Bài 4:
Hiểu thêm về Java

Trường ĐH Công nghệ, ĐHQG Hà Nội

Tài liệu tham khảo

- *Giáo trình Lập trình HĐT*, Chương 3, 4
- *Java How to Program*, Chapters 3-8

Nội dung

- Dữ liệu kiểu nguyên thủy
- Tham chiếu
- Giải phóng bộ nhớ
- Truyền tham số
- Tham chiếu **this**

Các kiểu dữ liệu

- Java là ngôn ngữ định kiểu mạnh
  - Tất cả các biến đều phải có kiểu
- Kiểu dữ liệu nguyên thủy
  - Các biến được thao tác thông qua tên
    - int a = 5;
    - if (a == b)…
- Các kiểu tham chiếu (references)
  - Tham chiếu đến các đối tượng
  - Các đối tượng được thao tác qua các tham chiếu
    - GradeBook myGradeBook = new GradeBook();

Các kiểu dữ liệu nguyên thủy

- Các kiểu dữ liệu nguyên thủy trong Java
  - Số: byte, int, long, float, double
    - Không có kiểu dữ liệu "unsigned"
    - Kích thước của mỗi kiểu dữ liệu là giống nhau trên tất cả các hệ điều hành
  - Lôgic: boolean (true/false)
  - Ký tự: char
- Dữ liệu nguyên thủy KHÔNG PHẢI là các đối tượng
  - int count = 0;
  - if (count == 5) …
- Có các lớp tương ứng gói kiểu nguyên thủy (wrapper classes)
  - Integer, Float, …
  - Integer count = new Integer(0);

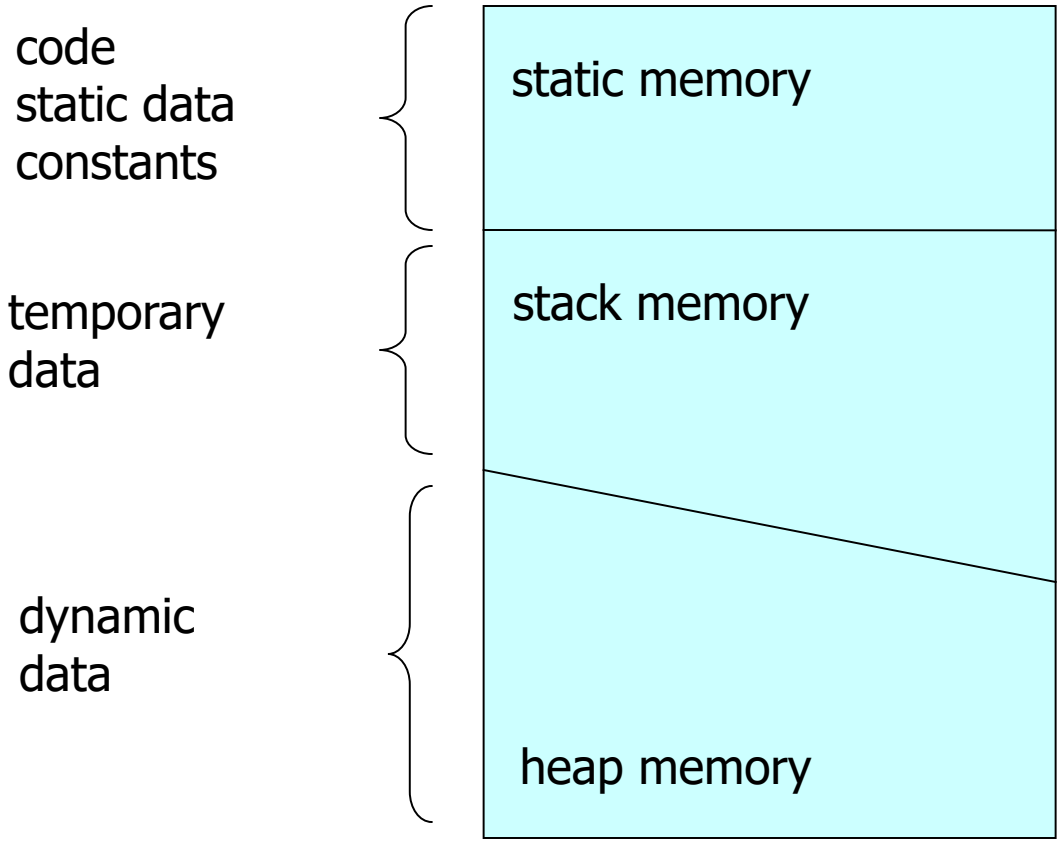
Các kiểu dữ liệu nguyên thủy

Kiểu	Kích cỡ	Giá trị cực tiểu	Giá trị cực đại	Lớp tương ứng
char	10 bits	0x0	0xffff	Character
byte	8 bits	-2^7	$2^7 - 1$	Byte
short	16 bits	-2^8	$2^8 - 1$	Short
int	32 bits	-2^{31}	$2^{31} - 1$	Integer
long	64 bits	-2^{63}	$2^{63} - 1$	Long
float	32 bits	1.401298464324 81707e-45	3.402823466385 28860e+38	Float
double	64 bits	4.940656458412 46544e-324	1.797693134862 31570e+308	Double
boolean	-			Boolean

Dữ liệu được lưu ở đâu?

- Dữ liệu nguyên thủy, các tham chiếu và các đối tượng được lưu ở đâu?
  - Registers: các lập trình viên không thể can thiệp
  - Stack: lưu các tham chiếu đối tượng
  - Heap: lưu các đối tượng
  - Vùng tĩnh (static storage): dữ liêu dùng trong suốt quá trình chương trình thực thi
  - Vùng hằng (constant storage): giá trị hằng thường được lưu trực tiếp trong phần mã nguồn chương trình
  - Vùng Non-RAM: lưu các luồng

Các vùng bộ nhớ cho ứng dụng



Các tham chiếu (references)

- Lưu "địa chỉ" đối tượng trong bộ nhớ máy tính
- Đối tượng được thao tác qua tham chiếu
  - Con trỏ đến đối tượng
  - Điều khiển trực tiếp các thuộc tính và phương thức
  - Không có các toán tử con trỏ
  - Phép gán (=) không sao chép nội dung đối tượng
- Được lưu trong vùng nhớ stack/static

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 "="

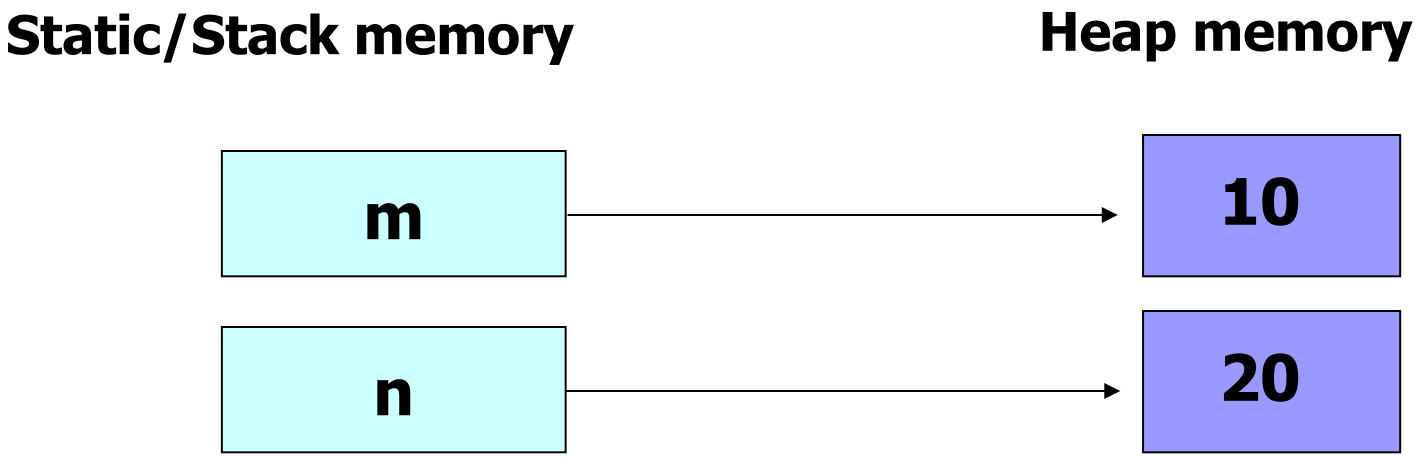
- Phép gán không sao chép như thông thường
  - sao chép nội dung của tham chiếu
  - 2 tham chiếu sẽ tham chiếu đến cùng đối tượng

```
Integer m  = new Integer(10);
Integer n  = new Integer(20);
m = n;
n.setValue(50);
System.out.print(m);
```

```
50
```

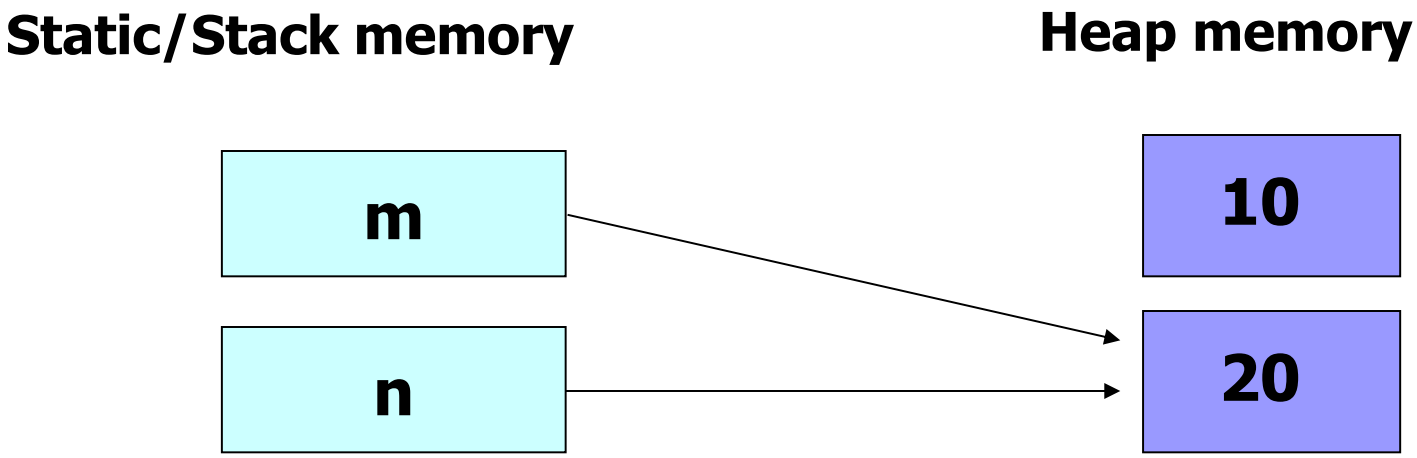
Ví dụ về "new" và "="

```
MyInteger m  = new MyInteger(10);
MyInteger n  = new MyInteger(20);
m = n;
n.setValue(50);
```



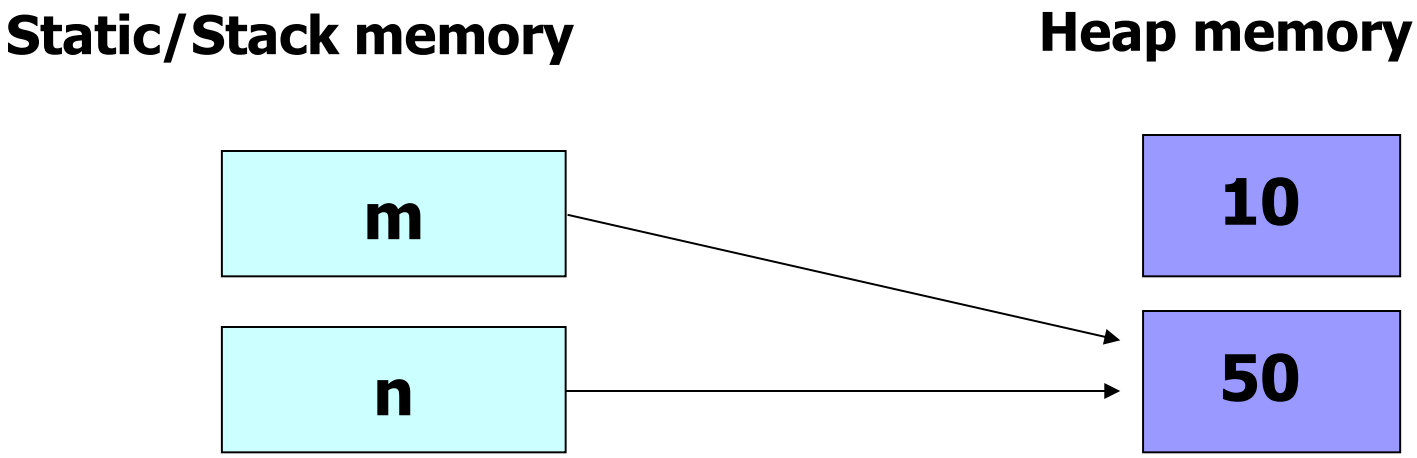
Ví dụ về "new" và "="

```
MyInteger m  = new MyInteger(10);
MyInteger n  = new MyInteger(20);
m = n;
n.setValue(50);
```



Ví dụ về "new" và "="

```
MyInteger m  = new MyInteger(10);
MyInteger n  = new MyInteger(20);
m = n;
n.setValue(50);
```



Toán tử "==" và "!="

- So sánh nội dung các biến
  - Giá trị của dữ liệu nguyên thủy
  - Giá trị của các tham chiếu
    - chỉ kiểm tra xem có tham chiếu đến cùng đối tượng
    - KHÔNG so sánh nội dung các đối tượng

```
MyInteger m1  = new MyInteger(10);
MyInteger m2  = new MyInteger(10);
System.out.println(m1 == m2);

int n1 = 1;
int n2 = 1;
System.out.println(n1 == n2);
```

```
false
true
```

So sánh nội dung đối tượng

- Phương thức "equals"
  - Được định nghĩa trước
    - Sẵn sàng để dùng
  - Do người dùng định nghĩa
    - equals() phải được định nghĩa, nếu không sẽ trả lại false
    - ghi đè / overriding (xem bài sau)

```
class MyInteger {
    private int value;
    public boolean equals (MyInteger other) {
        return (value == other.value);
    }
...
}
```

```
...
MyInteger m1 = new MyInteger(10);
MyInteger m2 = new MyInteger(10);
System.out.print(m1.equals(m2));
```

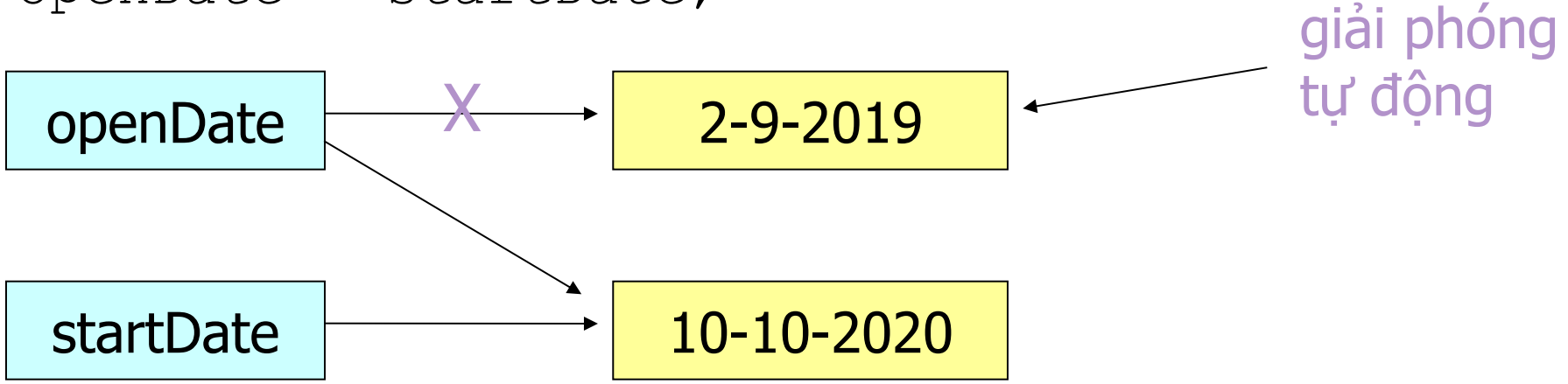
Thu hồi bộ nhớ (Garbage collection)

- 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 khi không còn cần thiết
  - 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 tham chiếu bằng 0

```
MyDate openDate = new MyDate(2,9,2019);
MyDate startDate = new MyDate(10,10,2020);
openDate = startDate;
```

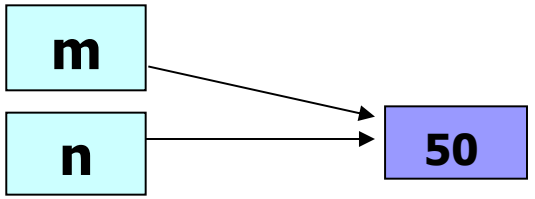


GC (tiếp)

- 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 về

- Truyền giá trị
  - Giá trị tham số được đưa vào stack
  - Là cơ chế duy nhất được dùng trong Java
    - Java không hỗ trợ truyền theo tham chiếu
- Hai loại tham số
  - Dữ liệu nguyên thủy
    - giá trị tham số được sao chép
    - các tham số có thể là hằng, ví dụ, 10, “abc”…
  - Tham chiếu đối tượng
    - tham chiếu (nơi lưu đối tượng) được sao chép
    - KHÔNG sao chép nội dung đối tượng



Truyền tham số và nhận giá trị trả về

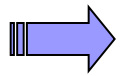
- Cách thức hoạt động
  - Tham số tương ứng với một biến địa phương để lưu giá trị truyền vào

Định nghĩa hàm

```
ReturnType foo(DataType v) {
   //processing v
}
```

Gọi hàm

```
...
foo(u); // u có kiểu DataType
```



tương đương

```
{
   DataType v = u;
   //processing v
}
```

v là biến địa phương

điều gì xảy ra khi DataType là
•kiểu nguyên thủy?
•kiểu tham chiếu?

Truyền tham số và nhận giá trị trả về

```
class Date {
    int year, month, day;
    public Date(int y, int m, int d) {
        year = y; month = m; day = d;
    }
    public void copy(Date d) {
        d.year = year;
        d.month = month;
        d.day = day;
    }
    public Date copy() {
        return new Date(day, month, year);
    }
...
}
```

y, m, d có kiểu nguyên thủy và sẽ nhận giá trị truyền vào

d là tham chiếu, sẽ nhận giá trị truyền vào là nơi lưu trữ đối tượng (object location)

trả về tham chiếu đến đối tượng vừa được tạo. Lưu ý, giá trị này là nơi lưu đối tượng, không phải nội dung đối tượng

Truyền tham số và nhận giá trị trả về

```
...
int thisYear = 2010;
Date d1 = new Date(thisYear, 9, 26);
```

```
class Date {
    int year, month, day;
    public Date(int y, int m, int d) {
        year = y; month = m; day = d;
    }
    public void copy(Date d) {
        d.year = year;
        d.month = month;
        d.day = day;
    }
    public Date copy() {
        return new Date(day, month, year);
    }
...
}
```

```
y = thisYear;
m = 9;
d = 26;
year = y;
month = m;
day = d;
```

Truyền tham số và nhận giá trị trả về

```
...
Date d1 = new Date(thisYear, 9, 26);
Date d2 = new Date(2000, 1, 1);
d1.copy(d2);
```

```
class Date {
    int year, month, day;
    public Date(int y, int m, int d) {
        year = y; month = m; day = d;
    }
    public void copy(Date d) {
        d.year = year;
        d.month = month;
        d.day = day;
    }
    public Date copy() {
        return new Date(day, month, year);
    }
...
}
```

```
d = d2;
d.year = year;
d.month = month;
d.day = day;
```

Truyền tham số và nhận giá trị trả về

```
...
Date d2 = new Date(2000, 1, 1);
Date d3 = d2.copy();
```

```
class Date {
    int year, month, day;
    public Date(int y, int m, int d) {
        year = y; month = m; day = d;
    }
    public void copy(Date d) {
        d.year = year;
        d.month = month;
        d.day = day;
    }
    public Date copy() {
        return new Date(day, month, year);
    }
...
}
```

```
Date temp =
  new Date(d2.day, d2.month, d2.year);
d3 = temp;
```

Tham chiếu “this”

- Tham chiếu this trỏ đến chính đối tượng đó
- Công dụng của this
  - tham chiếu tường minh đến thuộc tính và phương thức của đối tượng
  - truyền tham số và trả lại giá trị
  - dùng để gọi phương thức khởi tạo

"this" để tham chiếu tường minh

```
class Date {
    int year, month, day;
    public Date(int year, int month, int day) {
        this.year = year;
        this.month = month;
        this.day = day;
    }
    public void copy(Date d) {
        d.year = year;
        d.month = month;
        d.day = day;
    }
...
}
```

“this” làm tham số

```
class Document {
    Viewer vi; //reference to the document’s viewer
    ...
    Document(Viewer v) {
        vi = v;
        ...
    }
    void display() {
       //ask the object’s viewer
       //…to display the current document
        vi.display(this);
    }
...
}
```

“this” là giá trị trả về

```
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());
```

"this" để gọi phương thức tạo

- Phương thức tạo chỉ được gọi từ trong phương thức tạo khác và chỉ được gọi một lần (ở lệnh đầu tiên)

```
class Date {
    private int year, month, day;

    public Date(int y, int m, int d) { ... }

    // copy constructor
    Date(Date d) {
        this(d.year, d.month, d.day);
        System.out.println("copy constructor called");
    }
...
}
```

Tổng kết

- Dữ liệu kiểu nguyên thủy
- Tham chiếu
- Giải phóng bộ nhớ
- Truyền tham số
- Tham chiếu **this**