

BÀI 2. CẤU TRÚC LẬP TRÌNH TRONG JAVA

Khối lệnh
if...else..., switch
while, do...while, for
Mảng
Xâu ký tự

1

1. Khối lệnh

- Nhóm các lệnh được bao bằng cặp dấu { }
- Thực hiện các lệnh một cách tuần tự
- Phạm vi của biến:
 - Biến chỉ có phạm vi sử dụng trong khối lệnh đã khai báo
 - Với các khối lệnh lồng nhau có khai báo biến trùng tên, biến ở khối lệnh trong được ưu tiên

```
{
    int n = 0;
    n = n + 1; //n = 1
    {
        int n = 10, m = 10;
        n = m + 1; //n = 11
    }
    m = m + 1; //Sai cú pháp
}
```

2

2. CẤU TRÚC RỄ NHÁNH

if...else...,
switch

3

Cấu trúc if, if...else

```
if (boolean_expr){ //Nếu boolean_expression là true  
    //do something  
}
```

```
if (boolean_expr){ //Nếu boolean_expression là true  
    //do task1  
}  
else { //Nếu boolean_expression là false  
    //do task2  
}
```

4

Cấu trúc if...else lồng nhau

```

if (boolean_expr1){
    //do task1
}
else if (boolean_expr2) {
    //do task2
}
...

else if (boolean_exprN){
    //do taskN
}
else{
    //do other task
}

```

5

Chú ý

- Tương đương ?

```

if (boolean_expr)
    //do task1
else
    //do task2

```



```

if (boolean_expr)
    //do task1
if (!boolean_expr)
    //do task2

```

```

if (boolean_expr1)
    if (boolean_expr2)
        //do task1
else
    //do task2

```



Hoặc?

```

if (boolean_expr1){
    if (boolean_expr2)
        //do task1
    else
        //do task2
}

```



```

if (boolean_expr1){
    if (boolean_expr2)
        //do something
}
else
    //do something

```

6

Ví dụ

- Bài toán: Giải phương trình bậc 1
 - Đầu vào: Hai hệ số a, b
 - Đầu ra: Nghiệm của phương trình $ax + b = 0$

7

Cấu trúc switch

- `controlling_expr` phải trả về kiểu dữ liệu `byte, short, char, int, String`
- `value`: giá trị có kiểu `byte, short, char, int, String`
- `break`: thoát khỏi cấu trúc `switch`
- `default`: các giá trị còn lại

```
switch (controlling_expr){
    case value1:    //do task1
                    break;
    case value2:    //do task 2
                    break;
    ...
    case valueN:    //do taskN
                    break;
    default:       //do other task
}
```

8

Ví dụ

```
switch (number) {  
    case 1:    System.out.print("One");  
               break;  
    case 2:    System.out.print("Two");  
               break;  
    case 3:    System.out.print("Three");  
               break;  
    default:   System.out.print("I don't know.");  
}
```

9

Cấu trúc switch - Nhóm giá trị

```
switch (month) { }  
    case 1:  
    case 3:  
    case 5:  
    case 7:  
    case 8:  
    case 10:  
    case 12:   System.out.print("The month has 31 days");  
                break;  
    case 4:  
    case 6:  
    case 9:  
    case 11:   System.out.print("The month has 30 days");  
                break;  
    default:   System.out.print("The month has 28 or 29 days");  
}
```

10

2. CẤU TRÚC LẶP

while

do...while

for

Lệnh thay đổi cấu trúc: break, continue

11

Cấu trúc while và do...while

- Thực hiện lặp đi lặp lại một công việc khi biểu thức `boolean_expr` còn có giá trị `true`

```
while (boolean_expr) {
    //do something
}
```

```
do {
    //do something
} while (boolean_expr);
```

- Công việc có thể không được thực hiện lần nào
- Công việc được thực hiện tối thiểu 1 lần

12

Ví dụ

- Sử dụng cấu trúc while

```
int n, i, factorial = 1;
i = 1;
while(i <= n){
    factorial *= i;
    i++;
}
```

- Sử dụng cấu trúc do...while

```
int n, i, factorial = 1;
i = 1;
do{
    factorial *= i;
    i++;
} while(i <= n);
```

13

Cấu trúc for

- Cú pháp

```
for (start_expr; loop_condition; loop_increment){
    //do something
}
```

- Trong đó:

- start_expr : Biểu thức khởi tạo
- loop_condition : Biểu thức điều kiện thực hiện vòng lặp
- loop_change : Biểu thức thay đổi biến

- Ví dụ

```
int n, factorial = 1;
for (int i = 1; i < n; i++){
    factorial *= i;
}
```

14

Các lệnh thay đổi cấu trúc vòng lặp

- **continue**

- Bỏ qua việc thực hiện các câu lệnh nằm sau lệnh **continue** trong thân vòng lặp.
- Chuyển sang thực hiện một vòng lặp mới

- **break**

- Thoát khỏi vòng lặp ngay cả khi biểu thức điều kiện của vòng lặp vẫn còn được thỏa mãn.

- Hai dạng:

- Không gán nhãn: Chuyển ra ngoài vòng lặp, thực hiện câu lệnh ngay sau vòng lặp
- Gán nhãn: Chuyển ra ngoài vòng lặp, thực hiện câu lệnh tiếp theo sau vòng lặp được đánh dấu bởi nhãn

15

Ví dụ

```
int sum = 0;
outer: for(int i = 0; i < 10; i++){
    inner: for(int j = i; j < 10; j++){
        sum ++;
        if (j == 1) continue;
        if (j == 2) continue outer;
        if (j == 3) break;
        if (j == 4) break outer;
    } // terminate inner
} // terminate outer
System.out.println(" sum = " + sum);
```

16

3. XÂU KÝ TỰ

17

String trong Java

- Được xây dựng như là một lớp trong Java
- Khai báo một đối tượng

```
String variable = new String(literalString);
String variable = new String(char[] charArr)
```

Hoặc đơn giản hơn:

```
String variable = literalString;
```

- Trong đó:

variable: biến

literalString: Giá trị hằng xâu ký tự

charArr: Mảng ký tự

- Ví dụ:

```
String myUniversity = "HUST";
```

18

Các phương thức của String

- `char charAt(int index)`: trả về ký tự có chỉ số index
 - Chỉ số bắt đầu từ 0
- `int length()`: trả về kích thước của chuỗi
- `String toLowerCase()`: chuyển thành chữ thường
- `String toUpperCase()`: chuyển thành chữ thường
- `int compareTo(String anotherString)`: so sánh hai chuỗi theo thứ tự từ điển, ưu tiên chữ thường. Trả về:
 - `= 0`: 2 chuỗi giống nhau
 - `< 0`: chuỗi nhỏ hơn `anotherString`
 - `> 0`: chuỗi lớn hơn `anotherString`
- `int compareToIgnoreCase(String str)`: so sánh không kể chữ hoa, chữ thường

19

Các phương thức của String (tiếp)

- `boolean equals (Object object)`: so sánh với một đối tượng bất kỳ
 - Trong Java, một đối tượng bất kỳ đều có thể chuyển thành String
- `boolean equalsIgnoreCase(String anotherString)`: so sánh với một đối tượng bất kỳ, không phân biệt chữ hoa chữ thường
- `char[] toCharArray`: chuyển chuỗi thành mảng ký tự
- `String concat(String str)`: ghép nội dung `str` vào cuối chuỗi
- `int indexOf(int ch)`: trả về chỉ số của ký tự đầu tiên có giá trị bằng `ch`
- `int indexOf(String str)`: trả về vị trí của `str` trong chuỗi

20

Các phương thức của String (tiếp)

- `String replace (char oldChar, char newChar)`: trả về xâu mà tất cả ký tự `oldChar` trong xâu thành `newChar`
- `String replace (String oldString, String newString)`
- `String replaceFirst (String oldString, String newString)`
- `String[] split (String regex)`: chia xâu thành các xâu con bởi xâu `regex`
- `String trim()`: trả về xâu được loại bỏ dấu cách ở đầu và cuối

21

Các phương thức gọi trực tiếp từ lớp String

- `static String copyValueOf(char[] data)`: trả về xâu có chứa các ký tự của mảng `data`
- `static String copyValueOf(char[] data, int offset, int count)`: trả về xâu có chứa `count` ký tự từ chỉ số `offset` của mảng `data`
- `static String format()`: trả về xâu hiển thị giá trị của đối tượng bất kỳ theo định dạng
- `static String valueOf()`: trả về xâu chứa nội dung của một đối tượng nào đó

22

4. MẢNG

23

Khai báo mảng

- Mảng là tập hợp hữu hạn các phần tử cùng kiểu
- Số lượng phần tử xác định khi khai báo, không đổi
- Cú pháp

```
DataType[] array = new DataType[size];
```

```
DataType array[] = new DataType[size];
```

```
DataType[] array = {value1, value2, ..., valueN};
```

Trong đó:

array: Biến mảng

size: Số phần tử trong mảng, có thể sử dụng giá trị, biến, biểu thức

Value1, ... valueN : các giá trị khởi tạo

24

Mảng nhiều chiều

- Được coi là mảng của các mảng:
 - Mảng 2 chiều: mảng các mảng 1 chiều
 - Mảng 3 chiều: mảng của các mảng 2 chiều

- Khai báo mảng 2 chiều:

```
DataType[] [] array = new DataType[size1][size2];
DataType array[] [] = new DataType[size1][size2];
DataType[] [] array = {value11,value12,...,value1N}
                        {value21,value22,...,value2N};
```

- Tương tự cho mảng nhiều chiều khác

25

Thao tác với mảng

- Phương thức `length()`: Lấy số phần tử của mảng
- Truy cập vào phần tử của mảng 1 chiều: `array[index]`
`index`: chỉ số, bắt đầu từ 0
- Truy cập vào phần tử của mảng nhiều chiều:
`array[index1][index2]...[indexN]`

26

Ví dụ - Tìm kiếm trên mảng

```
/** The Search class lookup a key in the array . A message
is displayed to notify the index of the key in the array*/
import java.util.Scanner;
public class Search {
    /** The main method begins execution of Java application
    *@param args: input parameter
    */
    public static void main (String[] args) {
        int size = 0;
        Scanner inputData = new Scanner(System.in);
        System.out.print("Enter the size of the array:");
        size = inputData.nextInt();
        int[] intArr = new int[size];
    }
}
```

27

Ví dụ: Tìm kiếm trên mảng(tiếp)

```
//Enter value for each of element
System.out.println("Enter value for array:");
for(int i = 0; i < size; i++){
    System.out.print("intArr[" + i + "] = ");
    intArr = inputData.nextInt();
}

//Look up the key
int key = 0;
System.out.println("Enter value for array:");
key = inputData.nextInt();
boolean found = false;
```

28

Ví dụ: Tìm kiếm trên mảng(tiếp)

```
for(int i = 0; i < size; i++){
    if (intArr[i] == key){
        System.out.println("Found " + key +"at " + i);
        found = true;
        break;
    }
}
//Notify if could not find the key
if(!found)
    System.out.println ("Could not find " + key);
}
```

29