CHƯƠNG 1: TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH JAVA (4,3,14)

Lý thuyết

1. Java và ứng dụng Console

Ứng dụng Console là ứng dụng nhập xuất ở chế độ văn bản tương tự như màn hình Console của hệ điều hành MS-DOS.

Các ứng dụng kiểu Console thường được dùng để minh họa các ví dụ cơ bản liên quan đến cú pháp ngôn ngữ, các thuật toán, và các chương trình ứng dụng không cần thiết đến giao diện người dùng đồ họa.

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("\nHello World");
    }
}
```

2. Java và ứng dụng Applet

Java Applet là loại ứng dụng có thể nhúng và chạy trong trang web của một trình duyệt web.

3. Java và phát triển ứng dụng Desktop dùng AWT và JFC

Việc phát triển các chương trình ứng dụng có giao diện người dùng đồ họa trực quan giống như những chương trình được viết dùng ngôn ngữ lập trình VC++ hay Visual Basic đã được java giải quyết bằng thư viện AWT và JFC. JFC là thư viện rất phong phú và hỗ trợ mạnh mẽ hơn nhiều so với AWT. JFC giúp cho người lập trình có thể tạo ra một giao diện trực quan của bất kỳ ứng dụng nào.

Java và phát triển ứng dụng Web

Trang web thông tin nhà đất nổi tiếng ở TPHCM đó là: http://www. nhadat.com/ được xây dựng dựa trên nền công nghệ java.

Bạn có thể tìm hiểu chi tiết hơn về công nghệ J2EE tạo địa chỉ: http://java.sun.com/j2ee/

4. Cấu trúc chương trình HelloWordApp

Phương thức main(): là điểm bắt đầu thực thi một ứng dụng (chương trình chính). Mỗi ứng dụng Java phải chứa một phương thức main có dạng như sau:

public static void main(String[] args)

Phương thức main chứa ba bổ từ đặc tả sau:

- **public**: chỉ ra rằng phương thức main có thể được gọi bởi bất kỳ đối tượng nào.
- **static**: chỉ ra rằng phương thức main là một phương thức lớp.
- **void**: chỉ ra rằng phương thức main sẽ không trả về bất kỳ một giá trị nào.

Ngôn ngữ Java hỗ trợ ba kiểu chú thích sau:

/* text */

// text

/** documentation */. Công cụ javadoc trong bộ JDK sử dụng chú thích này để chuẩn bị cho việc tự động phát sinh tài liệu.

- Dấu mở và đóng ngoặc nhọn "{" và "}": là bắt đầu và kết thúc 1 khối lệnh.
- Dấu chấm phẩy ";" kết thúc 1 dòng lệnh

5. Tạo chương trình nguồn HelloWordApp

- Khởi động chương trình soạn thảo (Notepad, Notpad++,…) và gõ đoạn mã sau

/*Viết chương trình in dòng HelloWorld lên màn hình Console*/

```
class HelloWorldApp{
     public static void main(String[] args){
          System.out.println("HelloWorld"); //In dong chu HelloWorld
     }
}
```

Lưu lại với tên HelloWorldApp.java

6. Biên dịch tập tin nguồn HelloWordApp

- Mở cửa sổ Command Prompt.
- Chuyển đến thư mục chứa tập tin nguồn vừa tạo ra.
- Thực hiện câu lệnh: javac HelloWordApp.java.

7. Chạy chương trình HelloWordApp

- Tại dấu nhắc gõ lệnh: java HelloWordApp
- Nếu chương trình đúng bạn sẽ thấy dòng chữ HelloWord trên màn hình Console.

8. *Sử dụng phương thức/biến của lớp*

Cú pháp: Tên_lớp.Tên_biến

hoặc Tên_lớp.Tên_phương_thức(…)

Bài tập

Bài 1. Download JDK tại địa chỉ: http://java.sun.com/javase/downloads/index.jsp. Hãy tiến hành cài đặt JDK và thiết lập biến môi trường.

Bài 2. Mở NotePad và soạn thảo chương trình in ra câu thông báo "Hello Java". Lưu lại với tên là HelloJava.java; Mở Command Prompt, thực hiện biên dịch và thông dịch bằng lệnh: **javac [options] sourcecodename.java** và **java [options] classname**

Bài 3. Download và cài đặt công cụ lập trình java: hoặc JCreator hoặc JDeveloper hoặc Netbean hoặc Eclipse hoặc bất kỳ công cụ nào anh (chị) thích.

Bài 4. Viết chương trình java in ra dòng chữ họ tên của anh (chị).

CHƯƠNG 2: NGÔN NGỮ JAVA (8,6,28)

Lý thuyết

1. Biến

Biến là vùng nhớ dùng để lưu trữ các giá trị của chương trình. Mỗi biến gắn liền với một kiểu dữ liệu và một định danh duy nhất gọi là tên biến.

Tên biến thông thường là một chuỗi các ký tự (Unicode), ký số. Tên biến phải bắt đầu bằng một ký tự chữ cái, một ký tự _ hay ký tự $.

Tên biến không được trùng với các từ khóa (xem phụ lục các từ khóa trong java).

Trong java, biến có thể được khai báo ở bất kỳ nơi đâu trong chương trình.

Cách khai báo biến

<kiểu_dữ_liệu> <tên_biến>;

<kiểu_dữ_liệu> <tên_biến> = <giá_trị>;

Gán giá trị cho biến

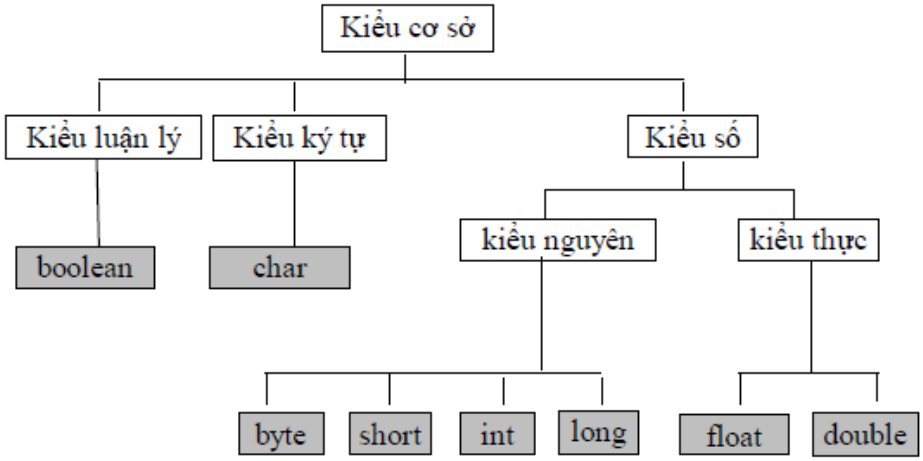
<tên_biến> = <giá_trị>;

Biến công cộng (toàn cục): là biến có thể truy xuất ở khắp nơi trong chương trình, thường được khai báo dùng từ khóa public, hoặc đặt chúng trong một class.

Biến cục bộ: là biến chỉ có thể truy xuất trong khối lệnh nó khai báo.

Lưu ý: Trong ngôn ngữ lập trình java có phân biệt chữ in hoa và in thường.

2. *Các kiểu dữ liệu cơ sở*



Kiểu	Kích thước (bytes)	Giá trị min	Giá trị max	Giá trị mặc định
byte	1	-256	255	0
short	2	-32768	32767	0
int	4	-2^{31}	$2^{31} - 1$	0
long	8	-2^{63}	$2^{63} - 1$	0L
float	4			0.0f
double	8			0.0d

3. *Hằng*

- Hằng là một giá trị bất biến trong chương trình
- Tên hằng được đặt theo qui ước giống như tên biến.
- Hằng số nguyên: trường hợp giá trị hằng ở dạng long ta thêm vào cuối chuỗi số chữ "l" hay "L". (ví dụ: 1L)
- 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".
- Hằng Boolean: java có 2 hằng boolean là true, false.
- Hằng ký tự: là một ký tự đơn nằm giữa nằm giữa 2 dấu ngoặc đơn.

4. *Lệnh, khối lệnh trong java*

```
{ // khối 1
    { // khối 2
        lệnh 2. 1
        lệnh 2. 2
```

…

} // kết thúc khối lệnh 2

lệnh 1. 1 lệnh 1. 2

…

} // kết thúc khối lệnh 1

{ // bắt đầu khối lệnh 3

// Các lệnh thuộc khối lệnh 3

// …

} // kết thúc thối lệnh 3

5. *Toán tử và biểu thức*

❖ *Toán tử số học*

Các toán tử số học được sử dụng trong các biểu thức toán học theo cách tương tự như chúng được sử dụng trong đại số học.

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

❖ *Toán tử trên bit*

Java định nghĩa một số toán tử thao tác bit có thể được áp dụng cho các kiểu giá trị integer, long, int, short, char, và byte.

Toán tử thao tác bit làm việc trên các bit. Giả sử nếu a = 60 và b = 13, thì trong định dạng nhị phân chúng sẽ như sau:

a = 0011 1100

b = 0000 1101

Kết quả:

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

Giả sử biến A giữ giá trị 60 và biến B giữ 13 thì khi đó:

Toán tử	Miêu tả	Ví dụ
&	Toán tử Và nhị phân sao chép một bit tới kết quả nếu nó tồn tại trong cả hai toán hạng	(A & B) sẽ cho kết quả 12, hay là 0000 1100
|	Toán tử Hoặc nhị phân sao chép một bit tới kết quả nếu nó tồn tại trong một hoặc hai toán hạng	(A | B) sẽ cho kết quả 61, hay là 0011 1101
^	Toán tử Hoặc loại trừ nhị phân sao chép bit nếu nó được thiết lập trong một toán hạng nhưng không phải trong cả hai	(A ^ B) sẽ cho kết quả 49, hay là 0011 0001
~	Toán tử đảo bit là toán tử một ngôi. Đảo bít 1 thành 0 và ngược lại	(~A) sẽ cho kết quả -61, hay là 1100 0011
<<	Toán tử dịch trái. Giá trị toán hạng trái được dịch chuyển sang trái bởi số các bit được xác định bởi toán hạng bên phải.	A << 2 sẽ cho kết quả 240, hay là 1111 0000
>>	Toán tử dịch phải. Giá trị toán hạng trái được dịch chuyển sang phải bởi số các bit được xác định bởi toán hạng bên phải	A >> 2 sẽ cho kết quả 15, hay là 1111
>>>	Toán tử dịch phải và điền 0 vào chỗ trống	A >>>2 sẽ cho kết quả 15, hay là 0000 1111

❖ *Toán tử quan hệ*

Giả sử biến A giữ giá trị 10, biến B giữ giá trị 20, thì:

Toán tử	**Miêu tả**	**Ví dụ**
==	Kiểm tra nếu giá trị của hai toán hạng có cân bằng hay không, nếu có thì điều kiện là true.	(A == B) là không true.
!=	Kiểm tra nếu giá trị hai toán hạng là cân bằng hay không, nếu không cân bằng, thì điều kiện là true	(A != B) là true.
>	Kiểm tra nếu toán hạng trái có lớn hơn toán hạng phải hay không, nếu có thì điều kiện là true	(A > B) là không true.
<	Kiểm tra nếu toán hạng phải có lớn hơn toán hạng trái hay không, nếu có thì điều kiện là true	(A < B) là true.
>=	Kiểm tra nếu toán hạng trái có lớn hơn hoặc bằng toán hạng phải hay không, nếu có thì điều kiện là true	(A >= B) là không true.
<=	Kiểm tra nếu toán hạng phải có lớn hơn hoặc bằng toán hạng trái hay không, nếu có thì điều kiện là true	(A <= B) là true.

❖ *Toán tử logic trong Java*

Giả sử biến A giữ true và biến B giữ false thì khi đó:

Toán tử	Miêu tả	Ví dụ
&&	Toán tử Và logic. Nếu cả hai toán hạng là khác không, thì khi đó điều kiện là true	(A && B) là false.
||	Toán tử Hoặc logic. Nếu một trong hai toán tử khác 0, thì điều kiện là true	(A || B) là true.
!	Toán tử Phủ định logic. Sử dụng để đảo ngược lại trạng thái logic của toán hạng đó. Nếu điều kiện toán hạng là true thì phủ định nó sẽ là false	!(A && B) là true.

❖ *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 điều kiện*

Cú pháp: <**điều kiện**> ? <**biểu thức 1**> : < **biểu thức 2**>

Nếu điều kiện đúng thì có giá trị, hay thực hiện <**biểu thức 1**>, còn ngược lại là <**biểu thức 2**>.

Trong đó:

<**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) ? 30 : 40;

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

❖ *Thứ tự ưu tiên*

Thứ tự ưu tiên tính từ trái qua phải và từ trên xuống dưới.

Cao nhất			
()	[]	.	
++	--	~	!
*	/	%	
+	-		
>>	*>>> (dịch phải và điền 0 vào bit trống)*	<<	
>	>=	<	<=
==	!=		
&			
^			
|			
&&			
||			
?:			
=	<toántử>=		
Thấp nhất			

6. *Cấu trúc điều khiển*

❖ *Cấu trúc if…else*

Dạng 1:

```
if (<điều_kiện>) {
```

<khối_lệnh>;

}

Dạng 2:

if (<điều_kiện>) {

<khối _lệnh 1>;

}

else {

<khối _lệnh 2>;

}

❖ *Cấu trúc switch … case*

switch (<biến>) {

case <giátrị_1>: <khối_lệnh_1>;

break;

….

case <giátrị_n>: <khối_lệnh_n>;

break;

default: <khối lệnh default>;

}

❖ *Cấu trúc lặp*

Dạng 1: while(…)

while (điều_kiện_lặp) {

<khối _lệnh>;

}

Dạng 2: do { … } while;

do {

<khối_lệnh>;

} while (<điều_kiện>);

Dạng 3: for (…)

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

❖ *Cấu trúc lệnh nhảy (jump)*

Lệnh **break**: trong cấu trúc switch chúng ta dùng câu lệnh break để thoát thỏi cấu trúc switch trong cùng chứa nó. Tương tự như vậy, 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).

Nhãn (**label**):

Không giống như C/C++, Java không hỗ trợ lệnh goto để nhảy đến 1 vị trí nào đó của chương trình.java dùng kết hợp nhãn (label) với từ khóa break và continue để thay thế cho lệnh goto.

Ví dụ:

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

7. Mảng

Khai báo:

```
        <kiểu dữ liệu>      <tên mảng>[];
hoặc    <kiểu dữ liệu>[]    <tên mảng>;
```

Ví dụ:

```
        int     arrInt[];
hoặc    int[]   arrInt;
        int[]   arrInt1, arrInt2, arrInt3;
```

Cấp phát bộ nhớ: *int arrInt = **new** int[100];*

Khởi tạo mảng:

```
int      arrInt[] = {1, 2, 3};
char     arrChar[] = {'a', 'b', 'c'};
String arrStrng[] = {"ABC", "EFG", 'GHI'};
```

Bài tập

Bài 1. Hãy xác định đâu là tên biến đúng, đâu là tên biến sai?

Area; radius; 5a; readInteger; hoc sinh; sinhVien1; sinhVien2; 1gia tri; if; hoTen; _giaTri; try; a ; _a: A; _b; _B; $d

Bài 2. Khai báo và gán giá trị cho 2 biến có kiểu số nguyên. In ra màn hình giá trị tổng, hiệu, thương của 2 biến.

Bài 3. Khai báo hằng PI = 3.14 kiểu số thực, với biến r là bán kính đường tròn – kiểu số thực được gán trong chương trình. Hãy viết chương trình tính diện tích và chu vi hình tròn, in kết quả ra màn hình.

Bài 4. Hãy viết chương trình tính tổng các chữ số của một số nguyên bất kỳ.

Ví dụ: Số 8545604 có tổng các chữ số là: 8+5+4+5+6+0+4= 32.

Bài 5. Khai báo các biến a, b, c kiểu số nguyên. Gán giá trị cho các biến. Tìm phần nguyên khi chia các số này cho 2, tìm phần dư khi chia các số này cho 3, in kết quả ra màn hình. Tăng giá trị a, b, c mỗi biến lên 1, in giá trị 3 số ra màn hình

Bài 6. Giải phương trình 2ax+b = 8c với a, b, c là số thực – được nhập từ bàn phím. Sau khi tính x, in kết quả ra màn hình, ép kiểu x về số nguyên, in kết quả ép kiểu của x ra màn hình.

Bài 7. Cho a, b, c là độ dài 3 cạnh tam giác (a, b, c >0 được nhập từ bàn phím). Tính diện tích s của tam giác, in kết quả ra màn hình. (Gợi ý: Sử dụng công thức Heron với S: diện tích; p: nửa chu vi)

Bài 8. Với a, b, c là các số thực được nhập từ bàn phím. Hãy kiểm tra xem đây có phải là độ dài 3 cạnh của 1 tam giác không, nếu có thì đó là của loại tam giác nào?

Bài 9. Giải phương trình bậc 2: ax2 + bx + c = 0 với a, b, c là các số thực được nhập từ bàn phím.in kết quả các nghiệm ra màn hình (Nếu a = 0 thì ra màn hình đây không phải là phương trình bậc 2 và in nghiệm)

Bài 10. Viết chương trình khai báo và nhập biến nguyên a một giá trị bất kỳ. Nếu a = 1 thì in ra màn hình là "Chủ nhật", a = 2 thì in ra "Thứ Hai", ... a = 7 thì in ra "Thứ Bảy". Nếu a không trong khoảng [1 ; 7] thì báo "Bạn đã sai, chỉ được nhập số nguyên từ 1 tới 7"

Bài 11. Khai báo và nhập giá trị cho biến a là 1 trong các số nguyên từ 0 đến 9. Hãy in ra tên các chữ số nhập vào dưới hình thức tiếng Anh. Ví dụ a bằng 1 thì chương trình chạy sẽ in ra "1 đọc là One"; a = 2 thì in ra "2 đọc là Two"…

Bài 12. Khai báo và nhập giá trị cho 2 biến nguyên "thang" và "nam" . Yêu cầu "thang" thuộc tập hợp [1..12], năm không được âm. Nếu nhập sai tháng thì báo "Bạn đã nhập sai tháng", nếu nhập sai năm thì báo "Bạn đã nhập sai năm". Khi 1 trong 2 thông tin bị sai thì thoát chương trình, ngược lại:

+ Nếu năm đó là năm nhuận thì in ra thông báo: Đây là năm nhuận. không thì báo ra là năm thường.

+ Dựa vào thông tin năm đó là năm nhuận hay không và giá trị của tháng đó là tháng nào để báo ra tháng đó có bao nhiêu ngày.

Bài 13. Hãy khai báo và nhập dữ liệu cho biến a là 1 trong các số từ 1 đến 9.in ra màn hình bảng cửu chương tương ứng với số vừa nhập.

Bài 14. In ra toàn bộ bảng cửu chương.

Bài 15. Sắp xếp 1 dãy n số nguyên nhập vào từ bàn phím theo thứ tự tăng dần.

Bài 16. Nhập các phần tử của 2 ma trận cùng số hàng, số cột vào từ bàn phím. Tính tổng, hiệu, tích 2 ma trận.in kết quả ra màn hình.

Bài 17. Nhập vào một chuỗi các số nguyên (mỗi số cách nhau khoảng trắng). Cho biết:

a. Chuỗi vừa nhập có bao nhiêu số, đó là những số nào.
b. In ra các số là số nguyên tố.

Bài 18. Nhập vào một chuỗi các số thực (mỗi số cách nhau dấu chấm phẩy). Cho biết:

a. Có bao nhiêu số vừa nhập.
b. Hãy làm tròn các số (VD: 2.3 ⇨ làm tròn là 2; 2.5 ⇨ làm tròn là 3).

Bài 19. Một số được gọi là số thuận nghịch độc nếu ta đọc từ trái sang phải hay từ phải sang trái số đó ta vẫn nhận được một số giống nhau. Hãy liệt kê tất cả các số thuận nghịch độc có sáu chữ số (Ví dụ số: 558855).

Bài 20. Viết chương trình liệt kê tất cả các số nguyên tố có 5 chữ số sao cho tổng của các chữ số trong mỗi số nguyên tố đều bằng S cho trước.

Bài 21. Viết chương trình nhập vào vào ma trận A có n dòng, m cột, các phần tử là những số nguyên lớn hơn 0 và nhỏ hơn 100 được nhập vào từ bàn phím. Thực hiện các chức năng sau:

a. Tìm phần tử lớn nhất của ma trận cùng chỉ số của số đó.
b. Tìm và in ra các phần tử là số nguyên tố của ma trận (các phần tử không nguyên tố thì thay bằng số 0).
c. Sắp xếp tất cả các cột của ma trận theo thứ tự tăng dần và in kết quả ra màn hình.

Bài 22. Viết chương trình nhập vào vào mảng A có n phần tử, các phần tử là những số nguyên lớn hơn 0 và nhỏ hơn 100 được nhập vào từ bàn phím. Thực hiện các chức năng sau:

a. Tìm phần tử lớn nhất và lớn thứ 2 trong mảng cùng chỉ số của các số đó.
b. Sắp xếp mảng theo thứ tự giảm dần .
c. Nhập một số nguyên x và chèn x vào mảng A sao cho vẫn đảm bảo tính sắp xếp giảm dần.

Bài 23. Viết chương trình thực hiện chuẩn hoá một xâu ký tự nhập từ bàn phím (loại bỏ các dấu cách thừa, chuyển ký tự đầu mỗi từ thành chữ hoa, các ký tự khác thành chữ thường)

Bài 24. Viết chương trình thực hiện nhập một xâu ký tự và tìm từ dài nhất trong xâu đó. Từ đó xuất hiện ở vị trí nào? (Chú ý. nếu có nhiều từ có độ dài giống nhau thì chọn từ đầu tiên tìm thấy).

CHƯƠNG 3: HƯỚNG ĐỐI TƯỢNG TRONG JAVA (6,9,30)

Lý thuyết

1. ***Lớp***
   - Khai báo lớp:

     class <ClassName> {

     <kiểu dữ liệu> <field_1>; <kiểu dữ liệu> <field_2>;

     constructor

     method_1 method_2

     }
   - Tạo đối tượng của lớp

   ClassName objectName = **new** *ClassName()*;

   - Thuộc tính của lớp

   Vùng dữ liệu (fields) hay thuộc tính (properties) của lớp được khai báo bên trong lớp như sau:

   class <ClassName> {

   // khai báo những thuộc tính của lớp

   <tiền tố> <kiểu dữ liệu> field1;

   // …

   }

   Để xác định quyền truy xuất của các đối tượng khác đối với vùng dữ liệu của lớp người ta thường dùng 3 tiền tố sau:

   - **public**: có thể truy xuất từ tất cả các đối tượng khác
   - **private**: một lớp không thể truy xuất vùng private của 1 lớp khác.
   - **protected**: vùng protected của 1 lớp chỉ cho phép bản thân lớp đó và những lớp dẫn xuất từ lớp đó truy cập đến.
2. ***Hàm - Phương thức lớp (Method)***
   - Khai báo:

   <Tiền tố> <kiểu trả về> <Tên phương thức> (<danh sách đối số>) {

   <khối lệnh>;

   }

   Để xác định quyền truy xuất của các đối tượng khác đối với các phương thức của lớp người ta thường dùng các tiền tố sau:

- **public**: phương thức có thể truy cập được từ bên ngoài lớp khai báo.
- **protected**: có thể truy cập được từ lớp khai báo và những lớp dẫn xuất từ nó.
- **private**: chỉ được truy cập bên trong bản thân lớp khai báo.
- **static**: phương thức lớp dùng chung cho tất cả các thể hiện của lớp, có nghĩa là phương thức đó có thể được thực hiện kể cả khi không có đối tượng của lớp chứa phương thức đó.
- **final**: phương thức có tiền tố này không được khai báo chồng ở các lớp dẫn xuất.
- **abstract**: phương thức không cần cài đặt (không có phần source code), sẽ được hiện thực trong các lớp dẫn xuất từ lớp này.
- **synchoronized**: dùng để ngăn các tác động của các đối tượng khác lên đối tượng đang xét trong khi đang đồng bộ hóa. Dùng trong lập trình miltithreads.

<kiểu trả về>: có thể là kiểu void, kiểu cơ sở hay một lớp.

<Tên phương thức>: đặt theo qui ước giống tên biến.

<danh sách thông số>: có thể rỗng

3. Khởi tạo một đối tượng (Constructor)

Constructor phải có **cùng tên với lớp** và được gọi đến dùng từ khóa **new**.

Ví dụ:

```
public class xemay {
        // ...
        public xemay() {}
        public xemay(String s_nhasx, String s_model, f_chiphisx, int i_thoigiansx,
int i_so); {
                nhasx = s_nhasx;
                model = s_model;
                chiphisx = f_chiphisx;
                thoigiansx = i_thoigiansx;
                so = i_so;
                // hoặc
                // this. nhasx = s_nhasx;
                // this. model = s_model;
                // this. chiphisx = f_chiphisx;
                // this. thoigiansx = i_thoigiansx;
                // this. so = i_so;
        }
}
```

4. Biến this

Biến this được sử dụng trong khi chạy và tham khảo đến bản thân lớp chứa nó.

Ví dụ:

```
<tiền tố> class A {
        <tiền tố> int <field_1>;
        <tiền tố> String <field_2>; // Contructor của lớp A
        public A(int par_1, String par_2) {
                this. field_1 = par_1;
                this. field_2 = par_2;
        }
        <tiền tố> <kiểu trả về> <method_1>(){
                // ...
        }
        <tiền tố> <kiểu trả về> <method_2>(){
                this. method_1()
                // ...
        }
}
```

5. Khai báo chồng phương thức (overloading method)

Việc khai báo trong một lớp nhiều phương thức có cùng tên nhưng khác tham số (khác kiểu dữ liệu, khác số lượng tham số) gọi là khai báo chồng phương thức (overloading method).

Ví dụ:

```
public class xemay {

        // khai báo fields ...

        public float tinhgiaban(){

        return 2 * chiphisx;

        }

        public float tinhgiaban(float huehong) {

                return (2 * chiphisx + huehong);

        }

}
```

6. Tính kế thừa

Dùng từ khóa **extends** để chỉ lớp dẫn xuất.

```
class A extends B {
        // ...
}
```

7. Giao diện

Interface được khai báo như một lớp. Nhưng các thuộc tính của interface là các hằng (khai báo dùng từ khóa final) và các phương thức của giao tiếp là trừu tượng (mặc dù không có từ khóa abstract). Trong các lớp có cài đặt các interface ta phải tiến hành cài đặt cụ thể các phương thức này.

Ví dụ:

```
public interface sanpham {
    static final String nhasx = "Honda VN";
    static final String dienthoai = "08-8123456";
    public int gia(String s_model);
}
// khai báo 1 lớp có cài đặt interface
public class xemay implements sanpham {
    public int    gia(String s_model) {
        if (s_model. equals("2005"))
            return (2000);
        else
            return (1500);
    }
    public String chobietnhasx() { return (nhasx); }
}
```

Bài tập

Bài 1. Xây dựng lớp HocSinh gồm 3 thuộc tính: Họ tên, lớp, điểm trung bình. Các phương thức bao gồm: nhập và xuất thông tin. Tạo 2 đối tượng học sinh a và học sinh b.

Nhập giá trị các thuộc tính cho 2 đối tượng. So sánh điểm trung bình giữa 2 đối tượng tìm ra người có điểm cao hơn in ra màn hình.

Bài 2. Xây dựng lớp sinh viên gồm họ và tên, năm sinh, điểm toán, lý , hóa. Các phương thức của lớp: nhập, hiển thị thông tin, tính điểm trung bình. Khai báo một danh sách n sinh viên (với n>0, nhập từ bàn phím). Nhập dữ liệu cho danh sách và sắp xếp danh sách theo thứ tự tăng dần của điểm trung bình. Tìm và in ra thông tin của sinh viên có điểm trung bình lớn nhất.

Bài 3. Xây dựng lớp hình chữ nhật có thuộc tính là chiều dài, chiều rộng. Các phương thức để nhập, xuất thông tin, tính chu vi và diện tích hình chữ nhật.

Khai báo một danh sách n hình chữ nhật (n>0, nhập từ bàn phím). Nhập dữ liệu cho danh sách và cho biết hình chữ nhật nào diện tích lớn nhất.

Bài 4. Xây dựng lớp phân số với hai thuộc tính tử số và mẫu số. Xây dựng các phương thức:

- Hàm tạo, hàm hủy phân số.
- Các phép toán cộng, trừ, nhân, chia phân số.
- Tối giản phân số.

Viết chương trình ứng dụng việc nhập vào một phân số và in ra màn hình dạng tối giản của phân số đó.

Bài 5. Xây dựng lớp hàng hóa gồm: tên hàng, mã hàng, giá bán, số lượng, giờ mua. Nếu mua vào khoảng 8h -> 17h hàng ngày, được giảm giá 5%. Tạo phương thức để nhập, xuất thông tin và phương thức để tính tiền.

Khai báo một danh sách n hàng hóa (n>0, nhập từ bàn phím). Nhập dữ liệu cho danh sách và cho biết tổng tiền.

Bài 6. Xây dựng lớp sinh viên với các thuộc tính: hoTen, namSinh, lop, diemTb.

Tạo 1 constructor với 3 tham số truyền vào; 1 constructor không có tham số truyền vào.

Xây dựng phương thức nhập, xuất thông tin cho lớp.

Viết hàm main minh họa.

Bài 7. Xây dựng lớp Hình có 1 thuộc tính là cạnh, 1 constructor không đối số, 1 phương thức nhập và xuất. Xây dựng phương thức tính diện tích theo yêu cầu sau:

- Có 1 tham số truyền vào, nó sẽ tự hiểu là cần tính diện tích hình vuông
- Có 2 tham số truyền vào, nó tự hiểu là tính diện tích hình chữ nhật
- Có 3 tham số truyền vào, nó tự hiểu là tính diện tích hình tam giác.

Viết hàm main cho phép lựa chọn từ 1-3 tương ứng tính diện tích hình vuông, chữ nhật, tam giác; đồng thời yêu cầu nhập vào độ dài cạnh để tính diện tích. Sau đó in kết quả ra màn hình.

Bài 8. Xây dựng lớp hàng hóa gồm các thuộc tính: tên hàng, giá, số lượng, % giảm giá. Xây dựng constructor không tham số, phương thức nhập và xuất thông tin.

Tạo phương thức tính tiền tổng hóa đơn hàng hóa đó với 2 tham số truyền vào là giá, và số lượng.

Tạo phương thức nạp chồng phương thức trên, có thêm tham số truyền vào nữa là % giảm giá. Tính tiền dựa vào các thông số này.

Viết CTC Tạo danh sách n hàng hóa, nhập thông tin, tính tiền và in tất cả thông tin ra màn hình (bao gồm tổng số tiền)

Bài 9. Tạo package nhân sự, trong có 2 class là sinh viên và giảng viên

Class sinh viên gồm các thuộc tính: họ tên, lớp học, điểm toán, lý, hóa. Các phương thức nhập, xuất thông tin, tính điểm trung bình.

Class giảng viên gồm các thuộc tính: họ tên, năm sinh, chuyên môn, lớp dạy. Các phương thức nhập, xuất thông tin.

Tạo 2 danh sách n sinh viên và m giảng viên (n, m>0, nhập từ bàn phím). Nhập thông tin cho 2 danh sách và thực hiện yêu cầu sau:

- Nhập tên sinh viên. Cho biết những giáo viên nào có dạy sinh viên này.
- Nhập tên giáo viên và lớp dạy.in ra màn hình danh sách sinh viên của lớp mà giáo viên này dạy.
- Cho biết sinh viên có điểm trung bình cao nhất và những giáo viên nào có dạy sinh viên này.

Bài 10. Dùng tính kế thừa khai báo 3 class:

Class "Sinh Viên" gồm các thuộc tính: Họ tên, năm sinh, mã thẻ, tiền học phí còn nợ.
Class "Giảng Viên" gồm các thuộc tính: Họ tên, năm sinh, mã thẻ, tiền lương hàng tháng.

Class "Giám Đốc" gồm các thuộc tính: Họ tên, năm sinh, mã thẻ, tiền tiêu hàng tháng.

Xây dựng các phương thức getter và setter để lấy và gán giá trị cho các thuộc tính.

Xây dựng hàm main thực hiện các công việc sau:

a. Tạo 1 đối tượng Sinh viên, nhập giá trị vào từ bàn phím, in thông tin ra màn hình.

b. Khai báo một danh sách chứa n Giảng viên (n>0, nhập từ bàn phím). Nhập dữ liệu cho danh sách và in ra danh sách đã được sắp xếp tăng dần theo tên.
c. Khai báo một danh sách chứa n Giám đốc (n>0, nhập từ bàn phím) và nhập dữ liệu cho danh sách này. Cho biết những giám đốc nào có tiền tiêu hàng tháng cao nhất trong danh sách vừa nhập.

Bài 11. Một đơn vị sản xuất gồm có các cán bộ là công nhân, kỹ sư, nhân viên.
Mỗi cán bộ cần quản lý lý các thuộc tính: Họ tên, năm sinh, giới tính, địa chỉ.

- Các công nhân cần quản lý: Bậc (công nhân bậc 3/7, bậc 4/7 ...).
- Các kỹ sư cần quản lý: Ngành đào tạo.
- Các nhân viên phục vụ cần quản lý thông tin: công việc.

a. Xây dựng các lớp NhanVien, CongNhan, KySu kế thừa từ lớp CanBo.
b. Xây dựng các hàm để truy nhập, hiển thị thông tin và kiểm tra về các thuộc tính của các lớp.
c. Xây dựng lớp QLCB cài đặt các phương thức thực hiện các chức năng sau:
- Nhập thông tin mới cho cán bộ
- Tìm kiếm theo họ tên
- Hiển thị thông tin về danh sách các cán bộ

Bài 12. Các thí sinh dự thi đại học bao gồm các thí sinh thi khối A, thí sinh thi khối B, thí sinh thi khối C

- Các thí sinh cần quản lý các thuộc tính: Số báo danh, họ tên, địa chỉ, ưu tiên.
- Thí sinh thi khối A thi các môn: Toán, lý, hoá
- Thí sinh thi khối B thi các môn: Toán, Hoá, Sinh
- Thí sinh thi khối C thi các môn: văn, Sử, Địa

a. Xây dựng các lớp để quản lý các thí sinh sao cho sử dụng lại được nhiều nhất.
b. Xây dựng lớp TuyenSinh cài đặt các phương thức thực hiện các nhiệm vụ sau:

- Nhập thông tin về các thí sinh dự thi

- Hiển thị thông tin về một thí sinh

- Tìm kiếm theo số báo danh

Bài 13. Để quản lý hồ sơ học sinh của trường THPT, người ta cần quản lý những thông tin như sau:

- Các thông tin về : lớp, khoá học, kỳ học, và các thông tin cá nhân của mỗihọc sinh.

- Với mỗi học sinh, các thông tin cá nhân cần quản lý gồm có: Họ và tên,

tuổi, năm sinh, quê quán.

a. Hãy xây dựng lớp Nguoi để quản lý các thông tin cá nhân của mỗi học sinh.
b. Xây dựng lớp HSHocSinh (hồ sơ học sinh) để lý các thông tin về mỗi học sinh.
c. Xây dựng các phương thức : nhập, hiển thị các thông tin về mỗi cá nhân.
d. Cài đặt chương trình thực hiện các công việc sau:

- Nhập vào một danh sách gồm n học sinh (n- nhập từ bàn phím)

- Hiển thị ra màn hình tất cả những học sinh sinh năm 1985.

- Cho biết có bao nhiêu học sinh sinh năm 1985 và có quê ở Thái Nguyên.

Bài 14. Xây dựng một interface có tên là HCNInterface chứa phương thức sau:

dientichHCN(); getChieuDai() và getChieuRong(); setDaiRong(cd, cr) Sử dụng HCNInterface trên để xây dựng lớp Hinhchunhat chứa hai thuộc tính là: chieudai, chieurong và triển khai các phương thức trong HCNInterface trên?

Xây dựng lớp HCNTest thừa kế lớp Hinhchunhat chứa phương thức main thực hiện các công việc sau:

a. Khai báo một mảng chứa n hình chữ nhật (n>0, nhập từ bàn phím). Sau đó nhập chiều dài và chiều rộng cho n hình chữ nhật đó.
b. In ra màn hình thông tin: chiều dài, chiều rộng và diện tích của n hình chữ nhật.
c. In ra màn hình thông tin về hình chữ nhật có diện tích lớn nhất.

CHƯƠNG 4: GIAO DIỆN ĐỒ HỌA (6,6,24)

Lý thuyết

1. Compoment

Tất cả các thành phần cấu tạo nên chương trình GUI được gọi là component.

Ví dụ

- Containers,
- textfields, labels, checkboxes, textareas
- scrollbars, scrollpanes, dialog

2. Container

Container là đối tượng vật chứa hay những đối tượng có khả năng quản lý và nhóm các đối tượng khác lại. Một số đối tượng container trong java

- **Panel**: Đối tượng khung chứa đơn giản nhất, dùng để nhóm các đối tượng, thành phần con lại. Một Panel có thể chứa bên trong một Panel khác.
- **Frame**: khung chứa Frame là một cửa số window bao gồm một tiêu đều và một đường biên (border) như các ứng dụng windows thông thường khác. Khung chứa Frame thường được sử dụng để tạo ra cửa sổ chính của các ứng dụng.
- **Dialogs**: đây là một cửa sổ dạng hộp hội thoại (cửa sổ dạng này còn được gọi là pop-up window), thường được dùng để đưa ra thông báo, hay dùng để lấy dữ liệu nhập từ ngoài vào thông qua các đối tượng, thành phần trên dialog như TextField chẳng hạn.
- **ScrollPanes**: là một khung chứa tương tự khung chứa Panel, nhưng có thêm 2 thanh trượt giúp ta tổ chức và xem được các đối tượng lớn choán nhiều chỗ trên màn hình như những hình ảnh hay văn bản nhiều dòng.
- **Label**: Được dùng để hiển thị chuỗi (String).
- **TextField :** Là điều khiển text cho phép hiển thị text hoặc cho user nhập dữ liệu vào.
- **TextArea:** Được dùng khi text có nội dung từ hai dòng trở lên.
- **Button:** Các nút Push hay Command là cách dễ nhất để thực hiện các sư kiện.
- **Checkboxes and RadioButtons:** Checkboxes được dùng khi cho phép user nhiều cọn chọn lựa. Radiobuttons được dùng để user chỉ ra một lựa chọn duy nhất.

3. Layout Manager

Khung chứa container nhận các đối tượng từ bên ngoài đưa vào và nó phải biết làm thế nào để tổ chức sắp xếp "chỗ ở" cho các đối tượng đó. Các bộ quản lý trình bày mà thư viện AWT cung cấp cho ta bao gồm:

- **FlowLayout**: Sắp xếp các đối tượng từ trái qua phải và từ trên xuống dưới. Các đối tượng đều giữ nguyên kích thước của mình.
- **BorderLayout**: Các đối tượng được đặt theo các đường viền của khung chứa theo các cạnh West, East, South, North và Center tức Đông, Tây, Nam, Bắc và Trung tâm hay Trái, Phải, Trên, Dưới và Giữa tùy theo cách nhìn của chúng ta.
- **GridLayout**: Tạo một khung lưới vô hình với các ô bằng nhau. Các đối tượng sẽ đặt vừa kích thước với từng ô đó. Thứ tự sắp xếp cũng từ trái qua phải và từ trên xuống dưới.
- **GridBagLayout**: Tương tự như GridLayout, các đối tượng khung chứa cũng được đưa vào một lưới vô hình. Tuy nhiên kích thước các đối tượng không nhất thiết phải vừa với 1 ô mà có thể là 2, 3 ô hay nhiều hơn tùy theo các ràng buộc mà ta chỉ định thông qua đối tượng GridBagConstraint.
- **NullLayout**: Cách trình bày tự do. Đối với cách trình bày này người lập trình phải tự động làm tất cả từ việc định kích thước của các đối tượng, cũng như xác định vị trí của nó trên màn hình. Ta không phụ thuộc vào những ràng buộc đông, tây , nam, bắc gì cả.

4. Xử lý biến cố/sự kiện

Những sự kiện được phát sinh khi người dùng tương tác với giao diện chương trình (GUI). Những tương tác thường gặp như: di chuyển, nhấn chuột, nhấn một nút nhấn, chọn một MenuItem trong hệ thống thực đơn, nhập dữ liệu trong một ô văn bản, đóng cửa sổ ứng dụng, …

- Xử lý sự kiện chuột

Java cung cấp hai intefaces lắng nghe (bộ lắng nghe sự kiện chuột) là MouseListener và MouseMotionListener để quản lý và xử lý các sự kiện liên quan đến thiết bị chuột.

Các phương thức của interface **MouseListener**:

- *public void mousePressed(MouseEvent event)*: được gọi khi một nút chuột được nhấn và con trỏ chuột ở trên component.
- *public void mouseClicked(MouseEvent event)*: được gọi khi một nút chuột được nhấn và nhả trên component mà không di chuyển chuột.
- *public void mouseReleased(MouseEvent event)*: được gọi khi một nút chuột nhả sa khi kéo rê.
- *public void mouseEntered(MouseEvent event)*: được gọi khi con trỏ chuột vào trong đường biên của một component.
- *public void mouseExited(MouseEvent event)*: được gọi khi con trỏ chuột ra khỏi đường biên của một component.

Các phương thức của interface **MouseMotionListener**:

- *public void mouseDragged(MouseEvent even)*: phương thức này được gọi khi người dùng nhấn một nút chuột và kéo trên một component.
- *public void mouseMoved(MouseEvent event)*: phương thức này được gọi khi di chuyển chuột trên component.

- Xử lý sự kiện bàn phím

Để xử lý sự kiện bàn phím java hỗ trợ một bộ lắng nghe sự kiện đó là interface KeyListener.

Các phương thức của interface KeyListener

- Phương thức *keyPressed* được gọi khi một phím bất kỳ được nhấn.
- Phương thức *keyTyped* được gọi thực hiện khi người dùng nhấn một phím không phải "phím hành động" (như phím mũi tên, phím Home, End, Page Up, Page Down, các phím chức năng như: Num Lock, Print Screen, Scroll Lock, Caps Lock, Pause).
- Phương thức *keyReleased* được gọi thực hiện khi nhả phím nhấn sau khi sự kiện keyPressed hoặc keyTyped.

Thực hành

Bài 1. Viết chương trình xây dựng giao diện "máy tính cá nhân bao gồm: cộng, trừ, nhân, chia" tương tự chương trình Calculator trên windows.

Bài 2. Viết chương trình xây dựng giao diện giải phương trình bậc một.

Bài 3. Viết chương trình xây dựng giao diện giải phương trình bậc hai

Bài 4. Cải tiến lại bài tập 2 với giao diện cho phép lựa chọn giải phương trình bậc một, bậc hai.

Bài 5. Viết chương trình xây dựng giao diện "Lựa chọn tính chu vi, diện tích hình tròn và hình chữ nhật"

Bài 6. Viết chương trình xây dựng giao diện "chương trình nhập, xuất, sắp xếp tăng dần trong mảng 1 chiều"

Bài 7. Viết chương trình xây dựng giao diện "Nhập 2 số, Tìm ước chung lớn nhất, bội chung nhỏ nhất của 2 số"

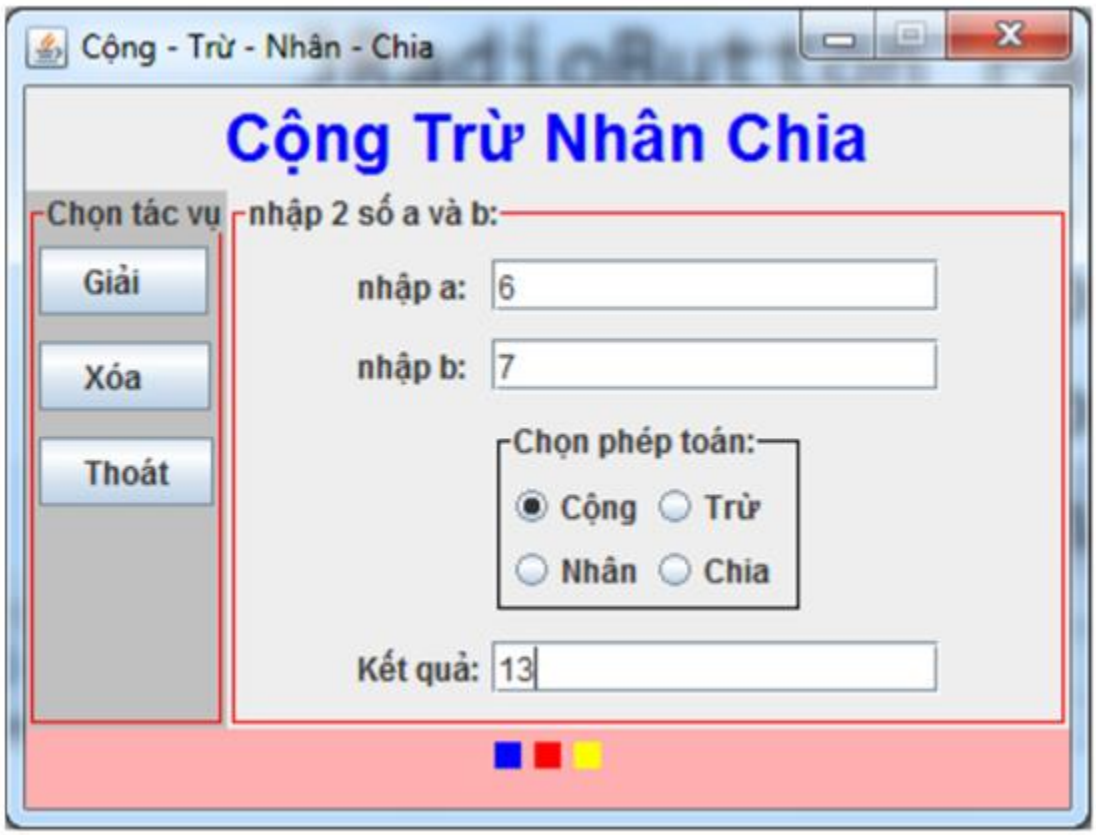
Bài 8. Viết chương trình xây dựng giao diện "nhập và cộng, trừ 2 phân số"

Bài 9. Xây dựng ứng dụng quản lý thông tin nhân viên theo mẫu:

a) Khi người dùng chọn nút "Đăng ký" tất cả thông tin đã nhập và chọn sẽ được hiển thị ra màn hình. Nếu người dùng chưa nhập hoặc chưa chọn, ứng dụng sẽ hiển thị thông báo yêu cầu người dùng hoàn nhập và chọn thông tin nào còn thiếu.

b) Khi người dùng chọn nút "Bỏ qua", thiết lập nội dung hiển thị của các thành phần trên màn hình giống như lúc ban đầu.

Bài 10. Thiết kế giao diện để thực hiện các phép toán : '+' '-' '*' ':'.



a) Thiết kế giao diện như hình.

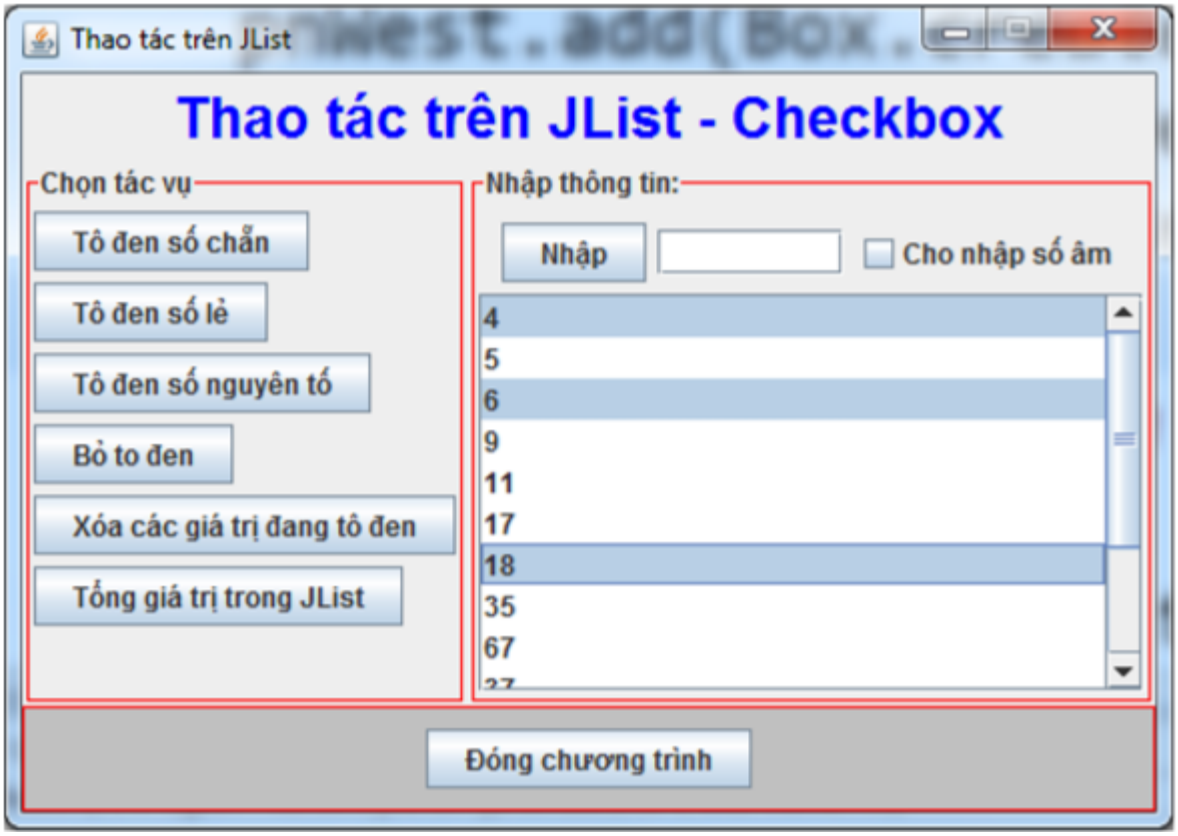
b) Các giá trị a và b bắt buộc nhập giá trị số. Nếu người dùng chưa nhập hoặc nhập không đúng, ứng dụng sẽ hiển thị thông báo nhắc.

c) Khi bấm nút Giải thì tùy thuộc vào phép toán được chọn mà thực hiện xử lý và thông báo lỗi.

d) Khi người dùng nhấn nút xóa thì thiết lập nội dung hiển thị của các thành phần trên màn hình giống như lúc ban đầu.

e) Khi người dùng nhấn nút Thoát thì chương trình sẽ hiển thị thông báo hỏi người dùng có muốn thoát hay không, nếu đồng ý thì cửa sổ chương trình sẽ đóng.

Bài 11. Xây dựng ứng dụng thao tác trên Jlist, JtextField, Jcheckbox



a) Chương trình cho phép nhập vào các số nguyên trong phần nhập thông tin, Khi người nhập giá trị và click nút "Nhập" thì sẽ cập nhập dữ liệu xuống JList, Nếu checked vào "Cho nhập số âm" thì các số âm mới được đưa vào JList còn không thì thông báo lỗi.

b) Thực hiện các yêu cầu tại khung "Chọn tác vụ"

c) Nút Đóng chương trình: Hiển thông báo hỏi người sử dụng có muốn đóng giao diện hay không. Nếu có thì tắt toàn bộ cửa sổ của chương trình.

CHƯƠNG 5: LUỒNG VÀ TẬP TIN (6,6,24)

Lý thuyết

1. Luồng

- Tất cả những hoạt động nhập/xuất dữ liệu (nhập dữ liệu từ bàn phím, lấy dữ liệu từ mạng về, ghi dữ liệu ra đĩa, xuất dữ liệu ra màn hình, máy in, …) đều được quy về một khái niệm gọi là luồng (stream). Luồng là nơi có thể "sản xuất" và "tiêu thụ" thông tin.
- Java định nghĩa hai kiểu luồng: byte và ký tự. Luồng byte (hay luồng dựa trên byte) hỗ trợ việc xuất nhập dữ liệu trên byte, thường được dùng khi đọc ghi dữ liệu nhị phân. Luồng ký tự được thiết kế hỗ trợ việc xuất nhập dữ liệu kiểu ký tự (Unicode).
- Những luồng được định nghĩa trước:
- Tất cả các chương trình viết bằng java luôn tự động import gói java.lang. Gói này có định nghĩa lớp System, bao gồm một số đặc điểm của môi trường run-time, nó có ba biến luồng được định nghĩa trước là in, out và err, các biến này là các fields được khai báo static trong lớp System.
    - ***System.out:*** luồng xuất chuẩn, mặc định là console. System.out là một đối tượng kiểu PrintStream.
    - ***System.in***: luồng nhập chuẩn, mặc định là bàn phím. System.in là một đối tượng kiểu InputStream.
    - ***System. err:*** luồng lỗi chuẩn, mặc định cũng là console. System.out cũng là một đối tượng kiểu PrintStream giống System.out.

2. Sử dụng luồng Byte

Phương thức	Ý nghĩa
InputStream	
int available( )	Trả về số lượng bytes có thể đọc được từ luồng nhập
void close( )	Đóng luồng nhập và giải phóng tài nguyên hệ thống gắn với luồng. Không thành công sẽ ném ra một lỗi IOException
void mark(int numBytes)	Đánh dấu ở vị trí hiện tại trong luồng nhập
boolean markSupported( )	Kiểm tra xem luồng nhập có hỗ trợ phương thức *mark()* và *reset()* không.
int read( )	Đọc byte tiếp theo từ luồng nhập

int read(byte buffer[ ])	Đọc buffer. length bytes và lưu vào trong vùng nhớ buffer. Kết quả trả về số bytes thật sự đọc được
int read(byte buffer[ ], int offset, int numBytes)	Đọc numBytes bytes bắt đầu từ địa chỉ offset và lưu vào trong vùng nhớ buffer. Kết quả trả về số bytes thật sự đọc được
void reset( )	Nhảy con trỏ đến vị trí được xác định bởi việc gọi hàm *mark()* lần sau cùng.
long skip(long numBytes)	Nhảy qua numBytes dữ liệu từ luồng nhập
OutputStream	
void close( )	Đóng luồng xuất và giải phóng tài nguyên hệ thống gắn với luồng. Không thành công sẽ ném ra một lỗi IOException
void flush( )	Ép dữ liệu từ bộ đệm phải ghi ngay xuống luồng (nếu có)
void write(int b)	Ghi byte dữ liệu chỉ định xuống luồng
void write(byte buffer[ ])	Ghi buffer. length bytes dữ liệu từ mảng chỉ định xuống luồng
void write(byte buffer[ ], int offset, int numBytes)	Ghi numBytes bytes dữ liệu từ vị trí offset của mảng chỉ định buffer xuống luồng

- Đọc dữ liệu từ console: chương trình minh họa việc đọc một mảng bytes từ System.in

```
Import java.io.*;
class ReadBytes {
    public static void main(String args[]) throws IOException {
            byte data[] = new byte[100];
            System.out.print("Enter some characters. ");
            System.in. read(data); System.out.print("You entered: ");
            for(int i=0; i < data. length; i++) System.out.print((char) data[i]);
    }
}
```

- Xuất dữ liệu từ console: minh họa sử dụng phương thức System.out. write() để xuất ký tự ‘X’ ra Console

```
import java.io.*;
class WriteDemo {
    public static void main(String args[]) {
            int b;
            b = 'X';
            System.out. write(b);
            System.out. write('\n');
    }
}
```

- Đọc dữ liệu từ file
  - Mở một file để đọc dữ liệu
    FileInputStream(String fileName) throws FileNotFoundException
    Nếu file không tồn tại: thì ném ra FileNotFoundException
  - Đọc dữ liệu: dùng phương thức read() int read( ) throws IOException: đọc từng byte từ file và trả về giá trị của byte đọc được. Trả về -1 khi hết file, và ném ra IOException khi có lỗi đọc.
  - Đóng file: dùng phương thức close() void close( ) throws IOException: sau khi làm việc xong cần đóng file để giải phóng tài nguyên hệ thống đã cấp phát cho file.

Ví dụ: Hiển thị nội dung của một file tên test.txt lưu tại D:\test.txt

```
import java.io.*;
class ShowFile {
    public static void main(String args[]) throws IOException {
            int i;
            FileInputStream fin;
            try {fin = new FileInputStream(“D:\\test.txt”);   }
            catch(FileNotFoundException exc) {
                    System.out.println("File Not Found");
                    return;
```

```
        }
        catch(ArrayIndexOutOfBoundsException exc) {
            System.out.println("Usage: ShowFile File");
            return;
        }
        do {
            i = fin. read();
            if(i != -1) System.out.print((char) i);
        } while(i != -1); fin.close();
    }
}
```

- Ghi dữ liệu xuống file
  - Mở một file để ghi dữ liệu

FileOutputStream(String fileName) throws

FileNotFoundException

Nếu file không tạo được: thì ném ra

FileNotFoundException

- Ghi dữ liệu xuống: dùng phương thức write() void write(int byteval) throws IOException: ghi một byte xác định bởi tham số byteval xuống file, và ném ra IOException khi có lỗi ghi.
- Đóng file: dùng phương thức close(), void close( ) throws IOException: sau khi làm việc xong cần đóng file để giải phóng tài nguyên hệ thống đã cấp phát cho file.

Ví dụ: copy nội dung một file text đến một file text khác.

```
import java.io.*;
class CopyFile {
    public static void main(String args[])      throws IOException {
        int i;
        FileInputStream fin;
        FileOutputStream fout;
        try {
            try{ fin = new FileInputStream("D:\\source.txt"); }
```

```
                catch(FileNotFoundException exc){
                        System.out.println("Input File Not Found");
                        return;
                }
                try { fout = new FileOutputStream("D:\\dest.txt"); }
                catch(FileNotFoundException exc) {
                        System.out.println("Error Opening Output File");
                        return;
                }
        }
        catch(ArrayIndexOutOfBoundsException exc) {
                System.out.println("Usage: CopyFile From To");
                return;
        }

        try {
                do {
                        i = fin. read();
                        if(i != -1) fout. write(i);
                } while(i != -1);
        }
        catch(IOException exc) { System.out.println("File Error"); }
    fin.close();    fout.close();
    }
}
```

3. *Sử dụng luồng ký tự*

Những phương thức định nghĩa trong lớp trừu tượng *Reader* và *Writer*.

Phương thức	Ý nghĩa
Reader	
abstract void close( )	Đóng luồng
void mark(int numChars)	Đánh dấu vị trí hiện tại trên luồng
boolean markSupported( )	Kiểm tra xem luồng có hỗ trợ thao tác đánh dấu *mark()* không?
int read( )	Đọc một ký tự

int read(char buffer[ ])	Đọc buffer. length ký tự cho vào buffer
abstract int read(char buffer[ ], int offset, int numChars)	Đọc numChars ký tự cho vào vùng đệm buffer tại vị trí buffer[offset]
boolean ready( )	Kiểm tra xem luồng có đọc được không?
void reset( )	Dời con trỏ nhập đến vị trí đánh dấu trước đó
long skip(long numChars)	Bỏ qua numChars của luồng nhập
Writer	
abstract void close( )	Đóng luồng xuất. Có lỗi ném ra IOException
abstract void flush( )	Dọn dẹp luồng (buffer xuất)
void write(int ch)	Ghi một ký tự
void write(byte buffer[ ])	Ghi một mảng các ký tự
abstract void write(char buffer[ ], int offset, int numChars)	Ghi một phần của mảng ký tự
void write(String str)	Ghi một chuỗi
void write(String str, int offset, int numChars)	Ghi một phần của một chuỗi ký tự

4. *Lớp File*

Lớp File không phục vụ cho việc nhập/xuất dữ liệu trên luồng. Lớp File thường được dùng để biết được các thông tin chi tiết về tập tin cũng như thư mục (tên, ngày giờ tạo, kích thước, …)

Một số phương thức thường gặp của lớp File (chi tiết về các phương thức đọc thêm trong tài liệu J2SE API Specification)

public String ***getName()***	Lấy tên của đối tượng File
public String ***getPath()***	Lấy đường dẫn của tập tin
Public boolean ***isDirectory()***	Kiểm tra xem tập tin có phải là thư mục không?

public boolean ***isFile()***	Kiểm tra xem tập tn có phải là một file không?
...	
public String[] ***list()***	Lấy danh sách tên các tập tin và thư mục con của đối tượng File đang xét và trả về trong một mảng.

Thực hành

Bài 1. Tạo file bt1.txt trong ổ D, gõ nội dung vào file và lưu lại. Viết chương trình hiển thị nội dung file bt1.txt.

Bài 2. Viết chương trình tạo file bt2.txt, sau đó ghi vào các ký tự từ a đến z.

Bài 3. Dùng DataOutputStream và DataInputStream để ghi dữ liệu kiểu int, double, boolean, double, char, string vào file bt3.txt. Đọc và hiển thị dữ liệu trong file bt3.txt 4. Viết chương trình tạo đối tượng SinhVien gồm các thuộc tính: họ tên, năm sinh, điểm trung bình. Ghi các thuộc tính này vào file theo kiểu nhị phân. Đọc và in thông tin của file này ra màn hình.

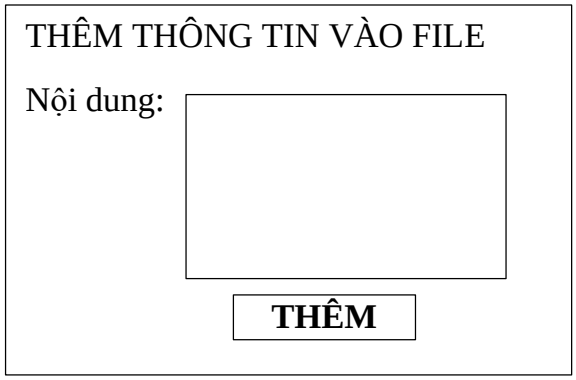
Bài 4. Viết chương trình nhập danh sách tên của 1 lớp học và ghi chúng xuống file tên là danhsach.txt. Việc đọc và ghi kết thúc khi người dùng nhập vào "stop".

Bài 5. Viết chương trình đọc và hiển thị nội dung file danhsach.txt ở bài tập 5.

Bài 6. Thiết kế giao diện như sau:

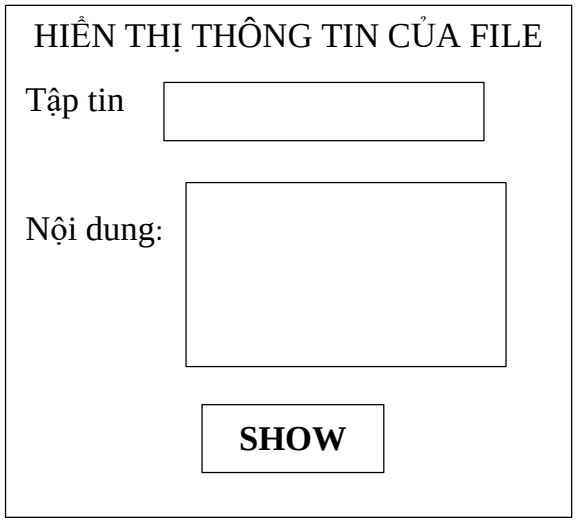
Khi click vào nút lệnh LƯU thì toàn bộ nội dung trong TEXT sẽ được lưu vào tập tin d:/bai7.txt.

Bài 7. Thiết kế giao diện như sau:



Yêu cầu: khi click vào nút THÊM thì nội dung của TEXT sẽ được thêm vào cuối tập tin d:/bai8.txt

Bài 8. Thiết kế giao diện như sau:



Yêu cầu:

- Gõ đường dẫn đến tập tin đã có trên hệ thống.
- Khi click vào nút SHOW thì nội dung tập tin (vừa gõ trên ô Text) sẽ được hiển thị tại vị trí ô text nội dung.