

Chương 2: Cơ bản về ngôn ngữ Java



ThS. Trần Văn Hữu



Nội dung

- ❖ Biến & Hằng
- ❖ Kiểu dữ liệu
- ❖ Toán tử, biểu thức
- ❖ Các cấu trúc điều khiển (chọn, rẽ nhánh, lặp)
- ❖ Lớp bao kiểu cơ sở
- ❖ Phương thức và cách sử dụng
- ❖ Một số lớp cơ bản
- ❖ Nhập/Xuất từ bàn phím



Biến (1)

- ❖ Biến?
- ❖ Tên biến: *Phân biệt chữ hoa chữ thường*
- ❖ Trong java, biến có thể được khai báo ở bất kỳ nơi đâu trong chương trình.
- ❖ Biến toàn cục?
- ❖ Biến cục bộ?



Biến (2)

❖ Cách khai báo biến:

<kiểu_dữ_liệu> <tên_biến>;

<kiểu_dữ_liệu> <tên_biến> = <giá_trị>;

❖ Gán giá trị cho biến:

<tên_biến> = <giá_trị>;

❖ Nguyên tắc đặt tên biến

- Gồm các ký tự chữ, ký tự số, dấu gạch dưới '_', và dấu '\$'.
- Bắt đầu bằng ký tự chữ.
- Không được trùng với từ khóa và từ dành riêng của java.
- Có phân biệt chữ hoa – thường.
- Chỉ gồm một từ đơn và nên viết chữ thường.
- Nếu tên biến gồm nhiều từ, ký tự đầu của từ đầu viết thường, ký tự đầu của mỗi từ tiếp theo viết hoa.



Biến (3)

❖ Lưu ý:

- Trong java nếu lúc khai báo không khởi tạo giá trị cho biến thì nó sẽ nhận 1 giá trị mặc định. Mỗi kiểu dữ liệu có 1 giá trị mặc định khác nhau.
 - 0 nếu kiểu dữ liệu là kiểu số
 - '\0' nếu kiểu dữ liệu là ký tự
 - false nếu kiểu dữ liệu là boolean
 - null nếu kiểu dữ liệu là lớp



Hằng

- ❖ Hằng?

- ❖ Tên đặt theo qui ước như tên biến

- ❖ Khai báo dùng từ khóa **final**

- ❖ Ví dụ:

```
final int x = 10; // khai báo hằng số nguyên x = 10
```

```
final long y = 20; // khai báo hằng số long y = 20
```

- ❖ **Hằng ký tự**: đặt giữa cặp nháy đơn “

- ❖ **Hằng chuỗi**: là một dãy ký tự đặt giữa cặp nháy đôi “”



Hàng 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)

Primitive Data Types

- ❖ byte
- ❖ char
- ❖ boolean
- ❖ short
- ❖ int
- ❖ long
- ❖ float
- ❖ double

Reference data types

- ❖ Array
- ❖ Class
- ❖ Interface

Kiểu dữ liệu cơ sở

Type	Size in bits	Values	Standard
<code>boolean</code>		<code>true</code> or <code>false</code>	
		[<i>Note:</i> The representation of a boolean is specific to the Java Virtual Machine on each computer platform.]	
<code>char</code>	16	'\u0000' to '\uFFFF' (0 to 65535)	(ISO Unicode character set)
<code>byte</code>	8	-128 to +127 (-2^7 to $2^7 - 1$)	
<code>short</code>	16	-32,768 to +32,767 (-2^{15} to $2^{15} - 1$)	
<code>int</code>	32	-2,147,483,648 to +2,147,483,647 (-2^{31} to $2^{31} - 1$)	
<code>long</code>	64	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 (-2^{63} to $2^{63} - 1$)	
<code>float</code>	32	<i>Negative range:</i> -3.4028234663852886E+38 to -1.40129846432481707e-45 <i>Positive range:</i> 1.40129846432481707e-45 to 3.4028234663852886E+38	(IEEE 754 floating point)
<code>double</code>	64	<i>Negative range:</i> -1.7976931348623157E+308 to -4.94065645841246544e-324 <i>Positive range:</i> 4.94065645841246544e-324 to 1.7976931348623157E+308	(IEEE 754 floating point)

The Java primitive types.

Kiểu dữ liệu cơ sở

❖ Kiểu số nguyên

- 4 kiểu số nguyên khác nhau là: `byte`, `short`, `int`, `long`
- Kiểu **mặc định** của các số nguyên là kiểu `int`
- Không có kiểu số nguyên không dấu

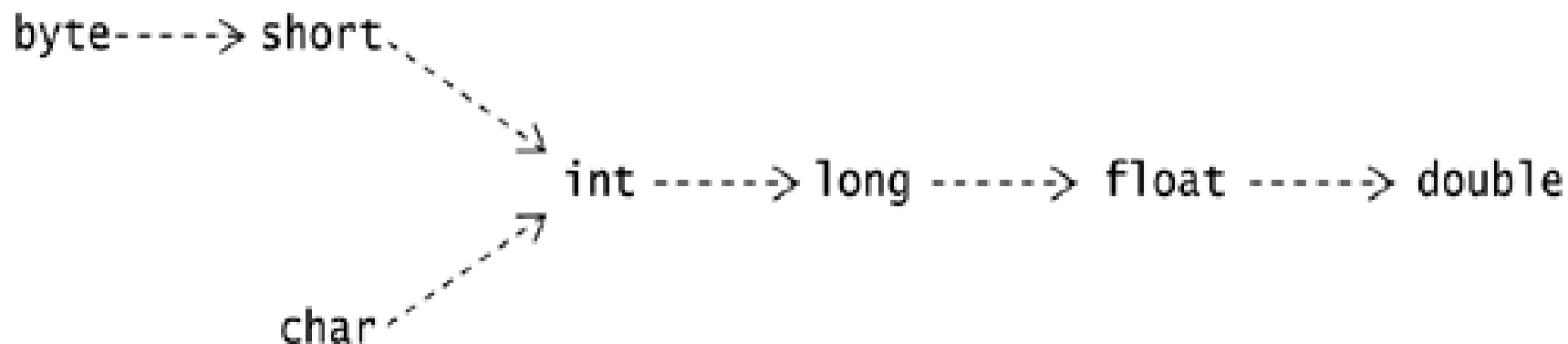
```
boolean b = false;  
if (b == 0){  
    System.out.println("Xin chào");  
}
```





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 tham chiếu (1)

❖ Kiểu mảng:

- Khai báo:

<i><kiểu dữ liệu>[]</i>	<i><tên mảng>; //mảng 1 chiều</i>
<i><kiểu dữ liệu></i>	<i><tên mảng>[]; //mảng 1 chiều</i>
<i><kiểu dữ liệu>[][]</i>	<i><tên mảng>; //mảng 2 chiều</i>
<i><kiểu dữ liệu></i>	<i><tên mảng>[][]; //mảng 2 chiều</i>

Kiểu dữ liệu tham chiếu (2)

❖ Khởi tạo mảng:

```
int    arrInt[ ]      = {1, 2, 3};
```

```
char   arrChar[ ]    = {'a', 'b', 'c'};
```

```
String arrString[ ]  = {"ABC", "EFG", "GHI"};
```

❖ Cấp phát & truy cập mảng:

```
int arrInt[ ] = new int[100];
```

```
int arrInt[100]; // Khai báo này trong Java sẽ báo lỗi.
```

Chỉ số mảng n phần tử: từ 0 đến $n-1$

Chú ý: Mảng trong java được xem là “một đối tượng”

Kiểu dữ liệu tham chiếu (3)

❖ Các hàm với mảng trong java

- Thuộc tính **length**: Cung cấp số phần tử của mảng

- Ví dụ: `int A[]={1, 2, 3, 4, 5};`
`int a=A.length; khi đó a=5.`

`public static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)`

- Hàm **System.arraycopy** trong gói thư viện **System**

- Ví dụ: `int a[]={1, 3, 5, 7, 9, 11, 13, 15}`
`int b[]={2, 4, 6, 8, 10, 12, 14}`
`System.arraycopy(a, 3, b, 2, 4);`

- Kết quả của đoạn mã lệnh trên là mảng b có giá trị mới là {2, 4, 7, 9, 11, 13, 14}

Kiểu dữ liệu tham chiếu (4)

❖ Kiểu mảng nhiều chiều:

```
int b[ ][ ];
```

```
b = new int[ 3 ][ 4 ];
```

```
int b[ ][ ];
```

```
b = new int[ 2 ][ ];
```

```
b[ 0 ] = new int[ 5 ];
```

```
b[ 1 ] = new int[ 3 ];
```

```
int b[ ][ ] = { { 1, 2 }, { 3, 4 } };
```

```
int b[ ][ ] = { { 1, 2 }, { 3, 4, 5 } };
```


Kiểu dữ liệu tham chiếu (5)

❖ Kiểu đối tượng:

- Khai báo đối tượng

<Kiểu đối tượng> <biến ĐT>;

- Khởi tạo đối tượng

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

- Truy xuất thành phần đối tượng

<biến ĐT>.<thuộc tính>

<biến ĐT>.<phương thức>([tham số])



Một số toán tử (1)

❖ Toán tử số học:

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

Một số toán tử (2)

❖ Toán tử quan hệ & 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)

Một số toán tử (3)

❖ Toán tử gán:

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

Một số toán tử (4)

❖ Toán tử điều kiện:

<điều kiện> ? <biểu thức 1> : <biểu thức 2>

int x = 10;

int y = 20;

int Z = (x < y) ? 30 : 40;

// Kết quả z = 30 do biểu thức (x < y) là đúng.



Cấu trúc chọn

❖ Cấu trúc điều kiện if ... else

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

```
if (<điều_kiện>)  
{  
    <khởi_lệnh1>;  
}  
else  
{  
    <khởi_lệnh2>;  
}
```



Ví dụ - Cấu trúc if...else

```
import java.util.Date;
public class TestIf{
    public static void main( String args[ ] ){
        Date today = new Date();
        if ( today.getDay() == 0 )
            System.out.println("Hom nay la chu nhat");
        else
            System.out.println("Hom nay khong la CN" );
    }
}
```

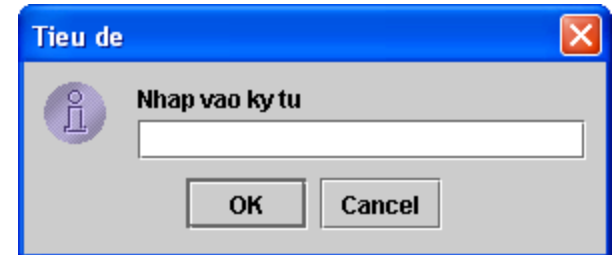
Cấu trúc chọn

❖ Cấu trúc switch ... case

```
switch (<biến>){  
    case <giá trị_1>:  
        <khởi_lệnh_1>;  
        break;  
  
    ....  
    case <giá trị_n>:  
        <khởi_lệnh_n>;  
        break;  
    default:  
        <khởi_lệnh default>;  
}
```


Ví dụ - Cấu trúc switch...case

```
import javax.swing.JOptionPane;
public class TestSwitch
{
    public static void main(String[] args)
    {
        char c;
        String str=JOptionPane.showInputDialog(null,
                                                "Nhập vào ký tự", "Tieu de", 0);
        c = str.charAt(0);
    }
}
```





Ví dụ - Cấu trúc switch...case

```
switch (c){  
    case 'a': case 'e': case 'i': case 'o': case 'u':  
        System.out.println("Ky tu nay la nguyen am");  
        break;  
    default:  
        System.out.println("Ky tu nay la phu am");  
}  
System.exit(0); // kết thúc chương trình  
}
```



Cấu trúc lặp

```
while (điều_kiện_lặp)      do
{
    khối _lệnh;
}                          } while (điều_kiện);
```

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



Ví dụ - Cấu trúc lặp

//Chương trình tính tổng các số lẻ từ 1 đến 100

```
public class TestFor{  
    public static void main(String[] args)  
    {  
        int tong = 0;  
        for ( int i=1; i<=100; i+=2)  
            tong+=i;  
        System.out.println("Tong = " + tong);  
    }  
}
```

Ví dụ - Cấu trúc lặp

// Tính tổng các số lẻ từ 1 đến 100

```
int tong = 0, i = 1;
```

```
while (i<=100)
```

```
{
```

```
    tong+=i;
```

```
    i+=2;
```

```
}
```

```
System.out.println("Tong = " + tong);
```



Cấu trúc lệnh nhảy

- ❖ break
- ❖ continue
- ❖ label: Java dùng kết hợp nhãn (label) với từ khóa break và continue để thay thế cho lệnh goto.

label1:

for (...)

{ for (...)

{ if (<biểu thức điều kiện>)

break label1;

else

continue label1;

}

}



Cấu trúc lệnh nhảy – Ví dụ 1

```
import javax.swing.JOptionPane;
public class BreakTest{
    public static void main( String args[] ){
        String output = "";
        int count;
        for ( count = 1; count <= 10; count++ ){
            if ( count == 5 )
                break;           // terminate loop
            output += count + " ";
        }
        .....
```



Cấu trúc lệnh nhảy – Ví dụ 1

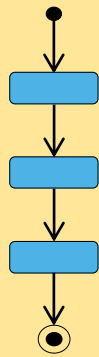
.....

```
output += "\nBroke out of loop at count = " + count;  
JOptionPane.showMessageDialog( null,  
    output, "Thong bao",  
    JOptionPane.INFORMATION_MESSAGE);  
System.exit( 0 ); // terminate application  
}  
}
```


Cấu trúc lệnh nhảy – Ví dụ 2

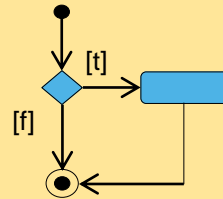
```
import javax.swing.JOptionPane;
public class ContinueTest{
    public static void main( String args[] ){
        String output = "";
        for ( int count = 1; count <= 10; count++ ) {
            if ( count == 5 )           // if count is 5,
                continue;              // skip remaining code in loop
            output += count + " ";
        }
        output += "\nUsed continue to skip printing 5";
        JOptionPane.showMessageDialog( null, output );
        System.exit( 0 );
    }
}
```

Sequence

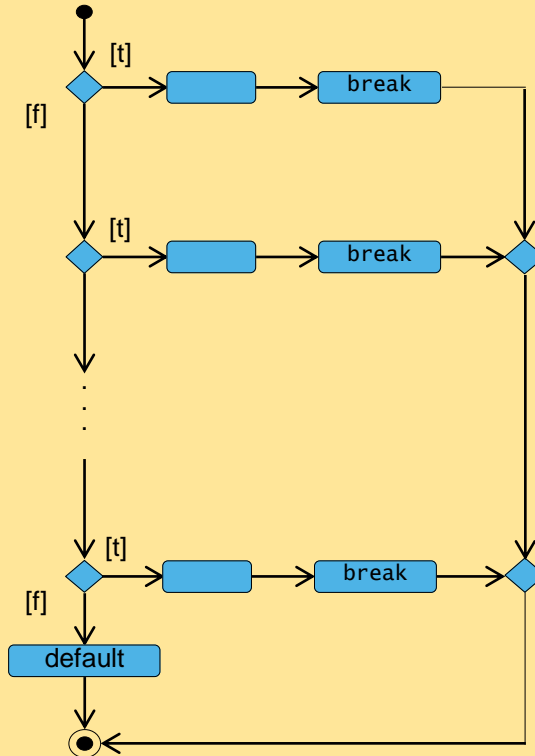


Selection

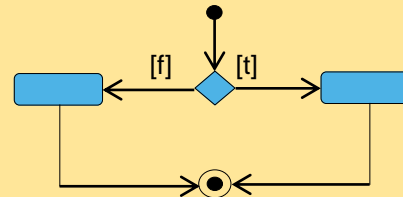
if statement (single selection)



switch statement (multiple selection)

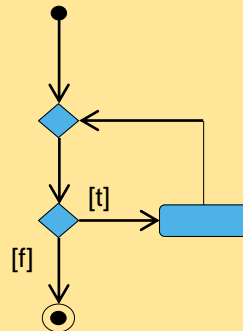


if else statement (double selection)

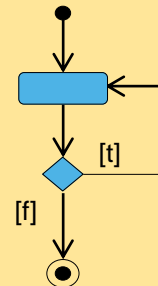


Repetition

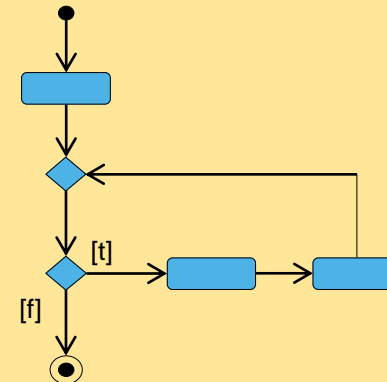
while statement



do while statement



for statement





Lớp bao kiểu dữ liệu cơ sở

❖ Tồn tại các lớp đối tượng tương ứng

Data type	Wrapper Class (java.lang.*)	Ghi chú
boolean	Boolean	<ul style="list-style-type: none">- 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	



Một số lớp cơ sở

- ❖ Các lớp dữ liệu nguyên thủy cung cấp:
 - Các phương thức tiện ích:
 - `valueOf(String s)`: Trả đối tượng thuộc kiểu tương ứng
 - `static parseType(String s)`: Trả giá trị nguyên thủy tương ứng
 - Hằng số:
 - `Type.MAX_VALUE`, `Type.MIN_VALUE`
 - `Type.POSITIVE_INFINITY`, `Type.NEGATIVE_INFINITY`



Ví dụ - Lớp bao kiểu dữ liệu cơ sở

```
class tong {  
    public static void main (String args[])  
    {  
        int i=0;  
        int tong=0;  
        for (i=0; i<args.length; i++)  
            tong += Integer.parseInt(args[i]);  
        System.out.print("Tong = " + tong);  
    }  
}
```



Một số lớp cơ sở - Lớp Character

- ❖ Cung cấp một số phương thức:
 - `static boolean` `isUppercase(char ch)`
 - `static boolean` `isLowercase(char ch)`
 - `static boolean` `isDigit(char ch)`
 - `static boolean` `isLetter(char ch)`
 - `static boolean` `isLetterOrDigit(char ch)`
 - `static char` `toUpperCase(char ch)`
 - `static char` `toLowerCase(char ch)`



Một số lớp cơ sở - Lớp Math

❖ Hằng số:

- `Math.E`
- `Math.PI`

❖ Các phương thức:

- `type abs(type)`
- `double ceil(double), double floor(double)`
- `int round(float), long round(double)`
- `type max(type, type), type min(type, type)`
- `double random()`: Sinh số ngẫu nhiên trong đoạn `[0.0, 1.0]`



Một số lớp cơ sở - Lớp Math

❖ Lũy thừa:

- `double pow(double, double)`
- `double log(double)`
- `double sqrt(double)`

❖ Lượng giác:

- `double sin(double)`
- `double cos(double)`
- `double tan(double)`



Một số lớp cơ sở - Lớp Arrays

- ❖ Nằm trong gói *java.util*
- ❖ Cung cấp 4 phương thức static để làm việc với mảng
 - *fill()*: khởi tạo các phần tử của mảng với một giá trị như nhau
 - *sort()*: sắp xếp mảng
 - *equals()*: so sánh hai mảng
 - *binarySearch()*: tìm kiếm nhị phân trên mảng đã sắp xếp



Một số lớp cơ sở - Lớp String

❖ Khởi tạo:

- `String (String)`
- `String (StringBuffer)`
- `String (byte[])`
- `String (char[])`
-

❖ Phương thức:

- `int length()`: kích thước của chuỗi
- `char charAt(int index)`: ký tự ở vị trí index
- `boolean equals(String)`
- `boolean equalsIgnoreCase(String)`
- `boolean startsWith(String)`
- `boolean endsWith(String)`
- `int compareTo(String)`



Một số lớp cơ sở - Lớp String

❖ Phương thức chuyển đổi:

- `String toUpperCase()`
- `String toLowerCase()`

❖ Ghép chuỗi:

- `String concat(String)`
- Toán tử “+”

❖ Trích chuỗi:

- `String trim()`: loại bỏ ký tự trắng
- `String substring(int startIndex)`
- `String substring(int startIdx, int endIdx)`



Một số lớp cơ sở - Lớp String

❖ Phương thức tìm kiếm:

- `int` `indexOf(char)`
- `int` `indexOf(char ch, int from)`
- `int` `indexOf(String)`
- `int` `indexOf(String s, int from)`
- `int` `lastIndexOf(char)`
- `int` `lastIndexOf(char, int)`
- `int` `lastIndexOf(String)`
- `int` `lastIndexOf(String, int)`

❖ Phương thức thay thế:

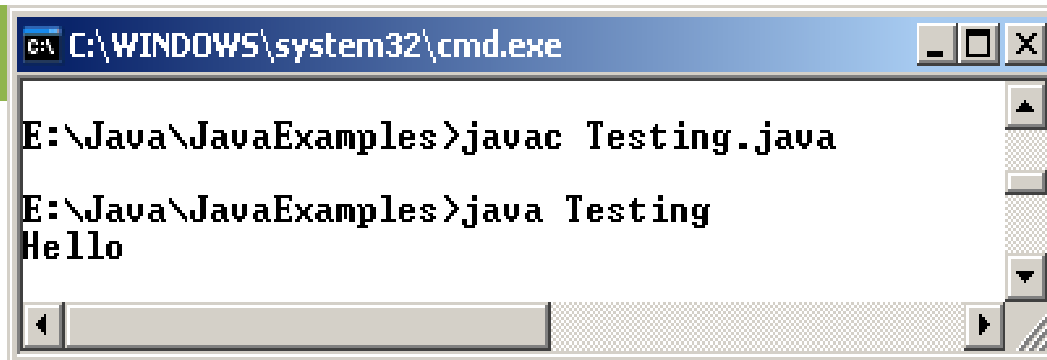
- `String` `replace(char ch, char new_ch)`

Một số lớp cơ sở - Lớp String

- ❖ Strings in Java once created **cannot be changed directly**

```
class Testing{  
    public static void main(String[] args){  
        String str = "Hello";  
        str.concat("And Goodbye");  
        System.out.println(str);  
    }  
}
```

Output



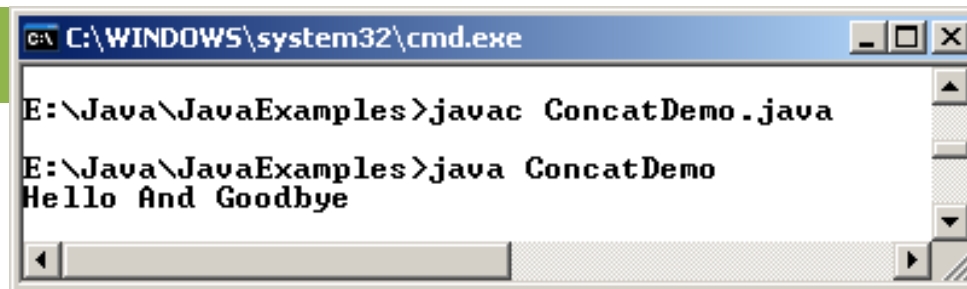
```
C:\WINDOWS\system32\cmd.exe  
E:\Java\JavaExamples>javac Testing.java  
E:\Java\JavaExamples>java Testing  
Hello
```

Một số lớp cơ sở - Lớp StringBuffer

- ❖ A StringBuffer is used to represent a string that **can be modified**.

```
class ConcatDemo{  
    public static void main(String[] args){  
        String str = "Hello";  
        StringBuffer sbObj = new StringBuffer(str);  
        str = sbObj.append(" And Goodbye").toString();  
        System.out.println(str);  
    }  
}
```

Output



```
C:\WINDOWS\system32\cmd.exe  
E:\Java\JavaExamples>javac ConcatDemo.java  
E:\Java\JavaExamples>java ConcatDemo  
Hello And Goodbye
```



Một số lớp cơ sở - Lớp StringBuffer

❖ Khởi tạo:

- `StringBuffer(String)`
- `StringBuffer(int length)`
- `StringBuffer()`: đặt kích thước mặc định 16

❖ Các phương thức:

- `int length()`, `void setLength(int length)`
- `char charAt(int index)`
- `void setCharAt(int index, char ch)`
- `String toString()`



Một số lớp cơ sở - Lớp StringBuffer

❖ Phương thức thêm xóa:

- append(**String**), ...
- insert(**int** offset, **String** s)
- insert(**int** offset, **char**[] chs), ...
- delete(**int** start, **int** end): xóa xâu con
- delete(**int** index): xóa một ký tự
- reverse(): đảo ngược



Lớp Date

Constructor	Purpose
Date()	Creates a Date using today's date.
Date(long dt)	Creates a Date using the specified number of milliseconds since January 1, 1970.
Date(int year, int month, int day)	Creates a Date using the specified year, month and day.



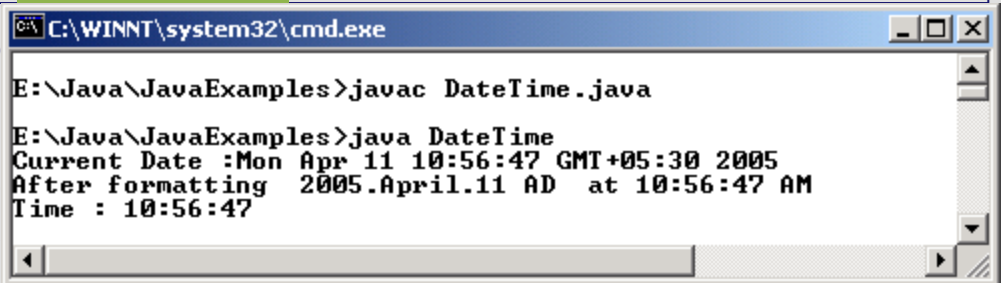
Lớp Date – Ví dụ

```
strDate = objDate.toString();
System.out.println("Current Date :" + strDate);
System.out.println("After formatting " +
                    sdfFormat.format(objDate));

//Extract GMT time
strTime = strDate.substring(11, ( strDate.length()-4));
//Extract the time in hours, minutes, seconds
strTime = "Time : " + strTime.substring(0,8);
System.out.println(strTime);
}
}
```

Output

```
sdfFormat = new Simple
                'at'
//current date is obtained
Date objDate = new Date
```



```
C:\WINNT\system32\cmd.exe
E:\Java\JavaExamples>javac DateTime.java
E:\Java\JavaExamples>java DateTime
Current Date :Mon Apr 11 10:56:47 GMT+05:30 2005
After formatting  2005.April.11 AD  at 10:56:47 AM
Time : 10:56:47
```

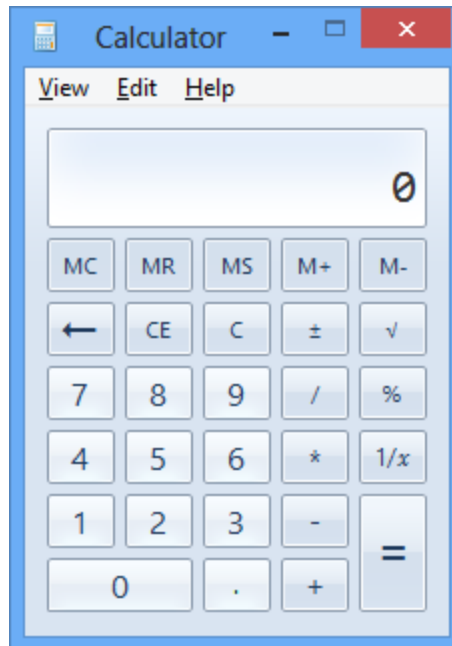


Lớp Calendar

```
System.out.println("Minute : " +  
                    objCalendar.get(Calendar.MINUTE));  
System.out.println("Second : " +  
                    objCalendar.get(Calendar.SECOND));  
//Add 30 minutes to current time,  
//then display date and time  
objCalendar.add(Calendar.MINUTE,30);  
Date objDate = objCalendar.getTime();  
System.out.println("\nDate and Time after adding 30  
                    minutes to current time:\n");  
System.out.println(objDate);  
}  
}  
  
objCalendar.get(Calendar.HOUR);
```

Lớp Runtime

- ❖ Used for **memory management** and **executing additional processes**.
- ❖ We can determine memory allocation details by using **totalMemory()** and **freeMemory()** methods





Nhập/ Xuất từ bàn phím

- ❖ Dùng lớp Scanner để nhập dữ liệu từ bàn phím trong Java.
- ❖ Nhập dữ liệu từ bàn phím sử dụng lớp JOptionPane.



Nhập/ Xuất từ bàn phím

Dùng lớp Scanner để nhập dữ liệu từ bàn phím trong Java.

- ❖ Scanner là một lớp mới có hình như là từ java 6 và có nhiều cái tiến bộ hơn hẳn (BufferedReader, JOptionPane). Scanner vì phân loại đc dữ liệu mà người dùng nhập vào. Chẳng hạn nó có thể đọc kiểu int hay float, double,..



Nhập/ Xuất từ bàn phím

```
import java.util.Scanner;

public class GetInputFromKeyboard {

    public static void main(String[] args) {
        // Tạo một đối tượng Scanner
        Scanner scanIn = new Scanner(System.in);
        // Biến name
        String name = "";
        // Biến age
        int age = 0;
        // Tiến hành đọc từ bàn phím, ấn phím enter để kết thúc
        try {
            System.out.println("Hãy nhập tên của bạn:");
            name = scanIn.nextLine();
            System.out.println("Hãy nhập tuổi của bạn:");
            age = scanIn.nextInt();
        } catch (Exception e) {
            System.out.println("Error!");
        }
        // hiển thị tên
        System.out.println("Bạn tên là : " + name + ".");
        System.out.println("Tuổi của bạn là : " + age + ".");
    }
}
```



Nhập/ Xuất từ bàn phím

- ❖ Mỗi lần các bạn muốn nhận dữ liệu nhập từ bàn phím của người dùng, bạn chỉ cần khai báo 1 đối tượng Scanner và truyền vào System.in. Sau đó các bạn gọi phương thức `nextLine()` của đối tượng Scanner thì Java sẽ tự động yêu cầu người dùng nhập dữ liệu và dùng phím enter để kết thúc nhập.
- ❖ **Một số phương thức của lớp Scanner**
- ❖ Đây là một số phương thức của lớp Scanner, nó sẽ trả về tương ứng kiểu dữ liệu mà bạn gọi:
 1. **`nextInt()` : trả về kiểu int**
 2. **`nextFloat()` : trả về kiểu float**
 3. **`nextBoolean()` : trả về kiểu boolean**
 4. **`nextByte()` : trả về kiểu byte**
 5. **`nextLine()` : trả về kiểu String.**



Nhập/ Xuất từ bàn phím

Nhập dữ liệu từ bàn phím sử dụng 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.

Ví dụ

```
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);
    }
}
```



Nhập/ Xuất từ bàn phím

Nhập dữ liệu từ bàn phím sử dụng lớp JOptionPane.

❖ Một số phương thức của JOptionPane

1. `showConfirmDialog()` : Hiển thị một câu hỏi lựa chọn giống như yes no cancel
2. `showInputDialog()` : Hiển thị box nhập
3. `showMessageDialog()` : Báo cho người dùng một sự kiện vừa xảy ra.



Bài tập

1. Viết chương trình tạo một mảng số nguyên ngẫu nhiên có n phần tử (n được nhập từ bàn phím, $n > 0$), mỗi phần tử có giá trị từ $0 \rightarrow 100$.
 - a) Xuất giá trị các phần tử của mảng.
 - b) Tìm phần tử có giá trị nhỏ nhất.
 - c) Đếm số lượng các phần tử là số nguyên tố.
 - d) Đếm số phần tử có tổng các chữ số lớn hơn 10.
 - e) Sắp xếp mảng tăng dần/giảm dần
 - f) Sắp xếp số chẵn giảm dần, số lẻ tăng dần
 - g) Tìm phần tử có giá trị x



Bài tập

2. Cho ma trận số nguyên cấp $n \times m$. Cài đặt các hàm thực hiện các chức năng sau:
- a) Nhập ma trận.
 - b) In ma trận.
 - c) Tìm phần tử nhỏ nhất.
 - d) Tìm phần tử lẻ lớn nhất.
 - e) Tìm dòng có tổng lớn nhất.
 - f) Tính tổng các số không phải là số nguyên tố.



Bài tập

3. Cho ma trận vuông số nguyên cấp n . Cài đặt các hàm thực hiện các chức năng sau:
- a) Nhập ma trận.
 - b) In ma trận.
 - c) Tổng các phần tử thuộc tam giác trên.
 - d) Tổng các phần tử thuộc tam giác dưới.
 - e) Kiểm tra xem ma trận có đối xứng qua đường chéo chính?



Các gói chuẩn của Java

- ❖ java.lang
- ❖ java.applet
- ❖ java.awt
- ❖ java.io
- ❖ java.util
- ❖ java.net
- ❖ java.awt.event
- ❖ java.rmi
- ❖ java.security
- ❖ java.sql



Từ khóa của Java

Danh sách từ khóa thường gặp

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



Hỏi & đáp

