SLIDE BÀI GIẢNG
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG JAVA

THS. NGUYỄN ĐÌNH THÀ

0905 550 551

Đà Nẵng – 2/18/2025

1

CHƯƠNG 1. ÔN TẬP NNLT JAVA

2

MỤC TIÊU

- Kết thúc chương này, sinh viên có khả năng:
  - Phân biệt được các thành phần cơ bản của NNLT JAVA
  - Đặt tên biến, hằng đúng theo yêu cầu
  - Phân biệt được các loại từ khoá
  - Sử dụng được các cấu trúc lệnh:
    - Lệnh rẽ nhánh
    - Lệnh lặp
    - Lệnh break và continue
  - Sử dụng được hàm
  - Sử dụng được mảng

3

Cấu trúc chương trình Java

```
package com.uda;
public class HelloWorld{
    public static void main(String[] args){
        // mã thực thi
    }
}
```

Lưu thành → HelloWorld.java

com.uda: tên gói chứa lớp

Sử dụng ký tự thường và dấu chấm. Có thể xem package như folder còn class như file.

HelloWorld: tên lớp

Phải giống tên file java. Viết hoa ký tự đầu của mỗi từ

main(): *phương thức bắt đầu chạy*

Lớp có thể có nhiều phương thức nhưng main() được gọi tự động khi ứng dụng chạy

4

Kiểu dữ liệu nguyên thủy

Kiểu	Mặc định	Bit	Khả năng lưu trữ	
			Giá trị nhỏ nhất	Giá trị lớn nhất
byte	0	8	-2^7	$+2^7-1$
short	0	16	-2^{15}	$+2^{15}-1$
int	0	32	-2^{31}	$+2^{31}-1$
long	0L	64	-2^{63}	$+2^{63}-1$
float	0.0F	32	$-3.40292347 \times 10^{38}$	$+3.40292347 \times 10^{38}$
double	0.0	64	$-1.7976931348623157 \times 10^{308}$	$+1.7976931348623157 \times 10^{308}$
char	'\u0000'	16	'\u0000'	'\uFFFF'
boolean	false	1	false	true

Giá trị mặc định là giá trị sẽ được gán cho biến khi khai báo không khởi đầu giá trị cho biến

5

Khai báo biến

Cú pháp:

<kiểu dữ liệu> <tên biến> [=giá trị khởi đầu];

Ví dụ:

```
int a; // khai báo biến không khởi đầu giá trị
double b = 5; // khai báo biến có khởi đầu giá trị
```

Khai báo nhiều biến cùng kiểu:

```
int a, b = 5, c;
```

Gán giá trị cho biến:

```
c = 9;
a = 15;
```

6

Đặt tên biến

Sử dụng ký tự alphabet, số, $ hoặc gạch dưới (_).
Không ~~bắt đầu bởi số~~, không sử dụng ~~từ khóa~~

Tên có phân biệt HOA/thường. Đặt tên biến theo kiểu camelCase

** Từ khóa là các từ được sử dụng để xây dựng ra ngôn ngữ lập trình java*

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

7

DEMO

Khai báo 2 biến số nguyên a, b và c và gán các giá trị cho a, b

Thực hiện phép cộng a và b được c

Xuất kết quả c. *Gợi ý lệnh in: System.out.println(c);*

8

Các hàm xuất ra màn hình

System.out.print(): Xuất xong không xuống dòng

System.out.println(): Xuất xong có xuống dòng

System.out.printf(): Xuất có định dạng, các ký tự định dạng

%d: số nguyên

%f: số thực (mặc định là 6 số lẻ), %.3f định dạng 3 số lẻ

%s: chuỗi

Ví dụ:

```
System.out.print("UDA");
System.out.println("CNTT");
System.out.printf("Năm %d", 2025);
```

9

DEMO

Khai báo 2 biến hoTen và tuoi

Sử dụng cả 3 hàm trên để xuất dòng sau

<<hoTen>> năm nay <<tuoi>> tuoi

10

Nhập từ bàn phím

java.util.Scanner cho phép nhận dữ liệu từ bàn phím một cách đơn giản

Tạo đối tượng Scanner:

```
Scanner sc = new Scanner(System.in);
```

Các phương thức thường dùng:

```
sc.nextLine();  // Nhận 1 dòng nhập từ bàn phím
sc.nextInt();  // Nhận 1 số nguyên nhập từ bàn phím
sc.nextDouble();  // Nhận 1 số thực nhập từ bàn phím
```

11

DEMO

Khai báo 2 biến hoTen và tuoi

Nhập họ tên và tuổi từ bàn phím

Xuất ra dòng theo định dạng sau:

<<hoTen>> năm nay <<tuoi>> tuoi

12

DEMO

Khai báo 2 biến hoTen, tuoi và queQuan

Nhập họ tên, tuổi và quê quán từ bàn phím

Xuất ra dòng theo định dạng sau:

<<hoTen>> que o <<queQuan>> nam nay <<tuoi>> tuoi

Chống trôi lệnh nhập dữ liệu

```
Integer.parseInt(sc.nextLine());  // nhập chuỗi và chuyển sang số nguyên
Double.parseDouble(sc.nextLine());  // nhập chuỗi, chuyển sang số double
Float.parseFloat(sc.nextLine());  // nhập chuỗi, chuyển sang số float
```

13

Sử dụng try…catch để kiểm lỗi

Xét trường hợp: `int a = sc.nextInt();`
Hoặc: `int a = Integer.parseInt(s);`

Điều gì sẽ xảy ra khi người dùng nhập không phải số hoặc chuỗi s không phải là chuỗi chứa số.

Hãy sử dụng lệnh try…catch để kiểm soát các lỗi trên

```
try {
    int a = sc.nextInt();
    System.out.println("Bạn đã nhập đúng");
}
catch (Exception ex){
    System.out.println("Vui lòng nhập số !");
}
```

14

Các hàm toán học

Java cung cấp các hàm tiện ích giúp chúng ta thực hiện các phép tính khó một cách dễ dàng như: Làm tròn số, Tính căn bậc 2, Tính lũy thừa, …

Ví dụ sau đây tính căn bậc 2 của 7:

```
double a = Math.sqrt(7);
```

Ngoài Math.sqrt() còn rất nhiều hàm khác được trình bày ở slide sau.

15

Các hàm toán học

Hàm	Diễn giải	Ví dụ
Math.min(a, b)	Lấy số nhỏ nhất của 2 số a và b	x = Math.min(5, 3.5) => x=3.5
Math.max(a, b)	Lấy số lớn nhất của 2 số a và b	x = Math.max(5, 3.5) => x=5
Math.pow(a, n)	Tính a^n (a lũy thừa n)	x = Math.pow(5, 3) => x=75
Math.sqrt(a)	Tính $\sqrt{a}$ (căn bậc 2 của a)	x = Math.sqrt(16) => x=4
Math.abs(a)	Lấy giá trị tuyệt đối của a	x = Math.abs(-5) => x=5
Math.ceil(a)	Lấy số nguyên trên của a	x = Math.ceil(3.5) => x=4
Math.floor(a)	Lấy số nguyên dưới của a	x = Math.floor(3.5) => x=3
Math.round(a)	Làm tròn số của a	x = Math.round(3.5) => x=4
Math.random()	Sinh số ngẫu nhiên từ 0 đến 1	x = Math.random() => x=0..1

16

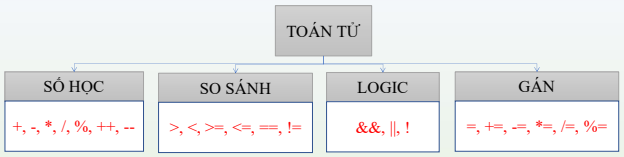
DEMO

1. Sinh số ngẫu nhiên từ 5 đến 12. Xuất số đó và căn bậc 2 của nó ra màn hình

2. Nhập 2 số thực a và b từ bàn phím. Tính và xuất a lũy b, giá trị nhỏ nhất của 2 số

17

Toán tử & biểu thức



Biểu thức là sự kết hợp giữa toán tử và toán hạng. Kết quả của biểu thức là một giá trị.

Giá trị của các biểu thức sau?

```
int x = 11 % 4;
boolean a = 9 < 2 && true || 4 > 3;
```

18

Toán tử điều kiện

Toán tử điều kiện là toán tử 3 ngôi duy nhất trong ngôn ngữ Java
Cú pháp:

<điều kiện> ? <giá trị đúng> : <giá trị sai>

Diễn giải:
Nếu biểu thức **<điều kiện>** có giá trị là true thì kết quả của biểu thức là **<giá trị đúng>**, ngược lại là **<giá trị sai>**

Ví dụ: tìm số lớn nhất của 2 số a và b

```
int a = 1, b = 9;
int max = a > b ? a : b;
```

Tìm số lớn nhất trong 3 số a, b và c?

Lệnh If

Cú pháp

```
if (<<điều kiện>>) {
    << Công việc >>
}
```

Diễn giải:
Nếu **điều kiện** có giá trị true thì **công việc** được thực hiện

Ví dụ:

```
double diem = 4;
if (diem >= 5) {
    System.out.println("Đậu");
}
```

DEMO

Nhập số từ bàn phím.

Nếu số dương thì tính và xuất căn bậc 2 của số đó ra màn hình

Lệnh if…else

Cú pháp

```
if (<<điều kiện>>) {
    << công việc 1 >>
} else {
    << công việc 2 >>
}
```

Ví dụ:

```
double diem = 4;
if (diem < 5) {
    System.out.println("Rớt");
}
else {
    System.out.println("Đậu");
}
```

Diễn giải:
Nếu **điều kiện** có giá trị true thì **công việc 1** được thực hiện, ngược lại **công việc 2** được thực hiện

DEMO

Nhập số từ bàn phím.

Nếu số dương thì tính và xuất căn bậc 2 của số đó ra màn hình, ngược lại thì thông báo lỗi.

Nhiều lệnh if

Cú pháp

```
if (<<điều kiện 1>>){
    << công việc 1 >>
}
else if (<<điều kiện 2>>){
    << công việc 2 >>
}
…
else {
    << công việc N+1 >>
}
```

Diễn giải:
Chương trình sẽ kiểm tra từ **điều kiện 1 đến N** nếu gặp **điều kiện i** đầu tiên có giá trị true thì sẽ thực hiện **công việc i**, ngược lại sẽ thực hiện **công việc N+1**

Ví dụ

```
double delta = Math.pow(b, 2) – 4 * a * c;
if (delta < 0) {
    System.out.println("Vô nghiệm");
}
else if (delta == 0) {
    System.out.println("Nghiệm kép");
}
else {
    System.out.println("2 nghiệm");
}
```

DEMO

Viết chương trình tính thuế thu nhập. Giả sử thu nhập gồm lương và thưởng

Thuế thu nhập được tính như sau:

Dưới 9 triệu: không đóng thuế

Từ 9 đến 15 triệu: thuế 10%

Từ 15 đến 30 triệu: 15%

Trên 30 triệu: 20%

25

Lệnh switch

Cú pháp

```
switch (<<biểu thức>>) {
    case <<giá trị 1>>:
        // Công việc 1
        break;
    case <<giá trị 2>>:
        // Công việc 2
        break;
    …
    default:
        // Công việc N+1
}
```

Diễn giải

- So sánh giá trị của biểu thức switch với giá trị của các case. Nếu bằng với giá trị của case nào thì sẽ thực hiện công việc của case đó, ngược lại sẽ thực hiện công việc của default.
- Nếu công việc của case không chứa lệnh break thì case tiếp sau sẽ được thực hiện
- default là tùy chọn

26

Ví dụ lệnh switch

```
double a = 5, b = 7, c = -1;
char op = '+';
switch(op) {
    case '+':
        c = a + b;
        break;
    case '-':
        c = a - b;
        break;
    case 'x':
    case ':':
        System.out.println("Đang xây dựng");
        break;
    default:
        System.out.println("Vui lòng chọn +, -, x và :");
}
```

Không có break

27

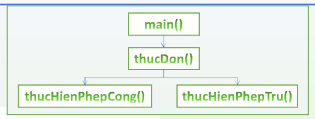
DEMO

Nhập tháng và năm từ bàn phím.

Xuất số ngày của tháng đã nhập

28

Tổ chức chương trình




```
import java.util.Scanner;

public class ChuongTrinh {
    public static void main(String[] args) {
        thucDon();
    }

    public static void thucDon() {

    public static void thucHienPhepCong() {

    public static void thucHienPhepTru() {
}
```

Hiển thị thực đơn chính của chương trình

29

Thiết kế thực đơn

```
System.out.println(">> MÁY TÍNH CÁ NHÂN <<");
System.out.println("+--------------------+");
System.out.println("| 1. Cộng            |");
System.out.println("| 2. Trừ             |");
System.out.println("| 3. Kết thúc        |");
System.out.println("+--------------------+");
System.out.println(" >> Chọn chức năng? ");

Scanner scanner = new Scanner(System.in);
int answer = scanner.nextInt();
if(answer == 1){
    thucHienPhepCong();
}
else if(answer == 2){
    thucHienPhepTru();
}
else if(answer == 3){
    System.exit(0);
}
```

Gọi phương thức thực hiện phép cộng

Gọi phương thức thực hiện phép trừ

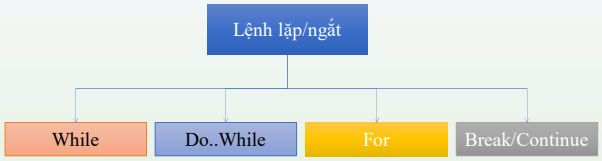
Thoát ứng dụng

30

DEMO

Tổ chức chương trình trên bằng cách đổi if…else sang switch…case

Lệnh lặp & ngắt



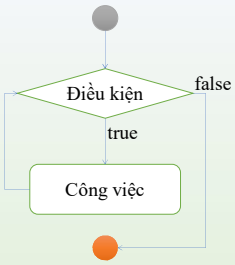
Lệnh lặp while

Cú pháp

```
while (<<điều kiện>>) {
    // công việc
}
```

Diễn giải:

Thực hiện công việc trong khi biểu thức điều kiện có giá trị là true.



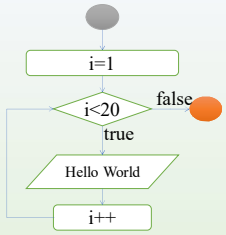
Lệnh lặp while

Ví dụ

```
int i = 1;
while (i < 20) {
    System.out.println("Hello World !");
    i++;
}
```

Diễn giải:

Đoạn mã trên xuất 19 dòng Hello World ra màn hình



DEMO

1. Xuất bảng cửu chương 7

2. Tính trung bình cộng các số chia hết cho 3 từ 25 đến 250.

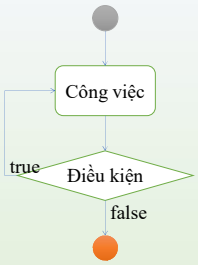
Lệnh lặp do…while

Cú pháp:

```
do {
    // công việc
}
while (<<điều kiện>>);
```

Diễn giải:

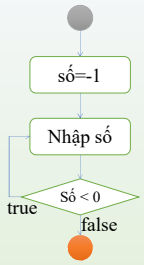
Tương tự lệnh lặp while chỉ khác ở chỗ điều kiện được kiểm tra sau, nghĩa là công việc được thực hiện ít nhất 1 lần.



Lệnh lặp do…while

Ví dụ

```
double so = -1;
do {
    so = sc.nextDouble();
}
while (so < 0);
```



Diễn giải:

Đoạn mã trên chỉ cho phép nhập số nguyên dương từ bàn phím.

37

DEMO

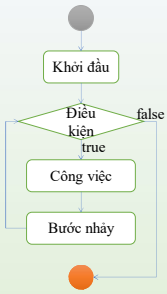
Nhập điểm từ 0 đến 10

38

Lệnh lặp for

Cú pháp

```
for (khởi đầu ; điều kiện; bước nhảy){
    // công việc
}
```



Diễn giải

B1: Thực hiện <<khởi đầu>>

B2: Kiểm tra <<điều kiện>>

True: B3

False: kết thúc

B3: Thực hiện << công việc >>

B4: Thực hiện <<bước nhảy>>

B5: Trở lại B2

39

DEMO

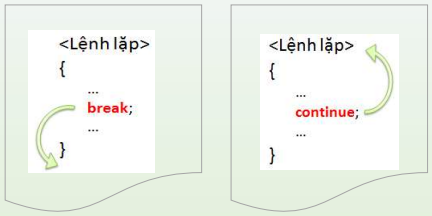
Bảng cửu chương với lệnh lặp for

40

Lệnh break & continue

break dùng để ngắt lệnh lặp

continue dùng để thực hiện lần lặp tiếp theo ngay lặp tức



41

Ví dụ break

Ví dụ:

```
int diem = 0;
while(true){
    diem = scanner.nextInt();
    if(diem >= 0 && diem <=10){
        break;
    }
    System.out.println("Điểm phải từ 0 đến 10");
}
```

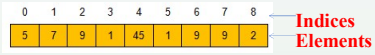
Diễn giải:

Nhập điểm hợp lệ (từ 0 đến 10)

42

Mảng là gì

Mảng là cấu trúc lưu trữ nhiều phần tử có cùng kiểu dữ liệu



Để truy xuất các phần tử cần biết chỉ số (index). Chỉ số được đánh từ 0.

Các thao tác mảng:

- Khai báo
- Truy xuất (đọc/ghi) phần tử
- Lấy số phần tử
- Duyệt mảng
- Sắp xếp các phần tử mảng

Khai báo mảng

- Khai báo không khởi tạo

```
int[] a; // khai báo mảng số nguyên a nhưng chưa biết số phần tử
a = new int[100]; // khởi tạo mảng a có 100 phần tử
int b[];// mảng số nguyên chưa biết số phần tử
String[] c = new String[5]; // mảng chứa 5 chuỗi
```

- Khai báo có khởi tạo

```
double[] d1 = new double[]{2, 3, 4, 5, 6}; // mảng số thực, 5 phần tử, đã được khởi tạo
double[] d2 = {2, 3, 4, 5, 6}; // mảng số thực, 5 phần tử, đã được khởi tạo
```

Truy xuất các phần tử

- Sử dụng chỉ số (**index**) để phân biệt các phần tử. Chỉ số mảng tính từ 0.

```
int a[] = {4, 3, 5, 7};
a[2] = a[1] * 4;  // 3*4=12
```

Sau phép gán này mảng là {4, 3, 12, 7};

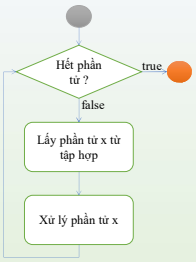
- Sử dụng thuộc tính **length** để lấy số phần tử của mảng

a.length có giá trị là 4

For each

Cú pháp:

```
for (<<kiểu>> x : <<tập hợp>>){
    // Xử lý phần tử x
}
```



Diễn giải:

- For each được sử dụng để duyệt tập hợp. Mỗi lần lấy 1 phần tử từ tập hợp và xử lý phần tử đó.

Duyệt mảng

2 vòng lặp thường được sử dụng để duyệt mảng là for và for-each.

```
int [] a = {4, 3, 5, 9};
for (int i = 0; i < a.length; i++) {
    System.out.println(a[i]);
}
```

for(;;)

for-each

```
int [] a = {4, 3, 5, 9};
for (int x : a) {
    System.out.println(x);
}
```

Duyệt mảng

Ví dụ sau tính tổng các số chẵn của mảng.

Lấy từng phần tử từ mảng với for-each

Nếu là số chẵn thì cộng vào tổng

```
int[] a = {9, 3, 8, 7, 3, 9, 4, 2};

double tong = 0;
for(int x : a){
    if(x % 2 == 0){
        tong += x;
    }
}

System.out.print("Tổng: " + tong);
```

DEMO

Nhập mảng số nguyên

+ Tính và xuất trung bình cộng

+ Xuất lập phương các phần tử

Thao tác mảng nâng cao

```
int[] a = {9, 3, 8, 7, 3, 9, 4, 2};

System.out.println("Mảng gốc: " + Arrays.toString(a));
                                  [9, 3, 8, 7, 3, 9, 4, 2]
Arrays.sort(a);
System.out.println("Sau sort: " + Arrays.toString(a));
                                  [2, 3, 3, 4, 7, 8, 9, 9]
int i = Arrays.binarySearch(a, 8);
System.out.println("Vị trí của 8 là " + i);
                    Vị trí của 8 là 5
Arrays.fill(a, 0);
System.out.println("Sau fill: " + Arrays.toString(a));
                                  [0, 0, 0, 0, 0, 0, 0, 0]
```

```
Mảng gốc: [9, 3, 8, 7, 3, 9, 4, 2]
Sau sort: [2, 3, 3, 4, 7, 8, 9, 9]
Vị trí của 8 là 5
Sau fill: [0, 0, 0, 0, 0, 0, 0, 0]
```

DEMO

Nhập mảng 5 phần tử là họ tên của SV và xuất tăng dần theo alphabet họ tên các sinh viên đã nhập.

Thuật toán sắp xếp

- Arrays.sort(mảng) không thể thực hiện:
  - Sắp xếp giảm
  - Các kiểu không so sánh được
- Giải pháp: tự xây dựng thuật toán sắp xếp

```
int a[] = {8,2,6,2,9,1,5};
for(int i=0; i<a.length-1; i++){
    for(int j=i+1; j<a.length; j++){
        if(a[i] > a[j]){
            int temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
}
```

Nếu thay đổi toán tử so sánh thành < thì thuật toán trở thành sắp xếp tăng dần.

DEMO

Nhập 2 mảng họ tên và điểm.

Xuất 2 mảng giảm theo điểm

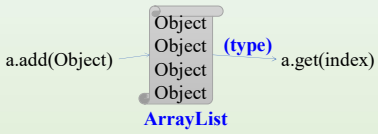
ArrayList là gì?

- Mảng có số phần tử cố định. Vì vậy có các nhược điểm sau:
  - Không thể bổ sung thêm hoặc xóa bớt các phần tử.
  - Lãng phí bộ nhớ
    - ✓ Nếu khai báo mảng với kích thước lớn để nắm giữ một vài phần tử.
    - ✓ Khai báo mảng với kích thước nhỏ thì không đủ chứa
- ArrayList giúp khắc phục nhược điểm nêu trên của mảng.
  ArrayList có thể được xem như mảng động, có thể thêm bớt các phần tử một cách mềm dẻo.
- ArrayList còn cho phép thực hiện các phép toán tập hợp như hợp, giao, hiệu…

ArrayList (không định kiểu)

```
ArrayList a = new ArrayList();
a.add("Cường");
a.add(true);
a.add(1);
a.add(2.5)
Integer x = (Integer)a.get(2);
```

+ Khi add thêm số nguyên thủy thì tự động chuyển sang đối tượng kiểu **wrapper**
+ Khi truy xuất các phần tử, cần **ép về kiểu gốc** của phần tử để xử lý



ArrayList

ArrayList định kiểu

ArrayList

ArrayList (Không định kiểu)

ArrayList có thể chứa các phần tử bất kể loại dữ liệu gì.
+ Các phần tử trong ArrayList được đối xử như một tập các đối tượng (kiểu **Object**)
+ Khi truy xuất các phần tử, cần **ép về kiểu gốc** của phần tử để xử lý

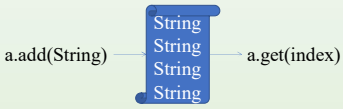
ArrayList<Type> (Có định kiểu)

ArrayList chỉ chứa các phần tử có kiểu đã chỉ định.
+ Khi truy xuất các phần tử **không cần ép** về kiểu gốc của phần tử để xử lý
+ Chặt chẽ, tránh rủi ro lập trình nhầm dữ liệu
+ Hiệu suất xử lý nhanh hơn

ArrayList<Type> định kiểu

```
ArrayList<String> a = new ArrayList<String>();
a.add("Cường");
a.add("Tuấn");
a.add("Phương");
a.add("Hạnh")
String s = a.get(2);
```

+ Khi truy xuất các phần tử **không cần ép** về kiểu gốc của phần tử để xử lý



ArrayList<String>

Chú ý: <Type> là kiểu dữ liệu không phải kiểu nguyên thủy (vd: Integer # ~~int~~)

Thao tác thường dùng

PHƯƠNG THỨC	MÔ TẢ
boolean add(Object)	Thêm vào cuối
void add(int index, Object elem)	Chèn thêm phần tử vào vị trí
boolean remove(Object)	Xóa phần tử
Object remove(int index)	Xóa và nhận phần tử tại vị trí
void clear()	Xóa sạch
Object set(int index, Object elem)	Thay đổi phần tử tại vị trí
Object get(int index)	Truy xuất phần tử tại vị trí
int size()	Số phần tử
boolean contains(Object)	Kiểm tra sự tồn tại
boolean isEmpty()	Kiểm tra rỗng
int indexOf(Object elem)	Tìm vị trí phần tử

Thao tác ArrayList

```
ArrayList<String> a = new ArrayList<String>();
a.add("Cường");     ⟵ [Cường]
a.add("Tuấn");      ⟵ [Cường, Tuấn]
a.add("Phương");    ⟵ [Cường, Tuấn, Phương]
a.add("Hồng");      ⟵ [Cường, Tuấn, Phương, Hồng]
a.add(1, "Hạnh");   ⟵ [Cường, Hạnh, Tuấn, Phương, Hồng]
a.set(0, "Tèo");    ⟵ [Tèo, Hạnh, Tuấn, Phương, Hồng]
a.remove(3)         ⟵ [Tèo, Hạnh, Tuấn, Hồng]
```

Duyệt ArrayList

Duyệt theo **chỉ số** với for hoặc sử dụng **for-each**. Với ArrayList for-each thường được sử dụng hơn

```
ArrayList<Integer> a = new ArrayList<Integer>();
a.add(5);
a.add(9);
a.add(4);
a.add(8)
```

```
for (int i = 0; i < a.size(); i++) {
    Integer x = a.get(i);
    <<xử lý x>>
}
```

```
for (Integer x : a) {
    <<xử lý x>>
}
```

DEMO

Nhập vào danh sách số thực ArrayList<Double>.

Tính tổng và xuất ra màn hình

61

Thao tác ArrayList nâng cao

Lớp tiện ích **Collections** cung cấp các hàm tiện ích hỗ trợ việc xử lý ArrayList

PHƯƠNG THỨC	MÔ TẢ
int binarySearch (List list, Object key)	Tìm kiếm theo thuật toán chia đôi
void fill (List list, Object value)	Gán giá trị cho tất cả phần tử
void shuffle (List list)	Hoán vị ngẫu nhiên
void sort (List list)	Sắp xếp tăng dần
void reverse (List list)	Đảo ngược
void rotate (List list, int distance)	Xoay vòng
void swap(List list, int i, int j)	Tráo đổi

62

Thao tác ArrayList nâng cao

```
ArrayList<Integer> a = new ArrayList<Integer>();
a.add(3);
a.add(9);
a.add(8);
a.add(2);                          // [3, 9, 8, 2]
Collections.swap(a, 0, 2);         // [8, 9, 3, 2]
Collections.shuffle(a);            // [X, X, X, X]
Collections.sort(a);               // [2, 3, 8, 9]
Collections.reverse(a);            // [9, 8, 3, 2]
```

63

DEMO

Nhập danh sách 5 câu hỏi. Tráo ngẫu nhiên và xuất danh sách câu hỏi đã tráo

64

Sắp xếp nâng cao

Có 3 cách sử dụng Collections.**sort()** để sắp xếp ArrayList<Object>

- Cách 1: Collections.**sort(ArrayList)** đối với các phần tử có khả năng so sánh (Integer, Double, String…)
- Cách 2: Collections.**sort(ArrayList, Comparator)** bổ sung tiêu chí so sánh cho các phần tử. Cách này thường áp dụng cho các lớp do người dùng định nghĩa (NhanVien, SinhVien…)
- Cách 3: Xem slide sau

65

Cách 2

Tiêu chí so sánh được chỉ ra để thực hiện việc sắp xếp. Trong bài này tiêu chí so sánh 2 SV là so sánh theo điểm.

```
ArrayList<SV> list = new ArrayList<>();
Comparator<SV> comp = new Comparator<SV>() {
    @Override
    public int compare(SV o1, SV o2) {
        return o1.diemTB.compareTo(o2.diemTB);
    }
};
Collections.sort(list, comp);
```

Kết quả của compare() được sử dụng để sắp xếp o1 và o2. Có 3 trường hợp xảy ra:
✓= 0: o1 = o2
✓> 0: o1 > o2
✓< 0: o1 < o2

66

Cách 3

Tiêu chí so sánh được chỉ ra để thực hiện việc sắp xếp. Trong bài này tiêu chí so sánh 2 people là so sánh theo firstName.

```
Collections.sort( people, (p1, p2) ->
            p1.getFirstName().compareTo(p2.getFirstName()) );
```

```
Collections.sort( people, (p2, p1) ->
            p1.getFirstName().compareTo(p2.getFirstName()) );
```

```
public void sortEmployee(ArrayList<NhanVien> listEmployee) {
    //Truong hop yeu cau sort voi so
//      listEmployee.sort((NhanVien n1, NhanVien n2) -> n2.getTuoi() - n1.getTuoi());

    // truong hop yeu cau sort voi string
    listEmployee.sort((NhanVien n1, NhanVien n2) -> n1.getTen().compareTo(n2.getTen()));
}
```

67

Chuỗi (String)

- String là xâu các ký tự.
  String s = "Hello World ";
- String là một class được xây dựng sẵn trong Java. String có rất nhiều phương thức giúp xử lý chuỗi một cách thuận tiện và hiệu quả.
- String là kiểu dữ liệu được sử dụng nhiều nhất trong lập trình

68

Ký tự đặc biệt

Ký tự	Hiển thị
\t	Ký tự tab
\r	Về đầu dòng
\n	Xuống dòng
\\	\
\"	"

```
System.out.print("\t+ Họ và tên: Tuấn\r\n\t+ Tuổi: 40");
```

```
+ Họ và tên: Tuấn
+ Tuổi: 40
```

69

Thao tác chuỗi

- So sánh
- Tìm vị trí của chuỗi con
- Lấy chuỗi con
- Tách và hợp chuỗi
- Chuyển đổi hoa thường
- Lấy độ dài
- …

```
String fullname = "Nguyễn Văn Tèo";
String first = fullname.substring(0, 6);
```

Nguyễn

70

String API

Phương thức	Mô tả
toLowerCase ()	Đổi in thường
toUpperCase ()	Đổi in hoa
trim()	Cắt các ký tự trắng 2 đầu chuỗi
length()	Lấy độ dài chuỗi
substring()	Lấy chuỗi con
charAt (index)	Lấy ký tự tại vị trí
replaceAll(find, replace)	Tìm kiếm và thay thế tất cả
split(separator)	Tách chuỗi thành mảng

71

String API

Phương thức	Mô tả
equals()	So sánh bằng có phân biệt hoa/thường
equalsIgnoreCase()	So sánh bằng không phân biệt hoa/thường
contains()	Kiểm tra có chứa hay không
startsWith()	Kiểm tra có bắt đầu bởi hay không
endsWith ()	Kiểm tra có kết thúc bởi hay không
matches ()	So khớp với hay không?
indexOf()	Tìm vị trí xuất hiện đầu tiên của chuỗi con
lastIndexOf()	Tìm vị trí xuất hiện cuối cùng của chuỗi con

72

DEMO

Đăng nhập hợp lệ khi mã tài khoản là "hello" và mật khẩu trên 6 ký tự

Thực hiện:

Nhập username và password từ bàn phím

Sử dụng equalsIgnoreCase() để so sánh username và length() để lấy độ dài mật khẩu

```
if(username.equalsIgnoreCase("hello") && password.length() > 6){
        …
}
else{
        …
}
```

73

DEMO

Nhập chuỗi bất kỳ

Tính tổng các số có trong chuỗi.

Ví dụ: chuỗi He2025 thì 2 + 0 + 2 + 5 Kết quả in ra: **9**

74

DEMO

- Quản lý sinh viên
  - Nhập mảng họ tên sinh viên
  - Xuất họ và tên (IN HOA) những sinh viên tên Tuấn hoặc họ Nguyễn
  - Xuất tên những sinh viên có tên lót là Mỹ
- Thực hiện:
  - fullname.**toUpperCase()**: đổi IN HOA
  - fullname.**startsWith("Nguyễn ")**: họ Nguyễn
  - fullname.**endsWith(" Tuấn")**: tên Tuấn
  - fullname.**contains(" Mỹ ")**: lót Mỹ
  - fullname.**lastIndexOf(" ")**: Lấy vị trí trắng cuối cùng
  - fullname.**substring(lastIndex + 1)**: Lấy tên

75

DEMO

- Tìm kiếm và thay thế chuỗi
- Thực hiện theo hướng dẫn sau
  - Nhập chuỗi nội dung, tìm kiếm và thay thế từ bàn phím

```
String content = sc.nextLine();
String find = sc.nextLine();
String replace = sc.nextLine();
```

  - Thực hiện tìm và thay thế:

```
String result = content.replaceAll(find, replace);
```

76

DEMO

Nhập chuỗi chứa dãy số phân cách bởi dấu phẩy và xuất các số chẵn

Thực hiện

Sử dụng split() để tách chuỗi thành mảng bởi ký tự phân cách là dấu phẩy

Duyệt mảng, đổi sang số nguyên và kiểm tra số chẵn

```
String[] daySo = chuoi.split(",")
for(String so : daySo){
    int x = Integer.parseInt(so);
    if(x % 2 == 0){
        Số chẵn
    }
}
```

77

Biểu thức chính qui

Bạn có biết các chuỗi sau đây biểu diễn những gì hay không?

- teo@dtu.edu.vn
- 43-H1-6661
- 43-H1-666.01
- 0913745789
- 192.168.11.200

1. Bạn có biết tại sao bạn nhận ra chúng không?
2. Làm thế nào để máy tính cũng có thể nhận ra như bạn?

78

Biểu thức chính qui

- Máy tính có thể nhận dạng như chúng ta nếu chúng ta cung cấp qui luật nhận dạng cho chúng. Biểu thức chính qui cung cấp qui luật nhận dạng chuỗi cho máy tính.
- Biểu thức chính qui là một chuỗi mẫu được sử dụng để qui định dạng thức của các chuỗi. Nếu một chuỗi nào đó phù hợp với mẫu dạng thức thì chuỗi đó được gọi là so khớp (hay đối sánh).
- Ví dụ: **[0-9]{3,7}**: Biểu thức chính qui này so khớp các chuỗi từ 3 đến 7 ký tự số.
  [0-9]: đại diện cho 1 ký tự số
  {3,7}: đại diện cho số lần xuất hiện (ít nhất 3 nhiều nhất 7)

Ví dụ biểu thức chính qui

```
Scanner scanner = new Scanner(System.in);

System.out.print("Số mobile: ");
String mobile = scanner.nextLine();

String pattern = "0[0-9]{9,10}";

if(mobile.matches(pattern)){
    System.out.println("Bạn đã nhập đúng số mobile");
}
else{
    System.out.println("Bạn đã nhập không đúng số mobile");
}
```

Biểu thức chính qui

Kiểm tra mobile có so khớp với pattern không?

s.matches(regex)

Xây dựng biểu thức chính qui

Regular Expression

Ký tự đại diện	
[xyz]	đại diện một ký tự x, y hay z
[ad-f]	đại diện một ký tự a, d, e hay f
[^xyz]	đại diện ký tự không thuộc [xyz]
\d	tương đương [0-9]
\w	tương đương [0-9a-zA-Z_]
\D	tương đương [^\d]
\W	tương đương [^\w]
\s	đại diện ký tự trắng (\r\n\t\f)
.	đại diện ký tự bất kỳ
^	chỉ ra mẫu bắt đầu
$	chỉ ra mẫu kết thúc
\\, \., \$, \^	đại diện '\', '.', '$' hay '^'

Số lần xuất hiện	
{M,N}	Ít nhất M, nhiều nhất N lần
{N}	Đúng N lần
?	0-1
*	0-N
+	1-N
Không	1

[0-9]{3, 7}

Ví dụ RegEx thường dùng, đơn giản

- Số CMND
  [0-9]{9}
- Số điện thoại di động việt nam
  0\d{9,10}
- Số xe máy sài gòn
  5\d-[A-Z]\d-((\d{4})|(\d{3}\.\d{2}))
- Địa chỉ email
  \w+@\w+(\.\w){1,2}

Ví dụ RegEx

```
Scanner in = new Scanner(System.in);

System.out.print("Email: ");
String email = in.nextLine();

System.out.print("Số điện thoại Huế: ");
String phone = in.nextLine();

String reEmail = "\\w+@\\w+\\.\\w+";
if(!email.matches(reEmail)){
    System.out.println("Không đúng dạng email !");
}

String rePhone = "0543\\d{6}";
if(!phone.matches(rePhone)){
    System.out.println("Không phải số điện thoại ở Huế !");
}
```

Email đơn giản

Số điện thoại để bàn ở Huế

Thực hành - Validation

Nhập thông tin nhân viên từ bàn phím. Thông tin của mỗi nhân viên phải tuân theo các ràng buộc sau. Xuất thông báo lỗi và yêu cầu nhập lại

Thông tin	Kiểm soát	RegEx
Mã nhân viên	5 ký tự hoa	[A-Z]{5}
Mật khẩu	Ít nhất 6 ký tự	.{6,}
Họ và tên	Chỉ dùng alphabet và ký tự trắng	[a-zA-Z]+
Email	Đúng dạng email	\w+@\w+(\. \w+){1,2}
Điện thoại	Điện thoại Sài gòn	083\d{7}
Số xe máy	Số xe máy Sài gòn	5\d-[A-Z]-((\d{4})|(\d{3}\.{2}))
Số CMND	10 chữ số	\d{10}
Website	Địa chỉ website	http://www\.\w+\.\w{2,4}

Bài tập

Sinh viên thực hiện các bài tập trong sách

85