy thành kiểu đối tượng
- ▶ Unboxing: tự chuyển từ kiểu dữ liệu đối tượng sang nguyên thủy

Ví dụ: Boxing và Unboxing

```
int x = 10;  
Integer y = new Integer(10);
```

```
int z = x + y; // autoboxing, cu the la unboxing, tuc la chuyen doi tu Integer ve int  
System.out.println(z);
```

```
int z1 = Integer.valueOf("45");  
int z2 = Integer.parseInt("45");  
Long.valueOf("56");  
Boolean.valueOf("true");
```

```
Integer k = 10; // thi o day, boxing, tu dong tu int thanh Integer  
k = new Integer(10);
```

```
int x1 = 10; int x2 = 10;  
Integer x3 = 10;
```

```
System.out.println(x1 == x2);  
System.out.println(x1 == x3);
```

```
Integer k1 = new Integer(10);  
Integer k2 = new Integer(10);  
int k3 = 10;
```

```
System.out.println(k1 == k2);  
System.out.println(k1 == k3);  
System.out.println(k1.equals(k2));
```

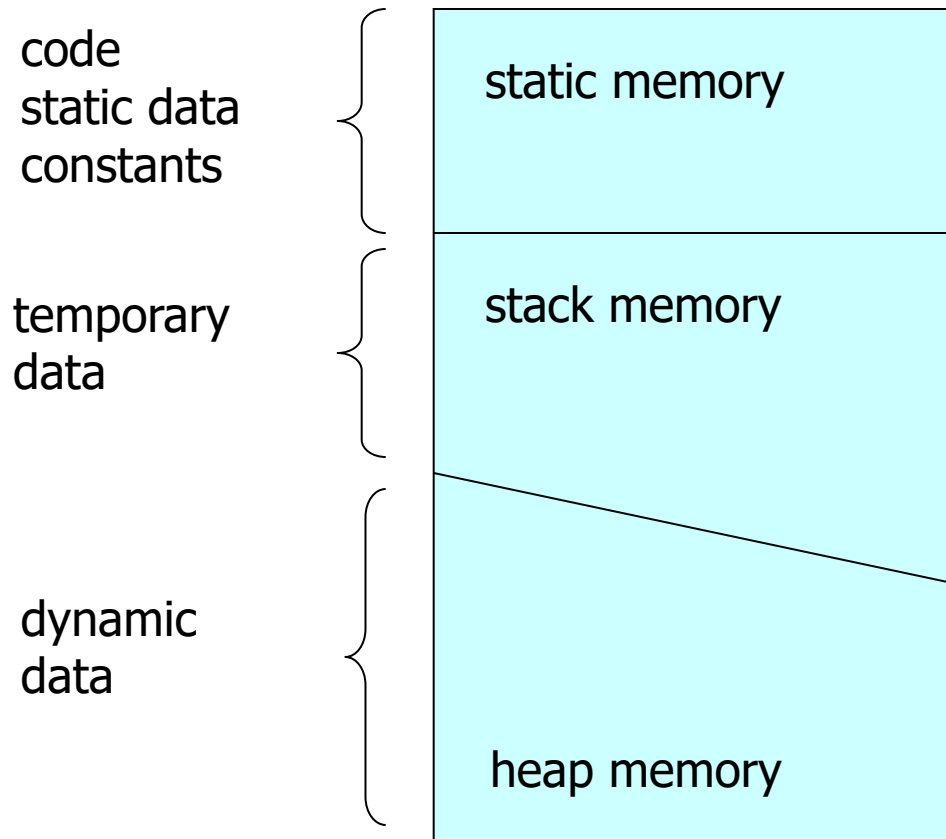
Kiểu dữ liệu	Kích thước (bits)	Giá trị cực tiểu	Giá trị cực đại
char	16	0x0	0xffff
byte	8	-128 (-2^7)	+127 (2^7-1)
short	16	-32768 (-2^{15})	32767 ($2^{15}-1$)
int	32	-2^{31} , 0x80000000	$+2^{31}-1$, 0x7fffffff
long	64	-2^{63}	$+2^{63}-1$
float	32	1.40129846432481707e-45	3.40282346638528860e+38
double	64	4.94065645841246544e-324	1.79769313486231570e+308
boolean	1	true; false	



Dữ liệu được lưu trữ ở đâu

- ▶ Dữ liệu kiểu nguyên thủy
 - ▶ Thao tác thông qua *tên biến*
- ▶ Dữ liệu được đóng gói (là thuộc tính) trong đối tượng
 - ▶ Đối tượng được thao tác thông qua tham chiếu
- ▶ Vậy biến kiểu nguyên thủy, tham chiếu và đối tượng được lưu trữ ở đâu?

3 vùng bộ nhớ cho ứng dụng



Tham chiếu

- ▶ Đối tượng được thao tác thông qua tham chiếu
 - ▶ là *con trỏ* tới đối tượng
 - ▶ thao tác trực tiếp tới thuộc tính và phương thức
 - ▶ không có các toán tử con trỏ
 - ▶ phép gán (=) không phải là phép toán copy nội dung đối tượng
- ▶ tham chiếu được lưu trữ trong vùng nhớ static/stack như các con trỏ trong C/C++

Toán tử New

- ▶ Phải tạo mọi đối tượng một cách tường minh bằng toán tử new
 - ▶ cấp phát vùng nhớ động
 - ▶ được tạo trong bộ nhớ Heap
- ▶ Ví dụ:
`MyDate d;`
`d = new MyDate();`

Phép gán "="

- ▶ Phép gán không phải là copy thông thường
 - ▶ copy nội dung của tham chiếu
 - ▶ 2 tham chiếu sẽ tham chiếu đến cùng đối tượng

```
Integer m = new Integer(10);
```

```
Integer n = new Integer(20);
```

```
m = n;
```

```
n.setValue(50);
```

```
System.out.print(m);
```

```
int m=10, n=20;
```

```
m=n;
```

```
n=50;
```

```
System.out.print(m);
```

"New" và "="

```
MyDate d;
```

```
MyDate birthday;
```

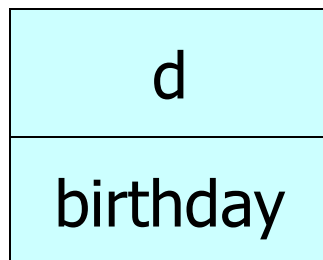
```
d = new MyDate(26, 9, 2005);
```

```
birthday = d;
```

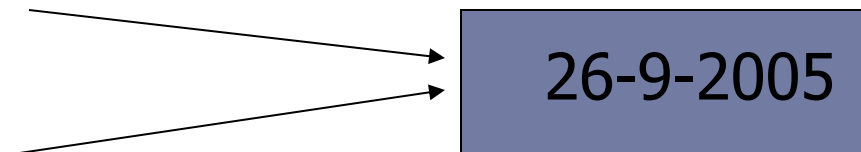
new operation

assign operation

Static/Stack memory



Heap memory



Toán tử quan hệ "=="

- ▶ So sánh nội dung của các dữ liệu kiểu nguyên thủy (int, long, float, ...)
- ▶ So sánh nội dung của tham chiếu chứ không so sánh nội dung của đối tượng do tham chiếu trỏ đến

```
Integer n1 = new Integer(47) ;  
Integer n2 = new Integer(47) ;  
Integer n3 = n2 ;  
System.out.println(n1 == n2) ; //???  
System.out.println(n2 == n3) ; //???
```

So sánh nội dung đối tượng

```
class MyDate {  
    ...  
    boolean equalTo(MyDate d) {  
        if (this.d==d&&this.m==m&&this.y==y)  
Return true;  
        }  
    }  
    ...  
MyDate d1 = new MyDate(10,10,1954) ;  
MyDate d2 = new MyDate(d1) ;  
MyDate d3 = new MyDate(10,10,1954) ;  
System.out.println(d1.equalTo(d2)) ;  
System.out.println(d1.equalTo(d3)) ;
```

Thu hồi bộ nhớ động (Garbage collection)

- ▶ Lập trình viên không cần phải giải phóng đối tượng
- ▶ JVM cài đặt cơ chế “Garbage collection” để thu hồi tự động các đối tượng không còn cần thiết
 - ▶ Tuy nhiên, *GC không nhất thiết hoạt động với mọi đối tượng* (không nhất thiết phải giải phóng bộ nhớ)
 - ▶ Không đảm bảo việc *phương thức hủy* luôn hoạt động
- ▶ GC tăng tốc độ phát triển và tăng tính ổn định của ứng dụng
 - ▶ Không phải viết mã giải phóng đối tượng
 - ▶ Do đó, *không bao giờ quên giải phóng đối tượng*

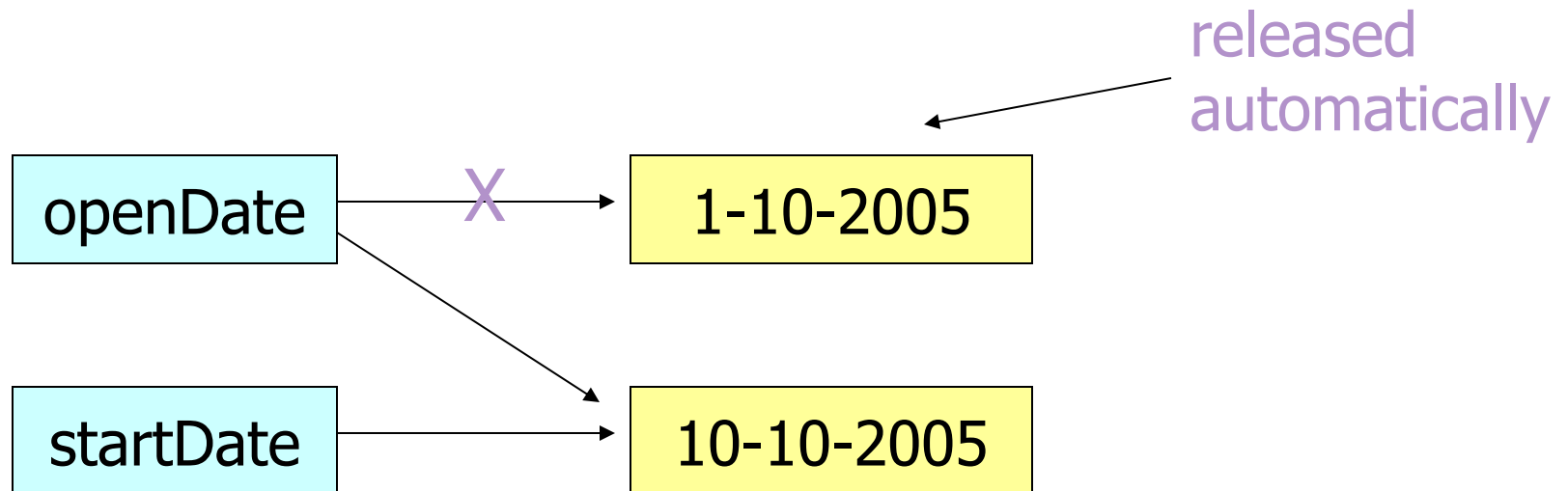
GC hoạt động như thế nào

▶ Sử dụng cơ chế đếm?

- ▶ mỗi đối tượng có một số đếm các tham chiếu trỏ tới
- ▶ giải phóng đối tượng khi số đếm = 0

Garbage Collection

```
MyDate openDate = new MyDate(1,10,2005);  
MyDate startDate = new MyDate(10,10,2005);  
openDate = startDate;
```



GC (cont.)

- ▶ Java cung cấp phương thức hủy `finalize()`
 - ▶ Không nhất thiết được thực hiện do hệ thống không phải luôn luôn thực hiện việc giải phóng bộ nhớ.
 - ▶ Ứng dụng `finalize()` trong debug chương trình: kiểm tra trạng thái cuối cùng của đối tượng.
- ▶ Có thể yêu cầu hệ thống giải phóng bộ nhớ bằng phương thức `System.gc()`

Truyền tham số và nhận giá trị trả lại

▶ Truyền giá trị

- ▶ đối với dữ liệu kiểu nguyên thủy
- ▶ giá trị của tham số được copy lên stack
- ▶ có thể truyền hằng số (vd: 10, 0.5, ...)

▶ Truyền tham chiếu

- ▶ đối với đối tượng
- ▶ nội dung của tham chiếu được copy lên stack

Truyền tham số trị

```
class MyDate {  
    ...  
    public boolean setYear(int y) {  
        ...  
    }  
    public int getYear() {  
        return year;  
    }  
    ...  
}  
...  
MyDate d = new MyDate();  
d.setYear(1975);  
int y=1975;  
d.setYear(y);
```

```
public boolean setYear(int y)  
{  
    y=0;  
}
```

Truyền tham chiếu

```
class MyDate {  
    int year, month, day;  
    public MyDate(int year, int m, int d) {  
        this.year = year; month = m; day = d;  
    }  
    public void copyTo(MyDate d) {  
        this.year = d.year;  
        month = d.month;  
        day = d.day;  
        d.day = 20;  
    }  
    public MyDate copy() {  
        return new MyDate(day, month, year);  
    }  
}
```

...



Truyền tham chiếu

```
MyDate d1 = new MyDate(2005, 9, 26);  
MyDate d2 = new MyDate(2000, 1, 1);  
d1.copyTo(d2);  
d2.getYear(); // ???  
MyDate d3 = new MyDate();  
d3 = d1.copy();
```

Bao nhiêu đối tượng thuộc lớp MyDate được tạo ra?

Tham chiếu `this`

- ▶ Java cung cấp tham chiếu `this` để trỏ tới chính đối tượng đang hoạt động
- ▶ `this` được sử dụng vào các mục đích như
 - ▶ tham chiếu tường minh đến thuộc tính và phương thức của đối tượng
 - ▶ truyền tham số và trả lại giá trị
 - ▶ dùng để gọi constructor

this làm giá trị trả lại

```
class Counter {  
    private int c = 0;  
    public Counter increase() {  
        c++;  
        return this;  
    }  
    public int getValue() {  
        return c;  
    }  
}  
  
...  
Counter count = new Counter();  
System.out.println(count.increase().increase().getValue());
```

Kết quả in ra màn hình?

Bao nhiêu đối tượng thuộc lớp Counter được tạo ra?

this làm tham số

```
class MyDocument {
    MyWindow w;
    ...
    MyDocument(MyWindow mw) {
        w = mw;
        ...
    }
    void display() {
        w.display(this);
    }
}

...
}

class MyWindow {
    public void display(MyDocument doc) {
        //statements for displaying document
    }
}
```

Gọi constructor bằng `this`

```
class MyDate {  
    private int year, month, day;  
  
    public MyDate(int y, int m, int d) {  
        ...  
    }  
  
    ...  
    public MyDate(MyDate d) {  
        year=d.getYear();  
        month=d.getMonth();  
        ...  
    }  
    ...  
    }  
    MyDate(MyDate d) {  
        year=d.year;  
        month=d.month;  
        day=d.day;  
    }  
    MyDate(MyDate d) {  
        this(d.year, d.month, d.day)  
    }  
}
```

Constructor chỉ được gọi bên trong một constructor khác và chỉ được gọi một lần ở thời điểm (vị trí) đầu tiên.

Phương thức và thuộc tính static

- ▶ Có thể khai báo phương thức và thuộc tính là tĩnh (static)
 - ▶ Độc lập với đối tượng(có thể sử dụng mà **không cần có đối tượng**)
 - ▶ Khai báo dữ liệu dùng chung
- ▶ Phương thức tĩnh
 - ▶ không sử dụng được thuộc tính thông thường (non-static)
 - ▶ không gọi được các phương thức thông thường



```
class Hello{
    public static String message="Hello, world";
    public static void sayHello() {
        System.out.println(message);
    }
    public void sayHello(String msg){
        System.out.println(msg);
    }
}
```

```
public class HelloTestDrive{
    public static void main(String[] args){
        Hello.sayHello();
        System.out.println(Hello.message);

        Hello h=new Hello();
        h.sayHello();
        h.sayHello("Hello, world");
    }
}
```

```
public class Hello{
    static Hello h = new Hello();
    public void sayHello() {
        System.out.println("Hello, world");
    }
    static Hello getRef() { return h;}
    public static void main (String[] args) {
        h.sayHello();
        Hello h2 = new Hello();
        h2.sayHello();
        // sayHello();
        getRef().sayHello();
    }
}
```

Gói các lớp đối tượng (package)

- ▶ Các lớp đối tượng được chia thành các gói
 - ▶ nếu không khai báo thì các lớp thuộc gói default
 - ▶ các lớp trong cùng một tệp mã nguồn luôn thuộc cùng một gói
- ▶ Tồn tại mức truy cập package
 - ▶ mức package là mặc định (nếu không khai báo tường minh là public hay private)
 - ▶ các đối tượng của các lớp thuộc cùng gói có thể truy cập đến non-private members của nhau
 - ▶ chỉ có thể tạo (new) đối tượng của lớp được khai báo là public của gói khác

HelloTestDrive.java:

```
class Hello {  
    void sayHello() {  
        System.out.println("Hello, world!");  
    }  
}  
  
public class HelloTestDrive {  
    public static void main(String[] args) {  
        Hello h = new Hello();  
        h.sayHello();  
    }  
}
```


Khai báo và sử dụng package

- ▶ Khai báo gói bằng lệnh `package`
 - ▶ các gói được lưu trữ theo cấu trúc cây thư mục
 - ▶ sử dụng tham số `-d` để tạo thư mục khi biên dịch
- ▶ Dùng lệnh `import` để khai báo việc sử dụng một gói đã có

hieuvd/hello/Hello.java:

```
package hieuvd.hello;  
public class Hello {  
    public void sayHello() {  
        System.out.println("Hello, world!");  
    }  
}
```

hieuvd/test/HelloTestDrive.java:

```
import hieuvd.hello.Hello;

public class Hello {
    public static void main(String[] args) {
        Hello h = new Hello();
        h.sayHello();
    }
}
```

Biên dịch & thực hiện

- ▶ Biên dịch (tại thư mục chứa hieuvd)

```
javac hieuvd/test/HelloTestDrive.java
```

- ▶ Thực hiện

```
java hieuvd.test.HelloTestDrive
```

Biên dịch & thực hiện

- ▶ Tạo file jar (tại thư mục hieuvd)

(jar cf myJar.jar myClass.class)

```
jar -cf hieuvd.jar ./hieuvd
```

- ▶ Thực hiện

```
java -classpath path_to_hieuvd.jar  
hieuvd.test.HelloTestDrive
```

Hoặc thêm đường dẫn đến hieuvd.jar vào biến môi trường CLASSPATH

Đối tượng hợp thành (Composition)

- ▶ Đối tượng có thể chứa các đối tượng khác (các thuộc tính không thuộc kiểu nguyên thủy)
- ▶ Thuộc tính là tham chiếu phải được tạo ra bằng new hoặc được gán cho một đối tượng đã tồn tại

```
class Person {  
    private String name;  
    private MyDate birthday = new MyDate(1,1,2000) ;  
    ...  
}
```

Get/Set thuộc tính không thuộc kiểu nguyên thủy

```
class Person {  
...  
    public MyDate getBirthday() {  
        return birthday;  
    }  
}
```

```
Person p = new Person(...);  
MyDate d = p.getBirthday();  
d.setYear(1900);
```

Get/Set bằng copy constructor

```
class Person {  
    private String name;  
    private MyDate birthday;  
    public Person(String s, MyDate d) {  
        name = s;  
        birthday = new MyDate(d);  
    }  
    public MyDate getBirthday() {  
        return new MyDate(birthday);  
    }  
    public void setBirthday(MyDate d) {  
        birthday = new MyDate(d);  
    }  
    ...  
}
```

Cách nào đúng???

Vào ra từ luồng dữ liệu chuẩn

- ▶ Luồng ra chuẩn: `System.out`
 - ▶ xuất ra luồng ra chuẩn (standard output)
 - ▶ có thể tái định hướng
- ▶ Luồng thông báo lỗi: `System.err`
 - ▶ xuất ra **Console** (thiết bị output chuẩn)
 - ▶ không thể tái định hướng
- ▶ Luồng dữ liệu vào chuẩn: `System.in`
 - ▶ chưa sẵn sàng cho sử dụng

Nhập dữ liệu từ luồng vào chuẩn

- ▶ **InputStream**: lớp đối tượng ứng với luồng vào chuẩn
 - ▶ **System.in**: đối tượng tương ứng
 - ▶ chưa có phương thức nhập dữ liệu
- ▶ **Scanner**: nhập dữ liệu kiểu nguyên thủy và chuỗi ký tự
 - ▶ **next**: nhập chuỗi ký tự
 - ▶ **next *Type*** : nhập một dữ liệu kiểu *Type*
 - ▶ **hasNext, hasNext *Type*** : kiểm tra xem còn dữ liệu không.

Ví dụ

```
Scanner sc = new Scanner(System.in);
```

```
System.out.println(sc.next());
```

```
int i = sc.nextInt();
```

```
while (sc.hasNextLong()) {  
    long aLong = sc.nextLong();  
}
```

Tham số dòng lệnh

CmdLineParas.java:

```
public class CmdLineParas {  
    public static void main(String[] args) {  
        for (int i=0; i<args.length; i++)  
            System.out.println(args[i]);  
    }  
}
```

Ví dụ:

```
#java CmdLineParas hello world  
hello  
world
```



Bài tập: Cấu trúc dữ liệu

- ▶ **Xây dựng Danh sách liên kết đơn**
 - ▶ Thêm phần tử vào đầu/cuối danh sách
 - ▶ Bớt phần tử khỏi danh sách
 - ▶ Tìm phần tử theo khóa
- ▶ **Xây dựng Stack và Queue**
 - ▶ Thêm bớt phần tử
 - ▶ Kiểm tra đầy, rỗng
 - ▶ Duyệt, tìm kiếm