

“Nuts and Bolts” (*)



covering

EXTRA EXAMPLES

- ** variables
- ** assignment
- ** keywords
- ** primitive types
- ** basic operations
- ** control structures



Chapter 3 (sections 3.1–3.5; 3.8) – “Core Java” book

Chapters 1+3 – “Head First Java” book

Chapters 2–4 – “Introduction to Java Programming” book

Chapter 2 (sections 2.1 – 2.5) – “Java in a Nutshell” book

(*) Basic, practical details of Java.



Operator Precedence: Examples 1+2

```
int a = 4;  
int b = 1;  
int c = 3;  
int result = a - b + c;  
System.out.println(result);
```

both operators are of equal
precedence in our table



∴ evaluate left to right!
result = 6 (not 0)

```
int a = 4;  
int b = 1;  
int c = 3;  
int result = a = b = c;  
System.out.println(result);
```

both operators are of equal
precedence in our table



∴ evaluate right to left!
result = 3

Operator Precedence: Examples 3 – 6

Post incrementing

```
int a = 4;
int result = a++ + a;
System.out.println("result = " + result + ", a = " + a );
```

result = ? a = ?

```
a = 4;
result = a + a++;
System.out.println("result = " + result + ", a = " + a );
```

result = ? a = ?

Pre incrementing

```
int a = 4;
int result = ++a + a;
System.out.println("result = " + result + ", a = " + a );
```

result = ? a = ?

```
a = 4;
result = a + ++a;
System.out.println("result = " + result + ", a = " + a );
```

result = ? a = ?

Examples: Moving the `i++` and resulting behaviour

```
int i = 0;
do {
    i++;
    System.out.println("i = " + i);
} while (i < 3);
```

Generates: `i = 1`
`i = 2`
`i = 3`

OR

```
int i = 0;
while (i < 3) {
    i++;
    System.out.println("i = " + i);
}
```

Generates: `i = 1`
`i = 2`
`i = 3`

```
int i = 3;
do {
    System.out.println("i = " + i);
    i++;
} while (i < 3);
```

Generates:
`i = 3`

OR

```
int i = 3;
while (i < 3) {
    System.out.println("i = " + i);
    i++;
}
```

Generates:
`*nothing*`

Using labelled statement blocks with break (1/2)

```
class TestBreakLabel {
    public static void main(String[] args) {
        outer: ←
            for (int i = 1; i < 5; i++) {
                System.out.println("Begin outer for i=" + i);
                inner:
                    for (int j = 1; j < 5; j++) {
                        if (j == i) break outer;
                        System.out.println("    inner: i=" + i + " j= " + j);
                    }
                System.out.println("End outer for i=" + i);
            }
        System.out.println("Finished.");
    }
}
```

This is just a **line marker**, **NOT** a block.



Use carefully as it can result in 'messy' code!

```
Mark CMD
> javac TestBreakLabel.java
> java TestBreakLabel
Begin outer for i=1
Finished.
>
```

Using labelled statement blocks with break (2/2)

```
class TestBreakNoLabel {  
    public static void main(String[] args) {  
        outer:  
        for (int i =1; i < 5; i++) {  
            System.out.println("Begin outer for i=" + i);  
            inner:  
            for (int j = 1; j < 5; j++) {  
                if (j == i) break;  
                System.out.println(  
                    "    inner: i=" +  
                    i + " j=" + j );  
            }  
            System.out.println(  
                "End outer for i=" + i);  
        }  
        System.out.println("Finished.");  
    }  
}
```



Use carefully as it can
result in 'messy' code!

```
C:\ CMD  
>javac TestBreakNoLabel.java  
>java TestBreakNoLabel  
Begin outer for i=1  
End outer for i=1  
Begin outer for i=2  
    inner: i=2 j=1  
End outer for i=2  
Begin outer for i=3  
    inner: i=3 j=1  
    inner: i=3 j=2  
End outer for i=3  
Begin outer for i=4  
    inner: i=4 j=1  
    inner: i=4 j=2  
    inner: i=4 j=3  
End outer for i=4  
Finished.  
>
```

Using labelled statement blocks with continue (1/2)

```
class TestContinueLabel {  
    public static void main(String[] args) {  
        outer:  
        for (int i = 1; i < 5; i++) {  
            System.out.println("Begin outer for i="+i);  
            inner:  
            for (int j = 1; j < 5; j++) {  
                if (j == i) continue outer;  
                System.out.println(  
                    "    inner: i=" +  
                    i + " j=" + j);  
            }  
            System.out.println(  
                "End outer for i=" + i);  
        }  
        System.out.println("Finished.");  
    }  
}
```



Use carefully as it can
result in 'messy' code!

```
> javac TestContinueLabel1.java  
  
> java TestContinueLabel1  
Begin outer for i=1  
Begin outer for i=2  
    inner: i=2 j=1  
Begin outer for i=3  
    inner: i=3 j=1  
    inner: i=3 j=2  
Begin outer for i=4  
    inner: i=4 j=1  
    inner: i=4 j=2  
    inner: i=4 j=3  
Finished.  
>
```

Using labelled statement blocks with continue (2/2)

```
class TestContinueNoLabel {  
    public static void main(String[] args) {  
        outer:  
        for (int i = 1; i < 5; i++) {  
            System.out.println(  
                "Begin outer for i=" + i);  
            inner:  
            for (int j = 1; j < 5; j++) {  
                if (j == i) continue;  
                System.out.println(  
                    "    inner: i=" +  
                    i + " j=" + j );  
            }  
            System.out.println(  
                "End outer for i=" + i);  
        }  
        System.out.println("Finished.");  
    }  
}
```

```
C:\> CMD  
>java TestContinueNoLabel  
Begin outer for i=1  
    inner: i=1 j=2  
    inner: i=1 j=3  
    inner: i=1 j=4  
End outer for i=1  
Begin outer for i=2  
    inner: i=2 j=1  
    inner: i=2 j=3  
    inner: i=2 j=4  
End outer for i=2  
Begin outer for i=3  
    inner: i=3 j=1  
    inner: i=3 j=2  
    inner: i=3 j=4  
End outer for i=3  
Begin outer for i=4  
    inner: i=4 j=1  
    inner: i=4 j=2  
    inner: i=4 j=3  
End outer for i=4  
Finished.
```



Use carefully as it can
result in 'messy' code!