LẬP TRÌNH DI ĐỘNG

Bài 2: Cơ Bản Về Ngôn Ngữ Java

Nội dung

1. Cấu trúc chương trình java đơn giản
2. Vào ra dữ liệu trong ứng dụng console
3. Kiểu dữ liệu
4. Biến
5. Phép toán
6. Câu lệnh lựa chọn
7. Lặp
8. Thử và ngoại lệ

Phần 1

Cấu trúc chương trình java đơn giản

Nhập a, b từ bàn phím và tính a^b

```
import java.io.*;

public class vidu {
    public static void main(String[] args) {
        Console console = System.console();
        if (console == null) System.exit(0);

        System.out.print("Nhap so A: ");
        int a = Integer.parseInt(console.readLine());
        System.out.print("Nhap so B: ");
        int b = Integer.parseInt(console.readLine());
        System.out.println("a^b = " + Math.pow(a, b));
    }
}
```

Phần 2

Vào ra dữ liệu trong ứng dụng console

In dữ liệu ra màn hình

- Sử dụng câu lệnh **`System.out.print(…)`** hoặc **`System.out.println(…)`**
  - Tất cả các tham số được tự động chuyển về kiểu String trước khi in ra
  - Muốn in nhiều giá trị cùng một lúc: sử dụng phép + để ghép các String với nhau
- Sử dụng **`System.out.format(…)`** để in dữ liệu được định dạng
  - Hỗ trợ nhiều định dạng phong phú
  - Cấu trúc định dạng khá phức tạp
  - Chậm

Nhập dữ liệu từ bàn phím

- Java 1.6: sử dụng class Console (như ví dụ)
- Java 1.5: sử dụng class Scanner

```
Scanner scanner = new Scanner(System.in);
System.out.print("Enter your name: ");
String name = scanner.nextLine();
```

- Java 1.4 trở về trước: dùng class BufferedReader

```
BufferedReader reader = new BufferedReader(
    new InputStreamReader(System.in)
);
System.out.print("Enter your name: ");
String name = reader.readLine();
```

Phần 3

Kiểu dữ liệu

Kiểu dữ liệu

- Kiểu dữ liệu trong java chia làm 2 loại:
  - Kiểu Primitive (cơ sở)
  - Kiểu Reference (tham chiếu)
- Ngoài ra còn một loại đặc biệt: immutable (là kiểu tham chiếu nhưng hoạt động như kiểu cơ sở), ví dụ như kiểu chuỗi (String)
- Kiểu primitive gồm:
  - Số nguyên: byte, short, int (*), long
  - Số thực: float, double (*)
  - Kí tự: char (16-bit unicode)
  - Logic: boolean (false/true)

Kiểu dữ liệu

- Kiểu reference:
  - Mảng (array)
  - Lớp (class)
  - Giao diện (interface)
- Chuyển kiểu:
  - Kiểu nhỏ hơn có thể chuyển về kiểu lớn hơn
  - Trường hợp ngược lại cần ép kiểu tường minh

```
int abc = 100;
long xyz = abc;
abc = (int) xyz
```

Kiểu dữ liệu

- Chỉ định kiểu dữ liệu khi viết hằng số:
  - Kiểu long: `1234L`
  - Kiểu float: `12.34f`
  - Kiểu double: `12.34d`
- Một số dạng đặc biệt khi viết hằng số:
  - Hệ cơ số 16: `0x1a`
  - Hệ cơ số 2: `0b11010`
  - Kí tự hex: `'\ufffd'`
  - Dấu ngăn: `123_456_789`

Phần 4

Biến

Biến

- Biến: ô nhớ được đặt tên để dễ dàng thao tác trong các phép toán, biểu thức, câu lệnh,…
- Quy tắc đặt tên trong java:
  - Phân biệt chữ hoa và chữ thường (case-sensitive)
  - Bao gồm các chữ cái, chữ số, dấu $ và dấu _
  - Chữ số không được đứng đầu tên
  - Không được trùng với từ khóa hoặc từ dành riêng
- Khuyến cáo:
  - Đặt tên có nghĩa, gợi nhớ
  - Không quá dài

Biến

- Ví dụ về tên đúng:

```
$$dolar$$      xâu_kí_tự
MyLaptop       diepvien007
```

- Một số quy tắc đặt tên:
  - Từ khóa của java, tên package: luôn viết chữ thường
  - Tên biến và hàm: Viết hoa các chữ đầu của từ, trừ từ đầu tiên
  - Tên class và interface: Viết hoa các chữ đầu của từ
  - Tên hằng số: Viết hoa hoàn toàn

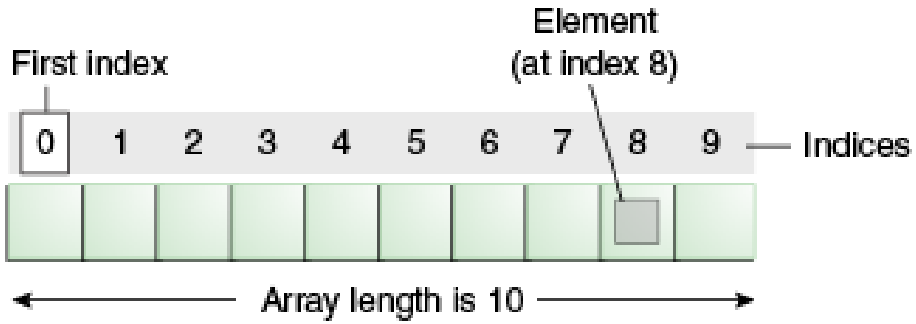
Biến

- Biến có thể khai báo hoặc khai báo và khởi trị luôn cho nó
- Biến địa phương (biến bên trong hàm) cần được khởi trị trước khi sử dụng, nếu không sẽ gây lỗi biên dịch
- Các loại biến khác thông thường sẽ được trình dịch tự động gán giá trị mặc định (bằng dãy các bit 0)
- Ví dụ:

```
float PI = 3.14f;   // khai báo + khởi trị
double pi;          // khai báo
pi = PI;            // gán giá trị
```

Mảng

- Mảng: dãy các biến có cùng kiểu dữ liệu, cùng tên và phân biệt bởi chỉ số
- Nhiều cách khai báo:




```
1. int a[], b;
2. int[] x, y;
3. int[] m = new int [100];
4. int[] n = { 0, 1, 2, 3, 4 };
```

- Thành phần "length": số phần tử của mảng

Phần 5

Phép toán

Phép toán

- Các phép toán toán học (arithmetic operators):
  - 5 phép cơ bản: cộng (+), trừ (-), nhân (*), chia (/), chia lấy dư (%)
  - Thứ tự ưu tiên như trong toán học
  - Phép % chỉ thực hiện với kiểu nguyên
  - Phép / trả về thương nếu cả hai phía đều nguyên
- Phép tăng (++) và giảm (--) chỉ làm việc với kiểu nguyên (chú ý kết quả trả về)

Phép toán

- Phép so sánh:
  - 6 phép cơ bản: bằng (==), khác (!=), nhỏ hơn (<), lớn hơn (>), nhỏ hơn hoặc bằng (<=), lớn hơn hoặc bằng (>=)
  - Kiểu logic có thể làm việc với phép bằng và khác
  - Các kiểu số có thể sử dụng lẫn lộn
- Phép logic:
  - Phép Đảo (!)
  - Phép Hoặc-Nghịch-Đảo (^)
  - Phép hai ngôi đoản-mạch: Và (&&), Hoặc (||)
  - Phép hai ngôi toàn-mạch: Và (&), Hoặc (|)

Phép toán

- Phép kiểm tra kiểu: x instanceof y - trả về đúng (true) nếu biến x có kiểu y
- Phép toán bit: coi các số như là dãy bit và thao tác theo từng bit
  - Phép môt ngôi: phép Đảo (~) - đảo giá trị của các bit
  - Phép hai ngôi: phép Và (&), phép Hoăc (|), phép hoặc nghịch đảo (^)

Phép toán

- Dịch bit: dịch trái (<<), dịch phải (>>), dịch phải dương (>>>)
- Chú ý: các phép dịch bit của java thực ra tương đương với thao tác nhân/chia cho 2^n
- Phép điều kiện:

  (điều kiện) ? giá trị A : giá trị B
- Rút gọn phép toán:

  a = a + b  a += b;

Phần 6

Câu lệnh lựa chọn

Câu lệnh lựa chọn

- Lựa chọn đơn:

```
if (điều kiện) lệnh/khối lệnh
if (0 == (x % 2)) str = “lẻ”;
```

- Lựa chọn đủ:

```
if (điều kiện) lệnh/khối lệnh
else lệnh/khối lệnh
if (0 == (x % 2)) str = “chẵn”;
else str = “lẻ”;
```

Câu lệnh lựa chọn

- Đa lựa chọn: switch

```
switch (month) {
   case 1:  monthString = "January"; break;
   case 2:  monthString = "February"; break;
   case 3:  monthString = "March"; break;
   default: monthString = "Invalid month";
}
```

- Chú ý: từ Java 7, switch có thể sử dụng với kiểu dữ liệu String

Phần 7

Lặp

Lặp

- Lặp while: lặp chừng nào còn đúng

```
while (điều kiện) lệnh/câu lệnh;
while (true) {}
```

- Lặp do-while: lặp cho đến khi sai

```
do
    lệnh/câu lệnh;
while (điều kiện);
```

Lặp

- Lặp for

```
for (initialization; termination; increment) {
    statement(s)
}
```

- Lặp for: duyệt danh sách

```
for (int i: number) {
    statement(s)
}
```

Điều khiển lặp

- “break”: chấm dứt vòng lặp hiện tại
- “continue”: trở về đầu vòng lặp, thực hiện một chu kỳ mới
- “return”: chấm dứt hàm hiện tại (đương nhiên sẽ chấm dứt vòng lặp)
- Chú ý: java không hỗ trợ câu lệnh goto nhưng break và continue có thể có tham số vị trị nhảy đến (tương tự goto – không nên sử dụng)

Phần 8

Thử và ngoại lệ

Thử và Ngoại lệ

- Java cung cấp cơ chế cho phép lập trình viên kiểm soát các đoạn mã có thể sinh lỗi, cơ chế này khá thông dụng trong nhiều ngôn ngữ lập trình hiện đại

```
try {
  // đoạn mã có thể gây lỗi
}
catch (KiểuLỗi x) {
  // khối xử lý lỗi (nếu có)
}
finally {
  // khối kết thúc công việc (dù lỗi hay không)
}
```

Thử và Ngoại lệ

- Chú ý:
  - Cần theo đúng thứ tự try-catch-finally
  - Có thể không có hoặc có nhiều khối catch
  - Có thể không có khối finally
  - Khối finally luôn được thực thi
- Lập trình viên có thể sinh lỗi bằng câu lệnh

```
throw X;    // X là một biến chứa lỗi, bị bắt bởi catch
```

- Lạm dụng khối TCF có thể làm chậm ứng dụng một cách không cần thiết

Thử và Ngoại lệ

- Một số tình huống thường xử dụng TCF
  - Kết nối CSDL
  - Kết nối mạng
  - Làm việc với file, vào/ra/chuyển đổi dữ liệu
- Phân biệt TCF và Mã lỗi
  - Không có phân biệt rõ ràng, tùy vào lập trình viên
  - TCF: lỗi không mong muốn, có thể khắc phục và tiếp tục công việc
  - Lỗi: công việc không thể tiếp tục thực hiện bình thường, cần hủy bỏ