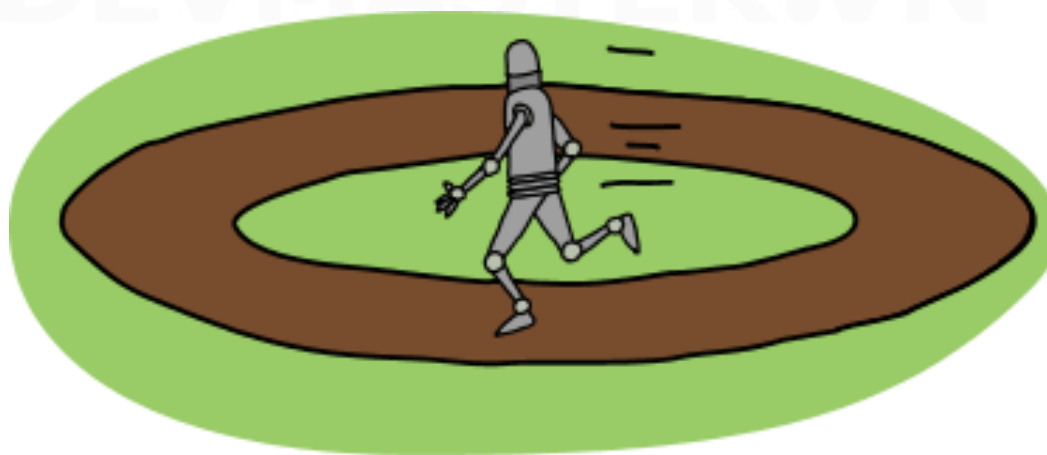


Bài 3

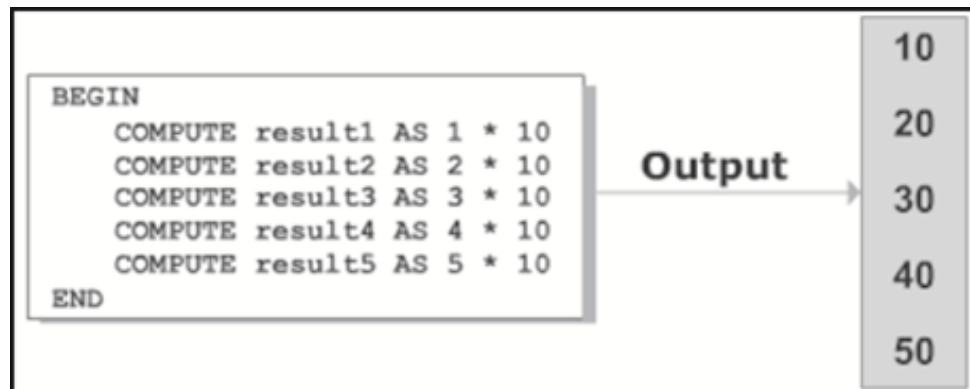
Cấu trúc điều khiển Lặp

- Khái niệm vòng lặp
- Do-While
- While
- For, for-each
- Lệnh nhảy



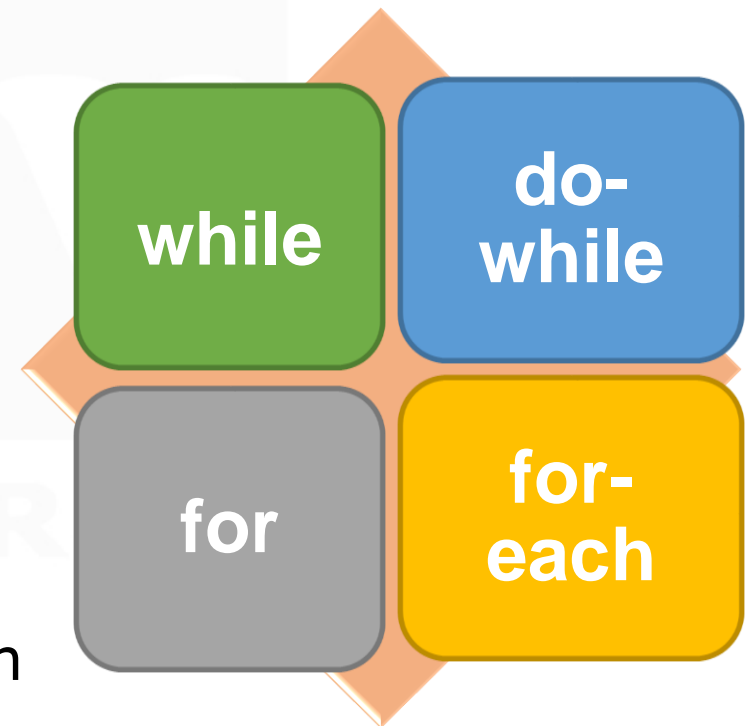
Vòng lặp là gì?

- Một chương trình máy tính bao gồm một tập hợp các câu lệnh (**statements**), chúng được thực hiện **tuần tự lần lượt**.
- Tuy nhiên, trong những tình huống nhất định có khối lệnh cần được thực hiện **lặp đi lặp lại** các bước nhất định để đáp ứng điều kiện quy định



Vòng lặp có ý nghĩa trong lập trình:

- Cho phép viết chương trình **ngắn gọn**.
- Bao gồm một lệnh hoặc khối lệnh có thể được **thực hiện nhiều lần**.
- Lệnh (khối lệnh) được thực hiện lặp đi lặp lại cho đến khi một **điều kiện** được đánh giá đúng (hoặc sai).



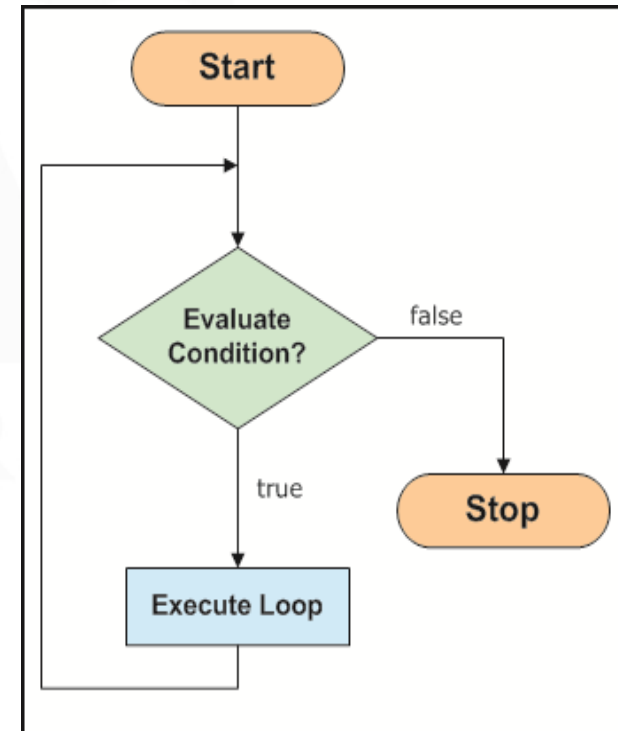
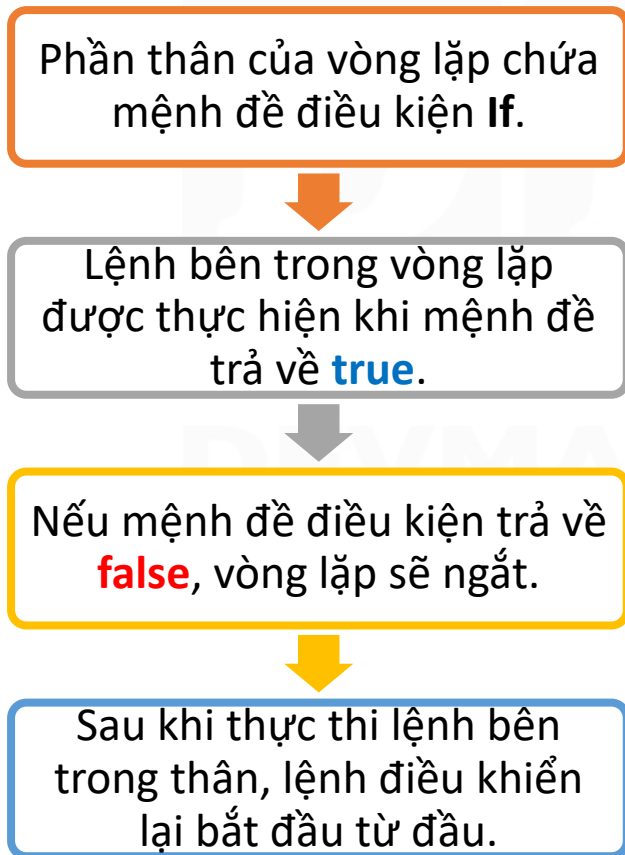
Vòng lặp **While**

- Là vòng lặp có cú pháp **cơ bản** nhất.
- Lệnh (khối lệnh) được thực thi khi điều kiện được xét là **ĐÚNG** (True).
- Vòng lặp này thường được dùng khi **số lần** thực hiện **không xác định**.

Cú pháp

```
while ( đ i ề n k i ệ n ) {  
  
    // một hoặc nhiều dòng lệnh...  
}
```

Vòng lặp **While** thực hiện theo trình tự:



Mẫu code ví dụ sau thể hiện bảng cửu chương 10 với vòng lặp **While**:

```
public class PrintMultiplesWithWhileLoop {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main (String[] args) {  
        // Variable, num acts as a counter variable  
        int num = 1 ;  
        // Variable, product will store the result  
        int product = 0 ;  
    }  
}
```

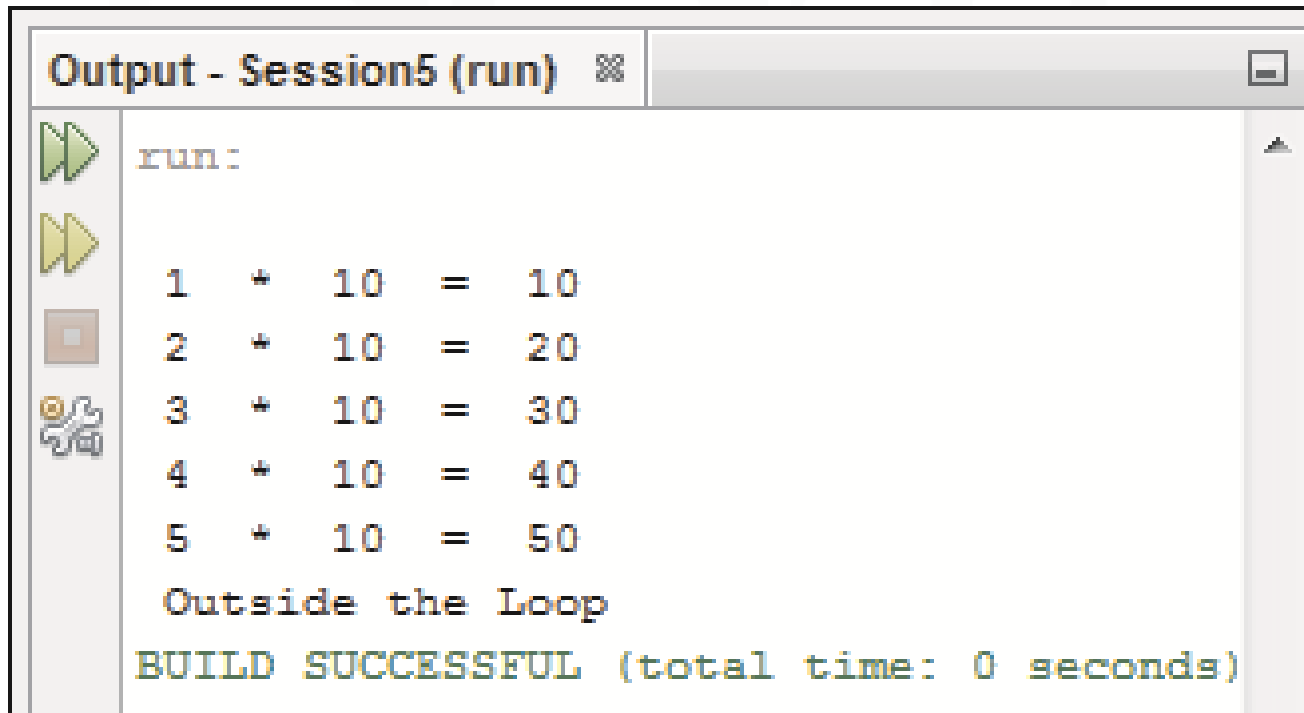
Mệnh đề điều kiện giới hạn số **num** nhỏ hơn hoặc bằng **5**:

```
// Tests the condition at the beginning of the loop
while (num <= 5) {
    product = num * 10;
    System.out.printf("\n %d * 10 = %d", num, product);
    num++; // Equivalent to n = n + 1
} // Moves the control back to the while statement
```

```
// Statement gets printed on loop termination
System.out.println("\n Outside the Loop");
}
}
```


While

Kết quả:



The screenshot shows an IDE output window titled "Output - Session5 (run)". The output text is as follows:

```
run:
1 * 10 = 10
2 * 10 = 20
3 * 10 = 30
4 * 10 = 40
5 * 10 = 50
Outside the Loop
BUILD SUCCESSFUL (total time: 0 seconds)
```

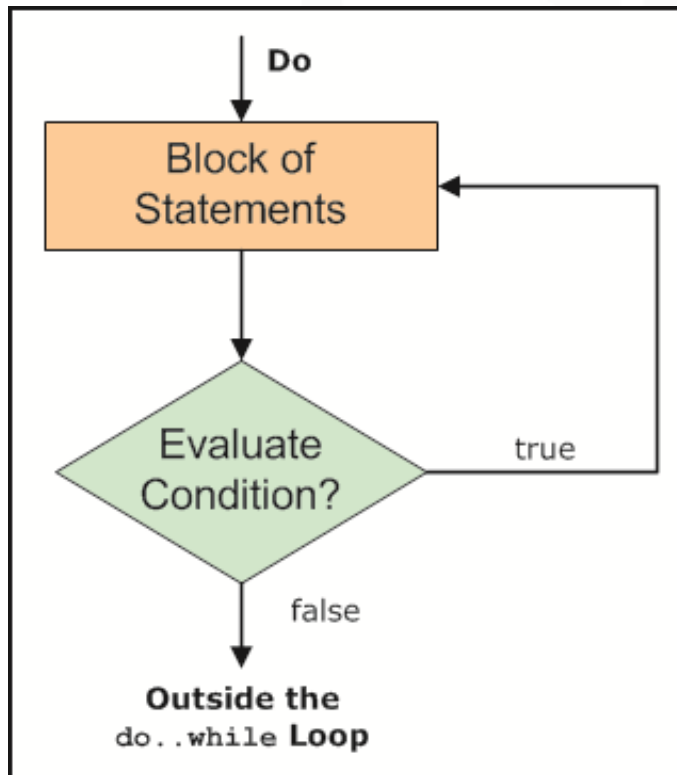
Vòng lặp **do - while**

- Khá tương đồng vòng lặp while, chỉ khác là điều kiện tồn tại vòng lặp được xem xét ở **cuối cùng**.
- Luôn đảm bảo **phần thân** (body) của vòng lặp được thực hiện ít nhất **1 lần**.

Cú pháp

```
do {  
  
    // Câu lệnh...;  
  
} while ( đ i ề u k i ệ n );
```

Vòng lặp **do-while** thực hiện theo trình tự:



Với mỗi lần lặp, do-while luôn thực hiện phần khối lệnh trong **thân** trước khi kiểm tra điều kiện

Nếu điều kiện là **true**, vòng lặp tiếp tục thực hiện lặp lại lần nữa.

Nếu điều kiện là **false**, vòng lặp kết thúc.

Những lệnh tiếp theo sau vòng lặp được thực hiện.

Mã lệnh **do-while** mẫu:

```
public class SumOfNumbers {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        int num = 1, sum = 0;  
  
        /**  
         * The body of the loop is executed first, then the condition is  
         * evaluated  
         */  
        do {  
            sum = sum + num;  
            num++;  
        } while (num <= 10);  
  
        // Prints the value of variable after the loop terminates  
        System.out.printf("Sum of 10 Numbers: %d\n", sum);  
    }  
}
```

Vòng lặp **for**

- Sử dụng khi đã **biết trước số lần lặp** của khối lệnh.
- Khối lệnh được thực hiện khi điều kiện tồn tại là **true**.
- **Điều kiện** luôn được **kiểm tra** trước mỗi lần lặp.

Cú pháp

```
f or (initialization; condition; increment/decrement) {  
    // one or more statements  
}
```

Vòng lặp **for** thực hiện theo trình tự:

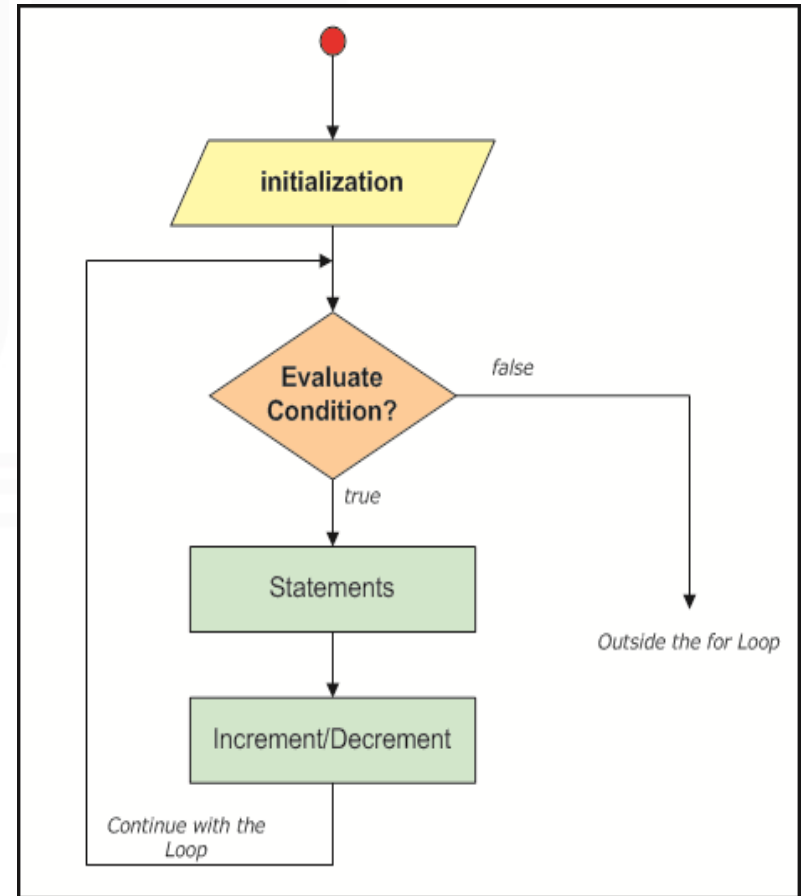
Biến đếm khởi tạo chỉ 1 lần khi vòng lặp bắt đầu.

Tiếp theo mệnh đề điều kiện kiểm tra giá trị biến đếm có thỏa mãn giá trị mục tiêu.

Nếu mệnh đề là true – phần thân vòng lặp được thực hiện, ngược lại sẽ bị ngắt.

Cuối cùng phần lặp của vòng lặp thực thi, biểu thức này thường tăng hoặc giảm giá trị biến điều khiển (biến đếm)

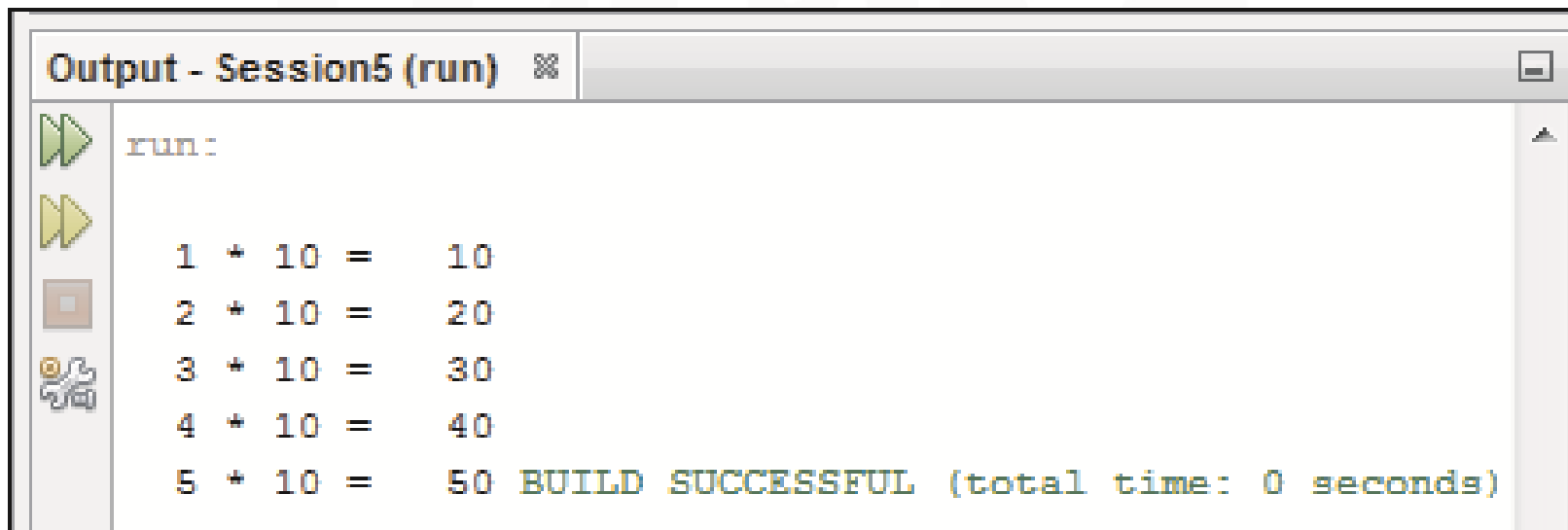
Trong lần lặp kế tiếp, mệnh đề điều kiện được đưa ra đánh giá...



Mã lệnh **for** mẫu:

```
public class PrintMultiplesWithForLoop {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        int num, product;  
  
        // The for Loop with all the three declaration parts  
        for (num = 1; num <= 5; num++) {  
            product = num * 10;;  
            System.out.printf("\n % d * 10 = % d ", num, product);  
        } // Moves the control back to the for loop  
    }  
}
```

Kết quả:



```
run:
1 * 10 = 10
2 * 10 = 20
3 * 10 = 30
4 * 10 = 40
5 * 10 = 50 BUILD SUCCESSFUL (total time: 0 seconds)
```


Vòng lặp **for** với bổ sung biểu thức (sử dụng dấu ,):

```
// Use commas in a for statement.  
class Comma {  
    public static void main(String args[]) {  
        int i, j;  
  
        for(i=0, j=10; i < j; i++, j--) ← Notice the two loop  
            System.out.println("i and j: " + i + " " + j); control variables.  
    }  
}
```

The output from the program is shown here:

```
i and j: 0 10  
i and j: 1 9  
i and j: 2 8  
i and j: 3 7  
i and j: 4 6
```

Vòng lặp **for** không đầy đủ thành phần:

```
/*  
 * The for loop starts with num value 1 and continues till value of  
 * flag is not true  
 */  
    for (; ! flag; num++) {  
        System.out.println("Value of num: " + num);  
        if (num == 5) {  
            flag = true;  
        }  
    } // End of for loop  
}
```

Vòng lặp **for** vô tận:

```
.....  
  
    for( ; ; ) {  
        System.out.println("This will go on and on");  
    }  
  
.....
```

Vòng lặp **for - each**

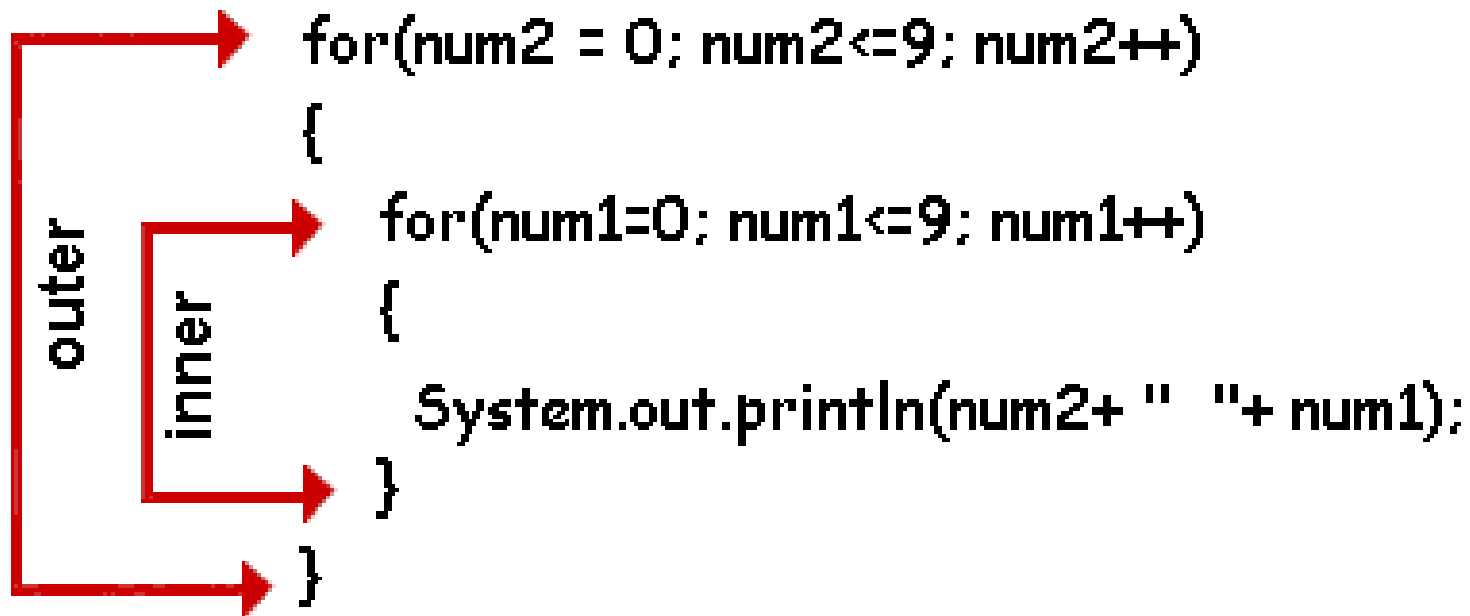
- Được thiết kế để lấy lần lượt tuần tự các giá trị trong một tập hợp đối tượng (ví dụ như mảng).
- Số lần thực hiện lặp bằng đúng số lượng phần tử của tập hợp.

Cú pháp

```
for (type var: collection) {  
    // block of statement  
}
```

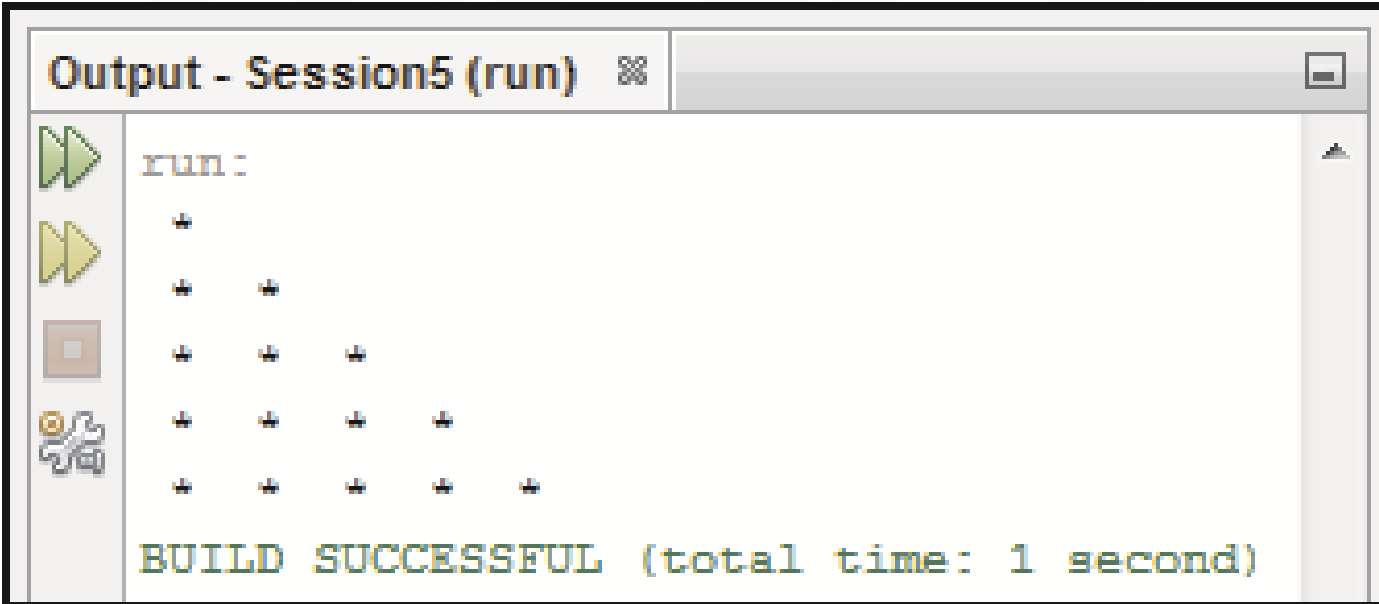
Vòng lặp **lồng**:

- Các vòng lặp ở trên có thể thực hiện đặt lồng nhau.
- Có thể đặt trong vòng lặp bất kỳ 1/3 loại vòng lặp.



Vòng lặp lồng

Sử dụng vòng lặp **lồng** in dữ liệu như hình



```
Output - Session5 (run)
run:
*
*  *
*  *  *
*  *  *  *
*  *  *  *  *
BUILD SUCCESSFUL (total time: 1 second)
```

1

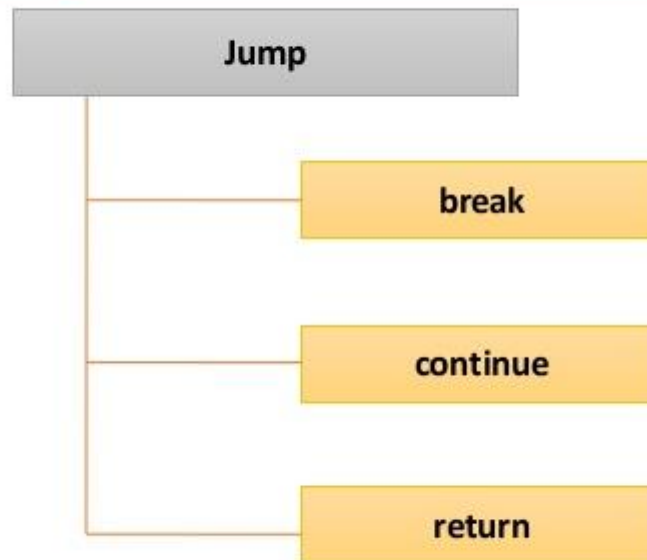
```

* * * * *
  * * * *
    * * *
      * *
        *
      * * *
    * * * *
  * * * * *
* * * * *

```

Java cung cấp 3 từ khóa: **break**, **continue** và **return** sử dụng để thay đổi luồng điều khiển dựa trên điều kiện

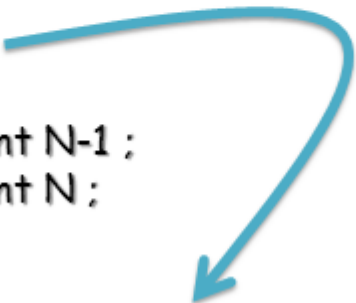
Jump Statements



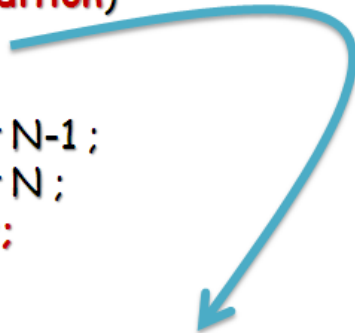
25

Nếu sử dụng trong vòng lặp, ngay lập tức nó sẽ dừng vòng lặp hiện tại và chuyển điều khiển cho dòng lệnh tiếp theo.

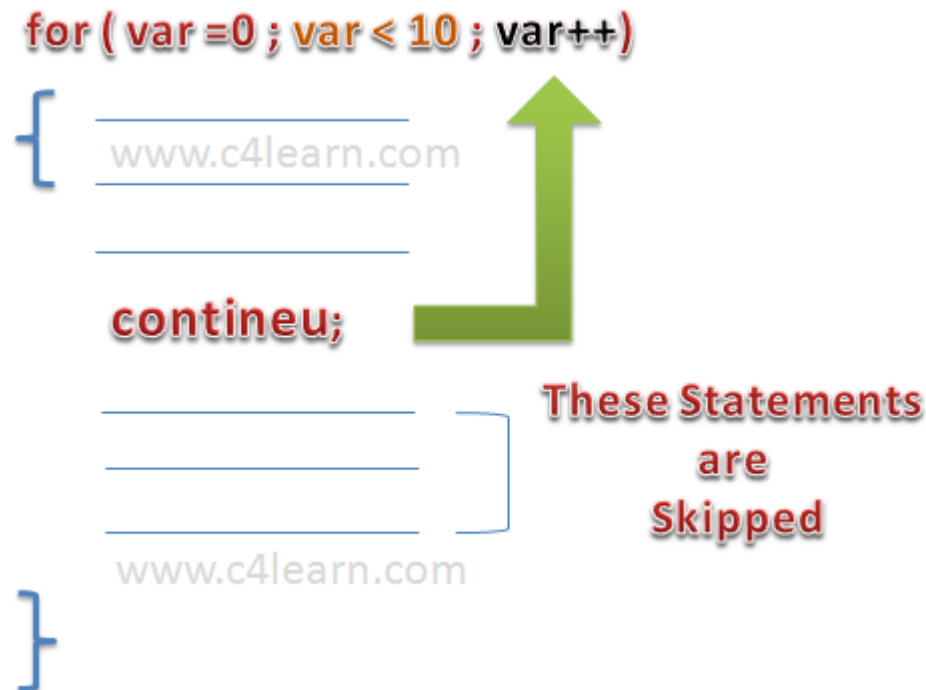
```
for ( initialization ; condition ; increment )  
{  
    Statement 1 ;  
    Statement 2 ;  
    Statement 3 ;  
    .....  
    .....  
    .....  
    break;  
    .....  
    Statement N-1 ;  
    Statement N ;  
}  
OutsideStatement 1;
```



```
Initialization;  
while (condition)  
{  
    Statement 1 ;  
    Statement 2 ;  
    Statement 3 ;  
    .....  
    .....  
    if ( If Condition )  
        break;  
    .....  
    Statement N-1 ;  
    Statement N ;  
    Increment ;  
}  
OutsideStatement 1;
```



Nếu sử dụng trong vòng lặp, ngay lập tức nó sẽ bỏ qua các lệnh còn lại trong thân vòng lặp và chuyển tới lần lặp kế tiếp.



Được sử dụng với hàm có yêu cầu giá trị trả về.

return type **method name** **value passed to the method**

```
int total( int aNumber) {  
  
    int a_Value = aNumber + 10;  
  
    return a_Value;  
  
}
```

- ✓ Vòng lặp cho phép lập trình ngắn gọn.
- ✓ Có 3 vòng lặp trong Java là: **while**, **do-while** và **for**.
- ✓ Các vòng lặp có thể lồng nhau không giới hạn.
- ✓ Java cung cấp 3 từ khóa lệnh nhảy (jump statement) là: **break**, **continue**, **return**.



Thank for watching!