Bài 2: Java căn bản

Nội dung

- Biến & Hằng
- Kiểu dữ liệu (kiểu cơ sở, kiểu tham chiếu)
- Toán tử, biểu thức
- Các cấu trúc điều khiển
- Lớp bao kiểu dữ liệu
- Nhập xuất trong Java

Biến

- **Định nghĩa:**
  - Biến là một vùng nhớ để lưu trữ các giá trị của chương trình
  - Mỗi biến gắn với 1 kiểu dữ liệu và 1 định danh duy nhất là tên biến
  - Tên biến thông thường là một chuỗi các ký tự (Unicode), ký số
    - Trong Java tên biến phân biệt chữ hoa và chữ thường.
    - Tên biến bắt đầu bằng 1 dấu _, $, hay 1 ký tự, không được bắt đầu bằng 1 ký số.
    - Tên biến không có khoảng trắng ở giữa tên.
  - Trong java, biến có thể được khai báo ở bất kỳ nơi đâu trong chương trình.

Biến (tt)

- **Khai báo**

  <kiểu dữ liệu> <tên biến>;

  <kiểu dữ liệu> <tên biến> = <giá trị>;

- **Gán giá trị**

  <tên biến> = <giá trị>;

- **Ví dụ:**

  - **int** age;

    age = 0; **hoặc**

  - **int** age = 0;

Phân loại biến

- **Biến trong Java có 3 loại:**
  - Local variable
  - Instance variable
  - Static variable

Phân loại biến (tt)

- **Local variable**
  - Được khai báo **bên trong** các *method (phương thức), constructor(phương thức khởi tạo), hoặc block (khối lệnh)*
  - Được tạo ra khi method, constructor hoặc block được gọi và biến này sẽ bị destroy (hủy) ngay khi thực hiện xong method, constructor hoặc block.
  - Chúng chỉ xuất hiện nội trong giới hạn của method, constructor hoặc block
  - Không có giá trị mặc định cho các biến local nên chúng nên được khai báo với một giá trị khởi tạo (initial value)

Phân loại biến (tt)

```
public class Test{
    public void pupAge()
    {
        int age = 0;
        age = age + 7;
        System.out.println("Puppy age is : " + age);
    }
}
```

```
public static void pupAge()
    {
        int age;
        age = age + 7;
        System.out.println("Puppy age is : " + age);
    }
```

```
Test.java:4:variable number might not
have been initialized
age = age + 7;
      ^
1 error
```

Phân loại biến (tt)

- **Instance variable**
  - Được khai báo trong *class* (lớp), nhưng **bên ngoài** các *method, constructor hoặc block*.
  - Được tạo ra khi một object (đối tượng) được tạo ra và bị hủy khi object bị hủy
  - Giữ các giá trị được tham chiếu tới bởi nhiều method, constructor hoặc block, hoặc là các phần khác.
  - Có thể được khai báo trong lớp trước hoặc sau khi sử dụng nó.

Phân loại biến (tt)

- Có giá trị mặc định của chúng: *number* thì là 0, *boolean* thì là false, object là null.
- Có thể được truy xuất trực tiếp bằng các gọi tên biến bên trong class. Còn với những *static method* hoặc *lớp khác* (khi instance variable được cho phép truy xuất) , thì chúng có thể được gọi thông qua

ObjectReference.VariableName (tên_đối_tượng.tên_biến)

Phân loại biến (tt)

```
class Employee{
        public String name;
        private double salary;
        public Employee (String empName){
                name = empName;
         }
         public void setSalary(double empSal){
                  salary = empSal;
         }
        public void printEmp(){
                System.out.println("name  : " + name );
                System.out.println("salary :" + salary);   }
                 public static void main(String args[]){
                Employee empOne = new Employee("Ransika");
        empOne.setSalary(1000);
                empOne.printEmp();
        }
```

Instance Variable

Phân loại biến (tt)

- **Static variable**
  - Được khai báo với từ khóa **static** và **bên trong** *class* nhưng **bên ngoài** *method, constructor hoặc block*
  - Biến static là duy nhất bên trong một class (tức là không có biến nào khác cùng tên với nó).
  - Được lưu trữ trong bộ nhớ static (vì vậy biến này sẽ còn tồn tại ứng dụng còn đang chạy).
  - Được tạo ra khi chương trình bắt đầu chạy và hủy khi chương trình dừng.
  - Hầu hết các biến static được khai báo ***public*** .
  - Giá trị mặc định cũng tương tự như instance variable
  - Có thể được truy xuất bằng cách gọi tên lớp: ***ClassName.VariableName (tên_lớp.tên_biến)***

Phân loại biến (tt)

```
public class Employee{
   // salary  variable is a private static variable
   private static double salary;
   public static void main(String args[]){
      salary = 1000;
      System.out.println(salary);
   }
}
public class Test{
   public static void main(String args[]){
            Employee.salary = 2000;
            System.out.println("salary:"+Employee.salary);
            }
}
```

Gọi trực tiếp biến static salary bên trong lớp Employee

Gọi biến salary từ ngoài

Hằng

- Là một giá trị bất biến trong chương trình
- Tên đặt theo qui ước như tên biến
- Được khai báo dùng từ khóa **final**,
- *Hằng số nguyên*: trường hợp giá trị hằng ở dạng long ta thêm vào cuối chuỗi số chữ "l" hay "L".
  - VD: final long y = 20L; // khai báo hằng số long y = 20
- *Hằng số thực*: trường hợp giá trị hằng có kiểu float ta thêm tiếp vĩ ngữ "f" hay "F", còn kiểu số double thì ta thêm tiếp vĩ ngữ "d" hay "D".
  - VD: final float x = 20F; // khai báo hằng số float x= 20
- *Hằng Boolean:* java có 2 hằng boolean là **true, false.**

Hằng (tt)

- *Hằng ký tự*: đặt giữa cặp nháy đơn ‘’
  - Ví dụ: ‘a’: hằng ký tự a
- Hằng chuỗi: là tập hợp các ký tự được đặt giữa hai dấu nháy kép “”. Một hằng chuỗi không có ký tự nào là một hằng chuỗi rỗng.
  - Ví dụ: “Hello Wolrd”

Hằng ký tự đặc biệt

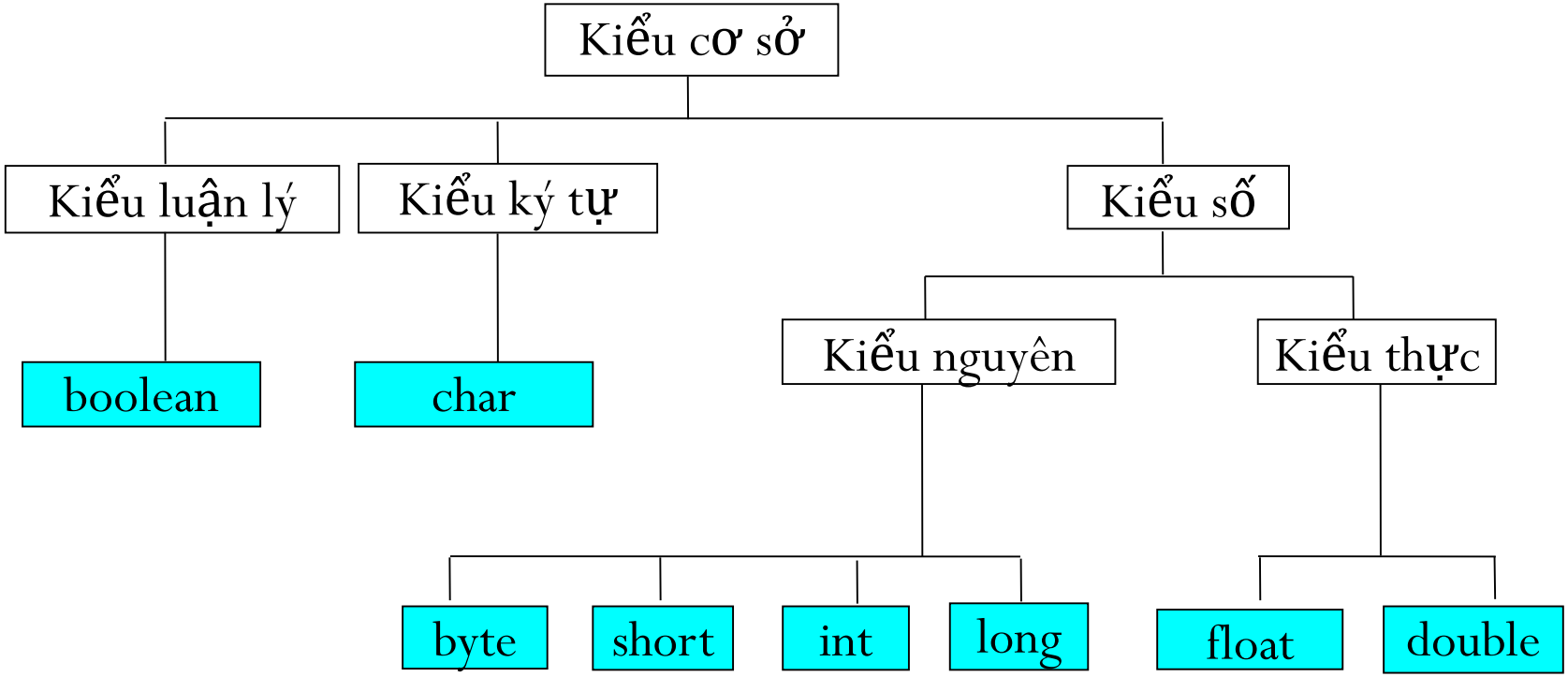
Ký tự	Ý nghĩa
\b	Xóa lùi (BackSpace)
\t	Tab
\n	Xuống hàng
\r	Dấu enter
\”	Nháy kép
\’	Nháy đơn
\\	\
\f	Đẩy trang
\uxxxx	Ký tự unicode

Kiểu dữ liệu

- Kiểu dữ liệu cơ sở (primitive data type)
- Kiểu dữ liệu tham chiếu (reference data type)

Kiểu dữ liệu cơ sở

- **Kiểu dữ liệu cơ sở (primitive data type)**
  - Có 8 kiểu dữ liệu cơ sở: *byte*, *short*, *int*, *long*, *float*, *double*, *boolean* và *char*



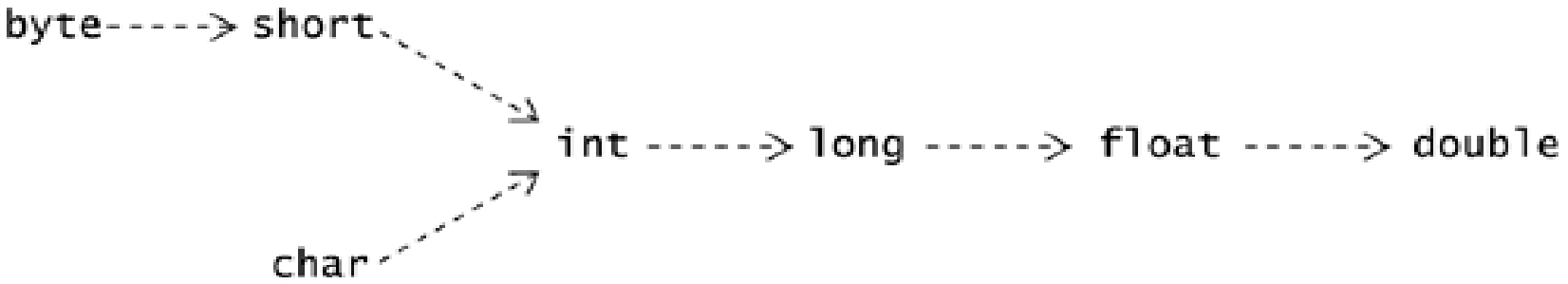
Kiểu dữ liệu cơ sở (tt)

Kiểu	Kích thước (bits)	Giá trị	Giá trị mặc định
boolean	[*Note:* The representation of a boolean is specific to the Java Virtual Machine on each computer platform.]	true và false	**false**
char	16	'\u0000' to '\uFFFF' (0 to 65535)	**null**
byte	8	–128 to +127 (-2^7 to $2^7 - 1$)	**0**
short	16	–32,768 to +32,767 (-2^{15} to $2^{15} - 1$)	**0**
int	32	–2,147,483,648 to +2,147,483,647 (-2^{31} to $2^{31} - 1$)	**0**
long	64	–9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 (–263 to 263 – 1)	**0l**
float	32	1.4012984632481707e–45 to 3.4028234663852886E+38	**0.0f**
double	64	4.94065645841246544e–324 to 1.7976931348623157E+308	**0.0d**

Kiểu dữ liệu cơ sở (tt)

- **Chuyển đổi kiểu dữ liệu:** khi có sự không tương thích về kiểu dữ liệu (gán, tính toán biểu thức, truyền đối số gọi phương thức)
- Chuyển kiểu hẹp (lớn → nhỏ): *cần ép kiểu*

  <tên biến 2> = (kiểu dữ liệu) <tên biến 1>;
- Chuyển kiểu rộng (nhỏ → lớn): *tự động chuyển*



Kiểu dữ liệu cơ sở (tt)

➢ Kiểu số nguyên:

- Java cung cấp 4 kiểu số nguyên khác nhau là: *byte*, *short*, *int*, *long*.
- Kiểu mặc định của các số nguyên là kiểu *int*.
- Các số nguyên kiểu *byte* và *short* rất ít khi được dùng.
- Trong java không có kiểu số nguyên không dấu như trong ngôn ngữ C/C++.
- Khai báo và khởi tạo giá trị cho các biến kiểu nguyên:
  - int x = 0;
  - long y = 100;

Kiểu dữ liệu cơ sở (tt)

➢ **Một số lưu ý đối với các phép toán trên số nguyên:**

- Nếu hai toán hạng kiểu *long* thì kết quả là kiểu *long*.
- Một trong hai toán hạng không phải kiểu *long* sẽ được chuyển thành kiểu *long* trước khi thực hiện phép toán.
- Nếu hai toán hạng đầu không phải kiểu *long* thì phép tính sẽ thực hiện với kiểu int.
- Các toán hạng kiểu *byte* hay *short* sẽ được chuyển sang kiểu *int* trước khi thực hiện phép toán.

```
byte x = 5;
byte y = 10;
byte z = x + y; //Dòng lệnh báo lỗi kiểu dữ liệu
Sửa lại byte z = (byte) (x + y);
```

Kiểu dữ liệu cơ sở (tt)

- Trong java không thể chuyển biến kiểu *int* và kiểu *boolean* như trong ngôn ngữ C/C++.

```
boolean b = false;
if (b == 0)
{
    System.out.println("Xin chao");
}
```

Báo lỗi: không được phép so sánh biến kiểu boolean với một giá trị kiểu int.

Kiểu dữ liệu cơ sở (tt)

➢ Kiểu dấu chấm động:

- Java cung cấp hai kiểu dữ liệu là *float* và *double*.
- Kiểu *float* có kích thước 4 byte và giá trị mặc định là 0.0f. Kiểu *double* có kích thước 8 byte và giá trị mặc định là 0.0d
- Số kiểu dấu chấm động không có giá trị nhỏ nhất cũng không có giá trị lớn nhất. Chúng có thể nhận các giá trị:
  - Số âm
  - Số dương
  - Vô cực âm
  - Vô cực dương
- Khai báo và khởi tạo giá trị cho các biến kiểu dấu chấm động:
  - *float x = 100.0/7; hoặc float x = 100.0f*
  - *double y = 1.56E6;*

Kiểu dữ liệu cơ sở (tt)

- **Một số lưu ý đối với các phép toán trên số dấu chấm động:**
  - Nếu mỗi toán hạng đều có kiểu dấu chấm động thì phép toán chuyển thành phép toán dấu chấm động.
  - Nếu có một toán hạng là *double* thì các toán hạng còn lại sẽ được chuyển thành kiểu *double* trước khi thực hiện phép toán.
  - Biến kiểu *float* và *double* có thể ép chuyển sang kiểu dữ liệu khác trừ kiểu boolean.

Kiểu dữ liệu cơ sở (tt)

➢ **Kiểu ký tự (char):**

- Kiểu ký tự *(char)* trong ngôn ngữ lập trình Java có kích thước là 2 bytes và chỉ dùng để biểu diễn các ký tự trong bộ mã Unicode.
- Kiểu *char* trong java có thể biểu diễn tất cả $2^{16} = 65536$ ký tự khác nhau.
- Giá trị mặc định cho một biến kiểu *char* là null.
- Khai báo
  - *char kyTu = 'a'*

Kiểu dữ liệu cơ sở (tt)

➢ **Kiểu luận lý (boolean):**

- Kiểu *boolean* chỉ nhận 1 trong 2 giá trị: **true** hoặc **false**.
- Trong java kiểu *boolean* không thể chuyển thành kiểu nguyên và ngược lại.
- Giá trị mặc định của kiểu *boolean* là **false**.
- Khai báo
  - *boolean b = false*

Kiểu dữ liệu tham chiếu

- **Kiểu dữ liệu tham chiếu (reference data type):**
  - Kiểu mảng
  - Kiểu đối tượng (class)

Kiểu dữ liệu tham chiếu (tt)

- **Kiểu mảng:**
  - Mảng là *tập hợp* các phần tử *có cùng tên* và *cùng kiểu dữ liệu*.
  - Mỗi phần tử được truy xuất thông qua *chỉ số*.
  - **Khai báo:**

<kiểu dữ liệu>[] <tên mảng>; // mảng 1 chiều

<kiểu dữ liệu> <tên mảng>[]; // mảng 1 chiều

<kiểu dữ liệu>[][] <tên mảng>; // mảng 2 chiều

<kiểu dữ liệu> <tên mảng>[][]; // mảng 2 chiều

Kiểu mảng

- **Khởi tạo:**

```
int     arrInt[] = {1, 2, 3};
char    arrChar[] = {'a', 'b', 'c'};
```

- **Cấp phát bộ nhớ cho mảng**

```
int arrInt[100]; // Khai báo này trong Java sẽ bị báo lỗi.
int [] arrInt = new int[100]; //Khai báo dùng từ khóa new
```

- **Truy cập mảng**

- Chỉ số mảng ***n*** phần tử: từ ***0*** đến ***n-1***
- Các phần tử được truy xuất thông qua chỉ số của nó đặt giữa cặp dấu ngoặc vuông ([]).

```
int arrInt[] = {1, 2, 3};
int x = arrInt[0]; // x sẽ có giá trị là 1.
int y = arrInt[1]; // y sẽ có giá trị là 2.
int z = arrInt[2]; // z sẽ có giá trị là 3.
```

Kiểu mảng - ArrayList

- ArrayList hỗ trợ mảng động có thể tăng kích thước khi cần
- ArrayList được tạo với một kích cỡ ban đầu. Khi kích cỡ này bị vượt, collection tự động được tăng. Khi các đối tượng bị gỡ bỏ, ArrayList có thể bị giảm kích cỡ.
- Phương thức khởi tạo
  - ArrayList()
  - ArrayList(Collection c)
  - ArrayList(int initialCapacity)
- Khởi tạo:

```
ArrayList<Object> obj = new ArrayList<Object>() ;
```

Kiểu mảng – ArrayList (tt)

➢Các phương thức cơ bản trong ArrayList

Phương thức	Miêu tả
void add(int index, Object element)	Chèn phần tử tại vị trí index trong list. Ném IndexOutOfBoundsException nếu index này ở bên ngoài dãy (index < 0 || index > size())
boolean add(Object o)	Thêm phần tử ở cuối list
void clear()	Gỡ bỏ tất cả phần tử trong list
Object clone()	Trả về một bản copy của ArrayList này
boolean contains(Object o)	Kiểm tra phần tử có trong ArrayList không. Trả về true nếu list này chứa phần tử đã cho.
Object get(int index)	Lấy phần tử tại vị trí index xác định. Ném IndexOutOfBoundsException nếu index đã cho là ở bên ngoài dãy (index < 0 || index >= size())
int indexOf(Object o)	Trả về vị trí index trong list của phần tử, hoặc -1 nếu List không chứa phần tử này

Kiểu mảng – ArrayList (tt)

Phương thức	Miêu tả
Object remove(int index)	Gỡ bỏ phần tử tại vị trí index. Ném IndexOutOfBoundsException nếu index ở ngoài dãy (index < 0 || index >= size())
protected void removeRange(int fromIndex, int toIndex)	Gỡ bỏ từ list này tất cả phần tử mà có index ở giữa fromIndex và toIndex
Object set(int index, Object element)	Thay thế phần tử tại vị trí đã cho trong list này bằng phần tử khác. Ném IndexOutOfBoundsException nếu index ở ngoài dãy (index < 0 || index >= size())
int size()	Trả về số phần tử trong list
void trimToSize()	Đưa dung lượng của ArrayList này về kích cỡ hiện tại của list đó

Kiểu đối tượng

➢ Kiểu đối tượng

- Dữ liệu kiểu đối tượng do người dùng định nghĩa. Chứa tập các *thuộc tính* và *phương thức*.
- **Khai báo đối tượng**

  <Kiểu đối tượng> <biến ĐT>;

  VD: Animal animal;
- **Khởi tạo đối tượng**

  <Kiểu đối tượng> <biến ĐT> = ***new*** *<Kiểu đối tượng>;*

  VD: Animal animal = new Animal("giraffe");
- **Truy xuất thành phần đối tượng**

  <biến ĐT>.<thuộc tính>

  VD: animal.tall;

  <biến ĐT>.<phương thức>

  VD: animal.addanimal();

Toán tử và biểu thức

- Toán tử số học
- Toán tử trên bit
- Toán tử quan hệ và logic
- Toán tử gán
- Toán tử ép kiểu
- Toán tử điều kiện

Toán tử và biểu thức (tt)

- **Toán tử số học:**

Toán tử	Ý nghĩa
+	Cộng
-	Trừ
*	Nhân
/	Chia nguyên
%	Chia dư
++	Tăng 1
--	Giảm 1

Toán tử và biểu thức (tt)

- **Toán tử trên bit:**

Toán tử	Ý nghĩa
&	AND
|	OR
^	XOR
<<	Dịch trái
>>	Dịch phải
~	Bù bit

Toán tử và biểu thức (tt)

- **Toán tử quan hệ và logic:**

Toán tử	Ý nghĩa
= =	So sánh bằng
!=	So sánh khác
>	So sánh lớn hơn
<	So sánh nhỏ hơn
>=	So sánh lớn hơn hay bằng
<=	So sánh nhỏ hơn hay bằng
||	OR (biểu thức logic)
&&	AND (biểu thức logic)
!	NOT (biểu thức logic)

Toán tử và biểu thức (tt)

- **Toán tử gán:**

Toán tử	Ví dụ	Ý nghĩa
=	a = b	gán a = b
+=	a += 5	a = a + 5
-=	b -= 10	b = b – 10
*=	c *= 3	c = c * 3
/=	d /= 2	d = d/2
%=	e %= 4	e = e % 4

Toán tử và biểu thức (tt)

➢ Toán tử ép kiểu:

- Ép kiểu rộng (*widening conversion*): từ kiểu nhỏ sang kiểu lớn (không mất mát thông tin)
- Ép kiểu hẹp (*narrow conversion*): từ kiểu lớn sang kiểu nhỏ (có khả năng mất mát thông tin)

<tên biến> = (kiểu_dữ_liệu) <tên_biến>;

Ví dụ:

```
float fNum = 2.2;
int iCount = (int) fNum;  // (iCount = 2)
```

Toán tử và biểu thức (tt)

➢ Toán tử điều kiện:

- **Cú pháp:** <điều kiện> ? <biểu thức 1> : < biểu thức 2>
- Nếu điều kiện đúng thì thực hiện <biểu thức 1>, còn ngược lại là <biểu thức 2>.
- <điều kiện>: là một *biểu thức logic*
- <biểu thức 1>, <biểu thức 2>: có thể là *hai giá trị, hai biểu thức* hoặc *hai hành động.*

- ***Ví dụ***:

```
int x = 10;
int y = 20;
int z = (x<y) ? 50 : 80;
// Kết quả z = 50 do biểu thức (x < y) là đúng.
```

Cấu trúc điều khiển

- Cấu trúc **if….else**
- Cấu trúc **switch… case**
- Cấu trúc lặp
- Cấu trúc lệnh nhảy **jump**

Cấu trúc điều khiển

- Cấu trúc **if….else**
  - **Dạng 1:**

```
if (<điều_kiện>) {
        <khối_lệnh>;
}
```

  - **Dạng 2:**

```
if (<điều_kiện>)  {
        <khối _lệnh1>;
}
else {
        <khối _lệnh2>;
}
```

Cấu trúc điều khiển

- Cấu trúc **switch….case**

```
switch (<biến>) {
        case <giátrị_1>:
                <khối_lệnh_1>;
                break;
        ....
        case <giátrị_n>:
                <khối_lệnh_n>;
                break;
        default:
                <khối lệnh default>;
}
```

Là giá trị hằng

Dùng để thoát khỏi cấu trúc switch

Cấu trúc điều khiển

```
switch(wash) {
  case 1: / / wash is 1 for Cotton
            System.out.println("Cotton selected");
  break;
  case 2: / / wash is 2 for Linen
            System.out.println("Linen selected");
  break;
  case 3: / / wash is 3 forWool
            System.out.println("Wool selected");
  break;
  default: / / Not a valid value for wash
            System.out.println("Selection error");
  break;
}
```

Cấu trúc điều khiển

- Cấu trúc **lặp**
  - **Dạng 1:**

```
while (<điều_kiện_lặp>) {
        <khối _lệnh>;
}
```

  - **Dạng 2:**

```
do {
        <khối_lệnh>;
} while (điều_kiện);
```

  - **Dạng 3:**

```
for (khởi_tạo_biến_đếm;đk_lặp;tăng_biến) {
        <khối _lệnh>;
}
```

Cấu trúc điều khiển

- Cấu trúc **lệnh nhảy jump:**
  - Lệnh **break**: trong cấu trúc lặp, câu lệnh break dùng để thóat khỏi cấu trúc lặp trong cùng chứa nó.
  - Lệnh **continue**: dùng để tiếp tục vòng lặp trong cùng chứa nó (ngược với break).
  - Dùng kết hợp nhãn (**label**) với từ khóa ***break*** và ***continue*** để thay thế cho lệnh ***goto*** (trong C).

Cấu trúc điều khiển

- Cấu trúc **lệnh nhảy jump:**
  - Ví dụ:

```
label:
for (…) {
    for (…) {
        if (<biểu thức điều kiện>)
            break label;
        else
            continue label;
    }
}
```

Lớp bao kiểu dữ liệu

Data type	Wrapper Class (java.lang.*)	Ghi chú
boolean	Boolean	- Gói (package): chứa nhóm nhiều class. - Ngoài các Wrapper Class, gói java.lang còn cung cấp các lớp nền tảng cho việc thiết kế ngôn ngữ java như: String, Math, …
byte	Byte	
short	Short	
char	Character	
int	Integer	
long	Long	
float	Float	
double	Double	

Ví dụ lớp bao kiểu dữ liệu

```
Float a=new Float(5.5);
  System.out.println(a.floatValue());
  Float fObj1 = new Float("5.35");
  Float fObj2 = new Float("5.34");
  int i2 = fObj1.compareTo(fObj2);
  if(i2 > 0){
    System.out.println("First is grater");
   }else if(i2 < 0){
    System.out.println("Second is grater");
   }else{
    System.out.println("Both are equal");
   }
   }
```

Ví dụ lớp bao kiểu dữ liệu

- Ví dụ lớp **String**:
  - Biểu diễn chuỗi

    String s1 = “abc”;

    String s2 = new String(“def”);

    String s3 = s1 + s2;
  - So sánh chuỗi

    s1.equals(“abc”)

    s2.equalIgoredCase(“def”);

Nhập xuất trong Java

- Cách đầu tiên là sử dụng lớp **BufferReader**
- Cách 2 là sử dụng lớp **Scanner** (hay dùng)
- Cách 3 là sử dụng **JOptionPane**

Lớp BufferedReader

- **BufferReader** là một lớp dùng để đọc dữ liệu từ bàn phím hay từ file.Có thể dùng lớp này để đọc một *chuỗi*, một *mảng* hoặc một *ký tự*.
- Tham số đầu vào của BufferReader có thể là **InputStreamReader** hoặc **FileReader** (Dùng để đọc file).
- Một số phương thức của lớp **BufferReader**:
  - **read()** : đọc một kí tự.
  - **readLine()** : đọc một dòng text.
- Khai báo

```
BufferedReader br= new BufferedReader(new InputStreamReader(System.in));
String number=br.readLine();
```

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class NhapXuat {
    public static int a, b,tong;
    public static void main(String[] args) {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String str;
        System.out.println("Nhap a:");
        str=br.readLine();
        a=Integer.parseInt(str);
       System.out.println("Nhap b:");
        str=br.readLine();
        b=Integer.parseInt(str);
        tong=a+b;
        System.out.println("tong "+tong);
    }
}
```

Lớp Scanner

- Phân loại được dữ liệu mà người dùng nhập vào: có thể đọc kiểu int hay float, double,…
- Một số phương thức của lớp **Scanner:**
  - **next() //dùng cho String**
  - **nextInt()**
  - **nextFloat()**
  - **nextBoolean()**
  - **nextByte()**
  - **nextLine()**

Lớp Scanner

Lớp java.util.Scanner

public boolean	**nextBoolean()**
public byte	**nextByte()**
public byte	**nextByte(int radix)**
public double	**nextDouble()**
public float	**nextFloat()**
public int	**nextInt()**
public int	**nextInt(int radix)**
public String	**nextLine()**
public long	**nextLong()**
public long	**nextLong(int radix)**
public short	**nextShort()**
public short	**nextShort(int radix)**

Lớp Scanner

Ví dụ:

```
Scanner in = new Scanner(System.in);
int number = in.nextInt();
String string = in.nextLine();
float real = in.nextFloat();
String string2 = in.nextLine();
```

Lớp JOptionPane

- JOptionPane là một lớp thừa kế từ lớp **JComponent**.
- Khi biên dịch program thì nó sẽ hiện lên một **dialog box** cho phép cho ta nhập dữ liệu.
- Một số phương thức của lớp **JOptionPane:**
  - **showConfirmDialog()** : Hiển thị một câu hỏi lựa chọn giống như *yes, no, cancel*
  - **showInputDialog()** : Hiển thị box nhập
  - **showMessageDialog()** : Báo cho người dùng một sự kiện vừa xảy ra.
- Dữ liệu nhập vào chỉ có kiểu *String*

Lớp JOptionPane

```
import javax.swing.JOptionPane;
public class InputFromKeyboardJOptionPane
{
    public static void main(String[] args)
    {
        String name = "";
        name=JOptionPane.showInputDialog("Please enter your name");
        String msg = "Hello " + name + "!";
        JOptionPane.showMessageDialog(null, msg);
        System.out.println("Name is:"+msg);
    }
}
```

Q & A