Execution Flow Control in Java

- Using Selection Statements
- Iteration Statements
- Block Breaker Statements

Objectives

- Develop code that implements an `if` or `switch` statement; and identify legal argument types for these statements.
- Develop code that implements all forms of loops and iterators, including the use of `for`, the enhanced for loop (for-each), `do`, `while`, labels, `break`, and `continue`; and explain the values taken by loop counter variables during and after loop execution.

Using Selection Statements

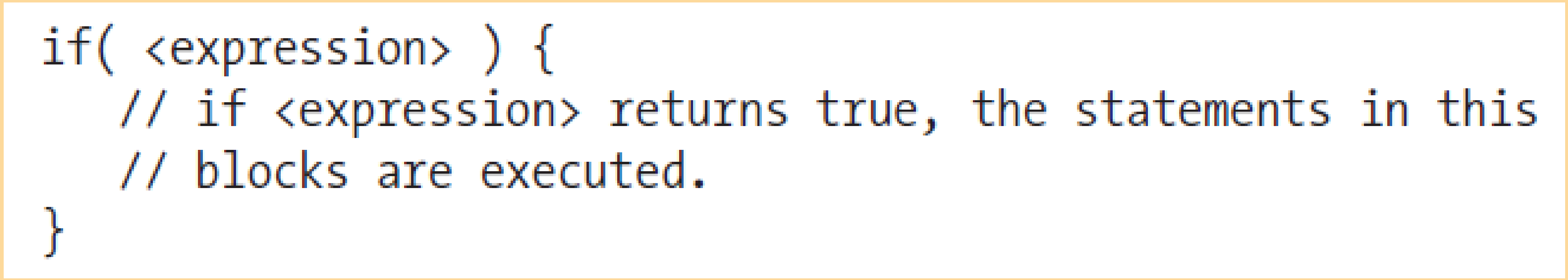
- A selection statement allows the conditional execution of a block of statements. If a condition is true, a block of statements will be executed once, else it will be skipped.
- Two types:
    - The `if` Statements
    - The `switch` Statement

The if Statements

- The if Construct
- The if-else Construct
- The if-else if Construct
- The if-else if-else Construct
- Summary of the if Constructs

The if Construct

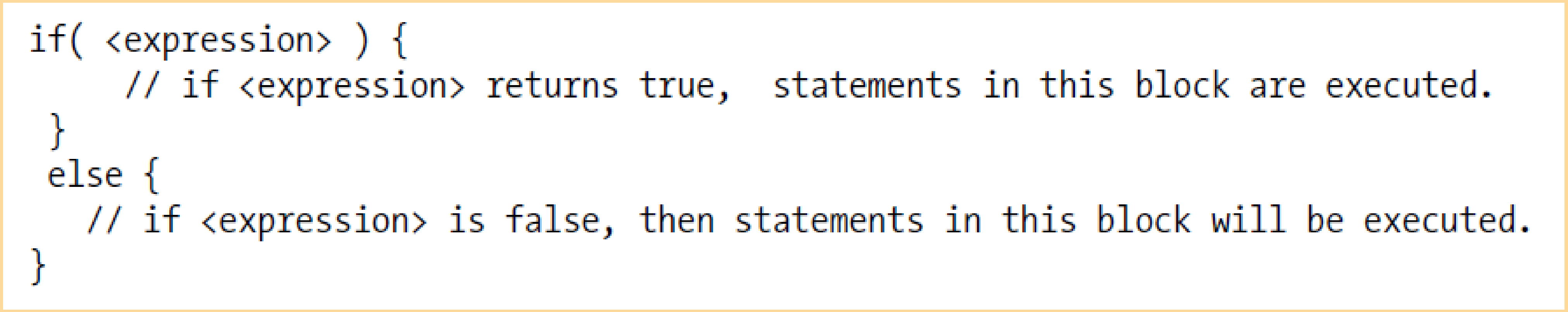
- The `if` construct allows the execution of a single statement or a block of statements
- The `<expression>`: a boolean value
- = is an assignment operator and not the comparison operator.



```
if( <expression> ) {
   // if <expression> returns true, the statements in this
   // blocks are executed.
}
```

The if-else Construct

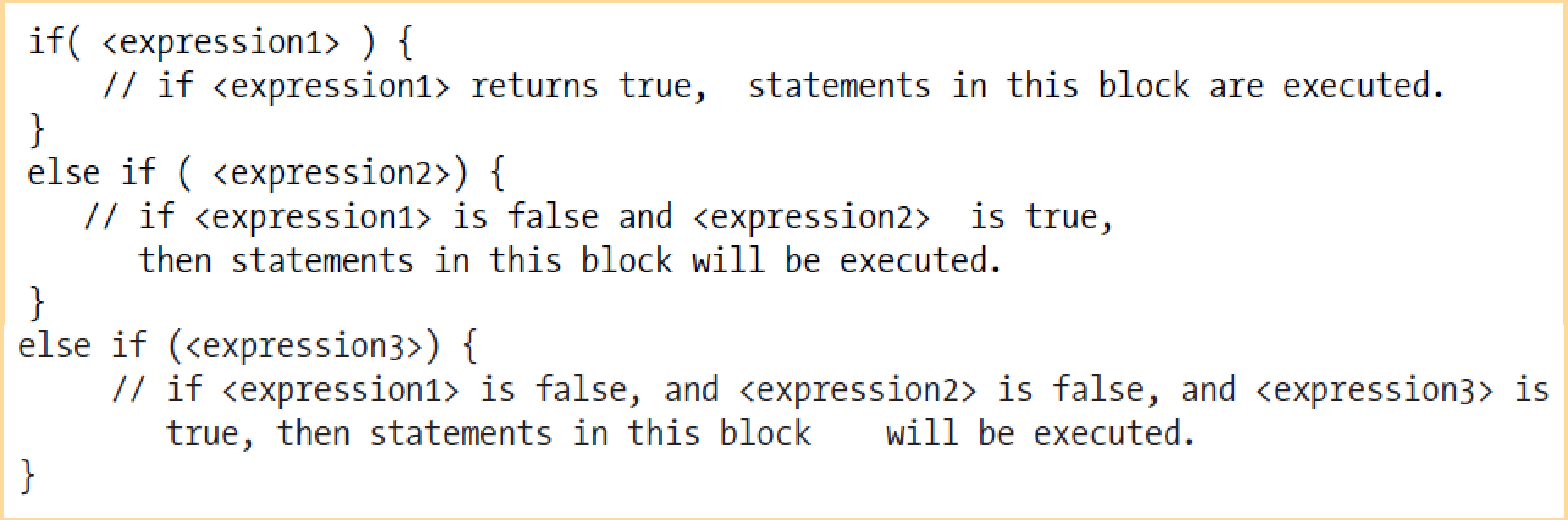
- If a condition is true, the first block of code will be executed, otherwise the second block of code will be executed.



```
if( <expression> ) {
     // if <expression> returns true,  statements in this block are executed.
 }
 else {
   // if <expression> is false, then statements in this block will be executed.
}
```

The if-else if Construct

- You can handle multiple blocks of code, and only one of those blocks will be executed at most.



```
if( <expression1> ) {
    // if <expression1> returns true,  statements in this block are executed.
}
else if ( <expression2>) {
   // if <expression1> is false and <expression2>  is true,
      then statements in this block will be executed.
}
else if (<expression3>) {
     // if <expression1> is false, and <expression2> is false, and <expression3> is
        true, then statements in this block    will be executed.
}
```

The if-else if-else Construct

- enables you to handle multiple blocks of code and ensure that one of them will certainly be executed

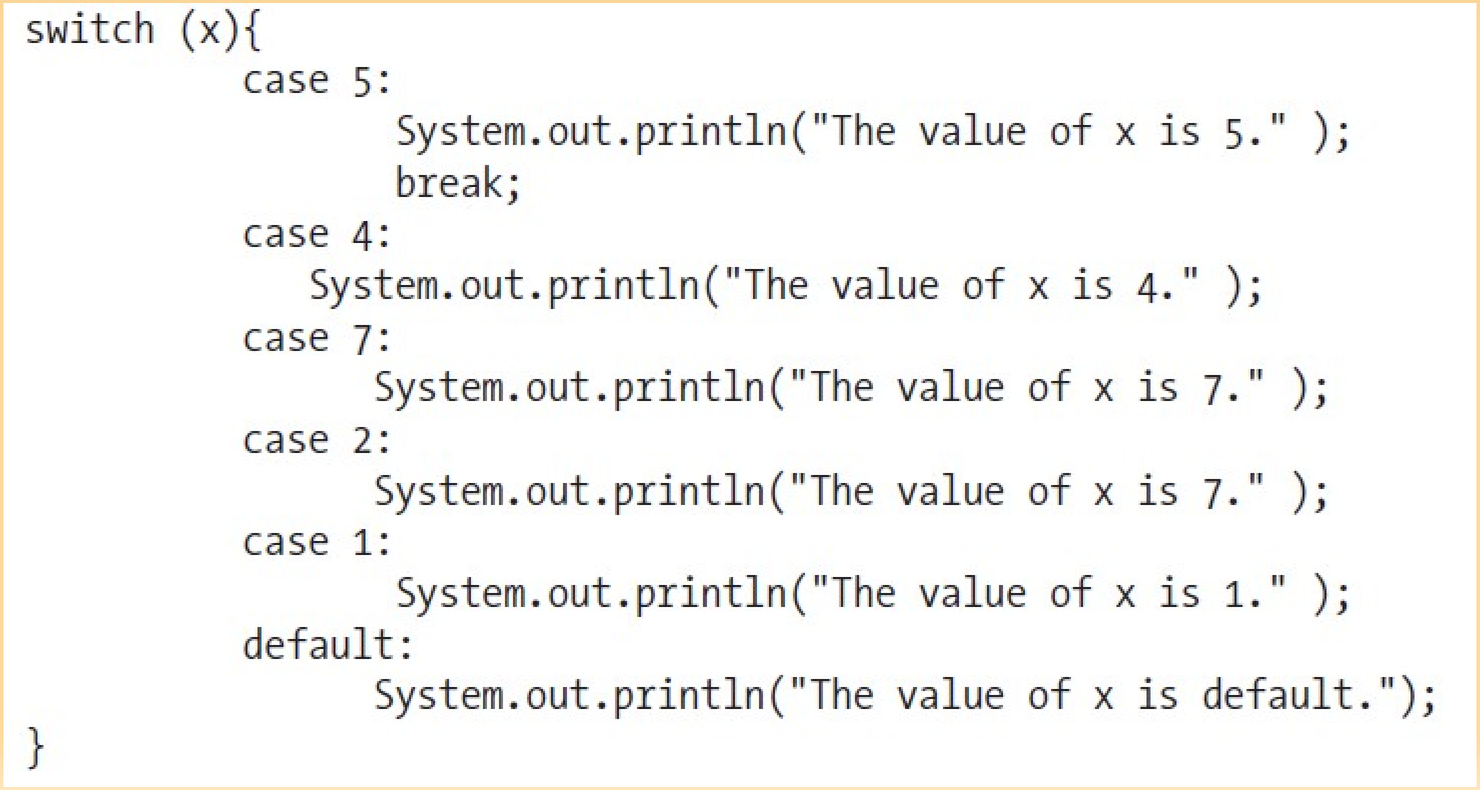
```
if( <expression1> ) {
  // if <expression1> returns true, statements in this block are executed.
 }
 else if (<expression2>) {
  // if <expression1> is false and <expression2>  is true,
     then statements in this block will be executed.
}
else if (<expression3>) {
  // if <expression1> is false and <expression2>  is false, and
    <expression3> is true, then statements in this block
     will be executed.
}
else {
 // if the expression in the if statement and the expressions
     in all the else if statements were false, then the statements
     in this block will be executed.
}
```

Summary of the `if` Constructs

- a single expression: `if` where it is possible that no block will be executed, and `if-else` where one block will certainly be executed.
- multiple expressions: `if-else if` where it is possible that no block will be executed, and `if-else if-else` where one block will certainly be executed.

The switch Statement

- used to make the choices for multiple blocks with the possibility of executing more than one of them.



```
switch (x){
          case 5:
                 System.out.println("The value of x is 5." );
                 break;
          case 4:
             System.out.println("The value of x is 4." );
          case 7:
                System.out.println("The value of x is 7." );
          case 2:
                System.out.println("The value of x is 7." );
          case 1:
                 System.out.println("The value of x is 1." );
          default:
                System.out.println("The value of x is default.");
}
```

Rules

- The comparison of values following the `case` labels with the value of the argument of `switch` determines the execution path.
- Once the execution path of a particular case is chosen, the execution falls through until it runs into a `break` statement.

Notes

- The argument of `switch()` must be one of the following types: `byte, short, char, int, or enum`.
- The argument of `case` must be a literal integral type number or a literal number expression.
- There should be no duplicate case labels

The `default` block

- The `default` does not have to be at the end of the switch.
- When the execution control faces a `default` block, it executes it.
- If there is no `break` statement in the `default` block, there will be fall through just like in any other block.

Iteration Statements

- a block of statements executed over and over again as long as a certain condition is `true`
- four iteration constructs:
    - `while`
    - `do-while`
    - `for`
    - `for-each`

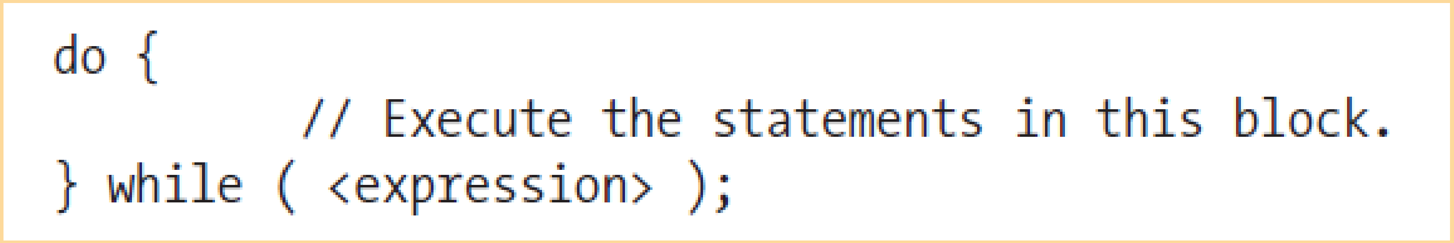
The while Loop Construct

- A block is executed for the first time only when a condition is true.
- After execution the condition is checked again, and as long as the condition stays `true`, the block is executed repeatedly.
- The code block in the `while` loop may not be executed at all

```
while ( <expression> ) {
// if the <expression> is true, execute the statements in this block.
// After the execution,  go back to check the condition again.
}
```

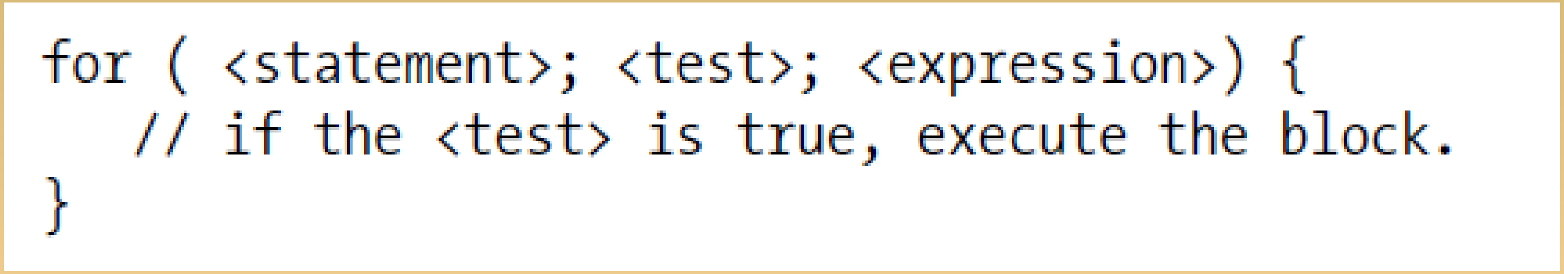
The do-while Loop Construct

- A block is executed once even before checking the condition
- The do-while loop will be executed at least once



```
do {
            // Execute the statements in this block.
} while ( <expression> );
```

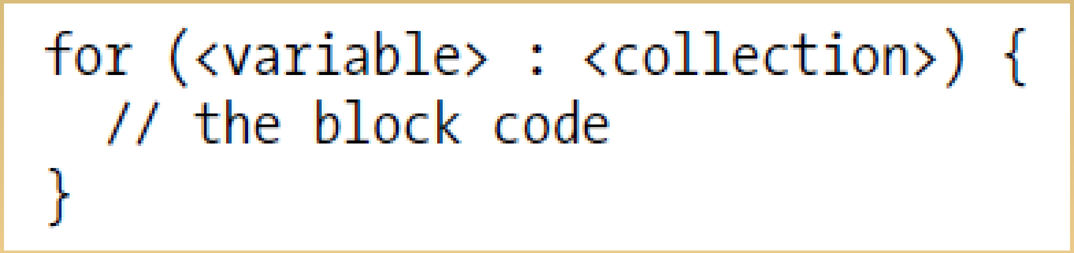
The for Loop Construct



```
for ( <statement>; <test>; <expression>) {
   // if the <test> is true, execute the block.
}
```

- <statement>: initialize the iteration variable , executed only once.
- <test>: A boolean condition. The for block is executed repeatedly until the <test> returns `false`.
- <expression>: Executed immediately after the execution of the for block.

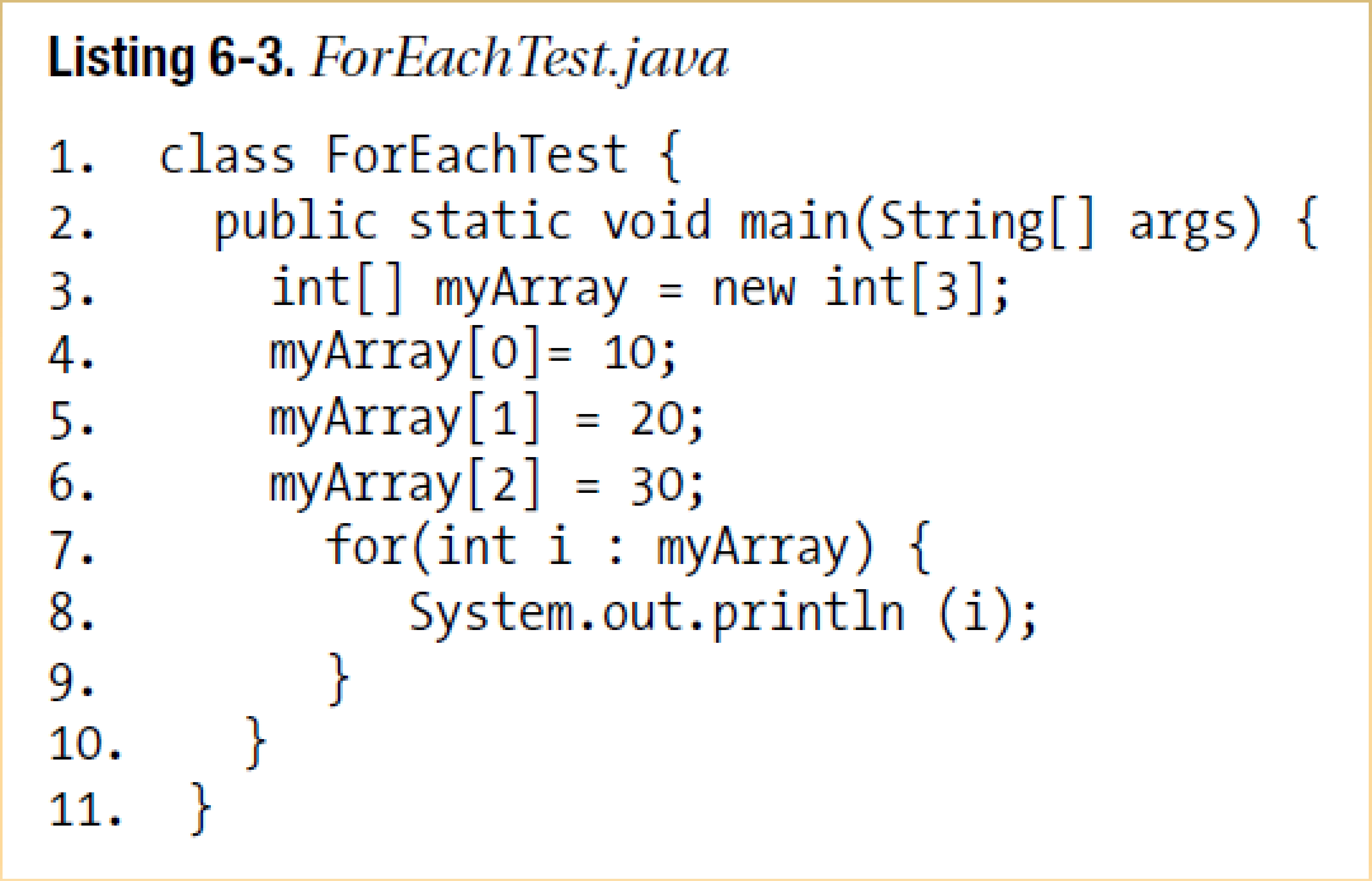
The for-each Loop Construct



```
for (<variable> : <collection>) {
  // the block code
}
```

- It sets the <variable> to the first element of the collection during the first iteration, to the second element during the second iteration, and so on.
- Iterations are performed automatically for all the elements of the collection.

for-each Example



Listing 6-3. *ForEachTest.java*

```java
class ForEachTest {
  public static void main(String[] args) {
    int[] myArray = new int[3];
    myArray[0]= 10;
    myArray[1] = 20;
    myArray[2] = 30;
      for(int i : myArray) {
        System.out.println (i);
      }
   }
 }
```

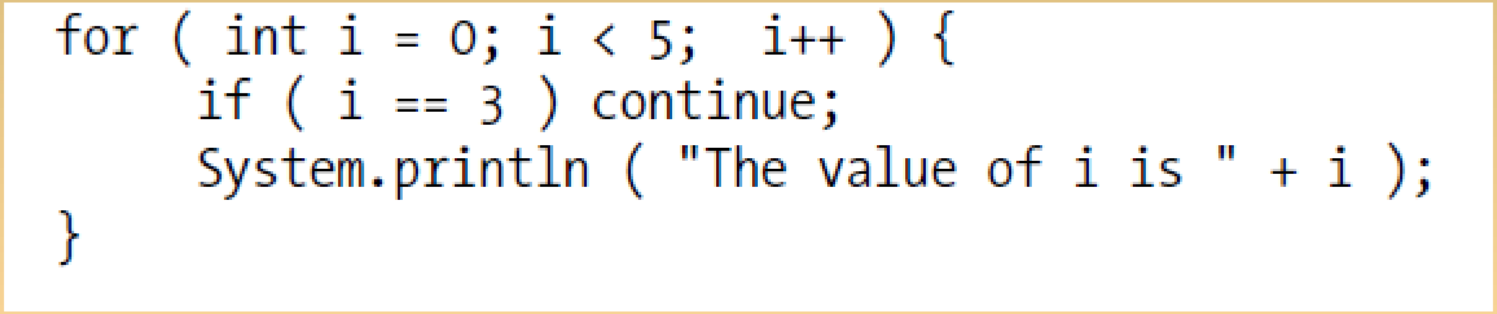
Block Breaker Statements

- to quit either the current iteration of a loop or the entire loop altogether:
  - The continue Statement
  - The break Statement

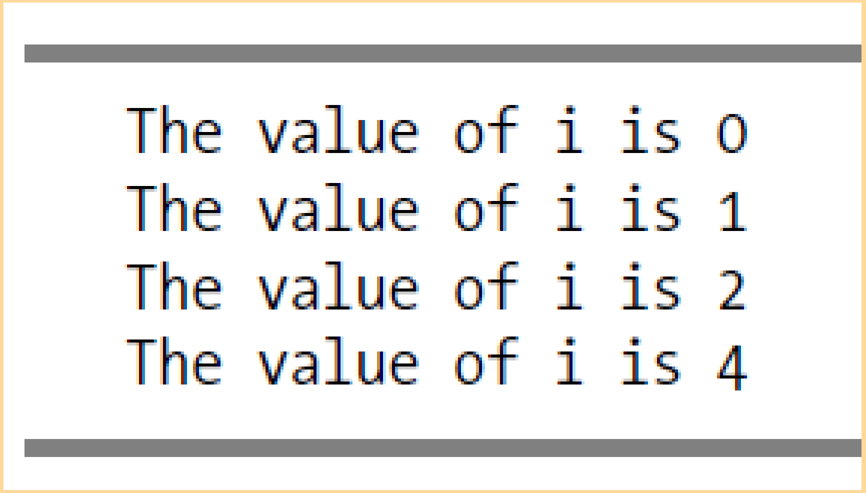
The `continue` Statement

- When this statement is executed, the current iteration is terminated, and the control jumps to the next iteration:
- `while`, `do-while`: jumps to the boolean condition
- `for`: jumps to the <expression> in the `for` (<statement>; <test>; <expression>) statement.

continue Example



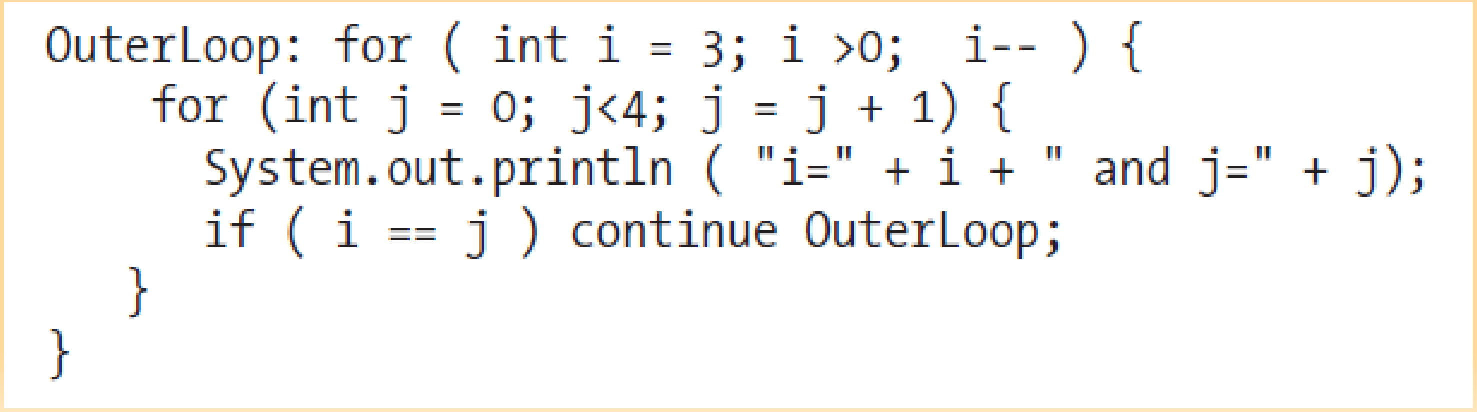
```
for ( int i = 0; i < 5;  i++ ) {
      if ( i == 3 ) continue;
      System.println ( "The value of i is " + i );
}
```



```
The value of i is 0
The value of i is 1
The value of i is 2
The value of i is 4
```

continue in Nested Loops

- need to specify from which loop you need to continue the next iteration: the labeled `continue` statement
- want the execution control to jump from an inner block to an outer block: The beginning of the outer block will be labeled

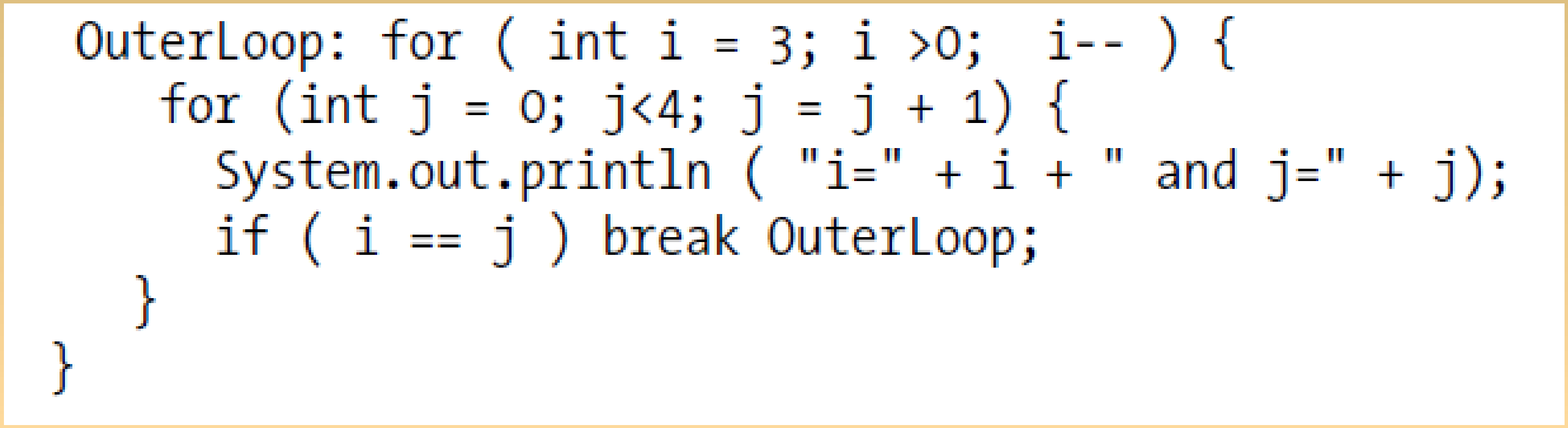


```
OuterLoop: for ( int i = 3; i >0;  i-- ) {
    for (int j = 0; j<4; j = j + 1) {
      System.out.println ( "i=" + i + " and j=" + j);
      if ( i == j ) continue OuterLoop;
    }
}
```

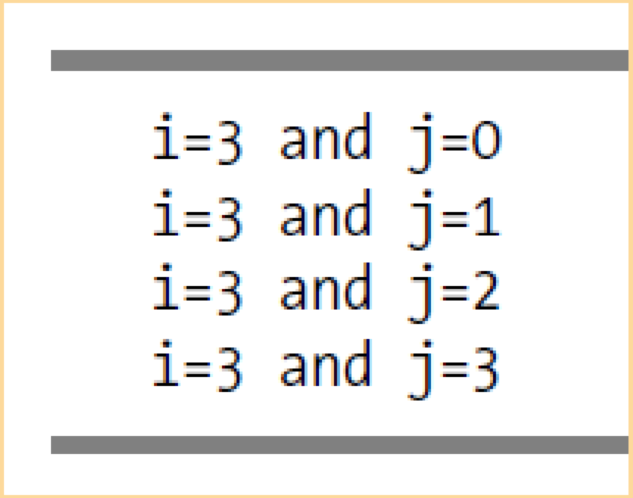
The `break` Statement

- throws the execution control out of the block altogether
- used either in a loop or in a `switch` block
- In case of nested loops, you might need to tell from which loop you want to break: the labeled `break` statement

break Example



```
OuterLoop: for ( int i = 3; i >0;  i-- ) {
    for (int j = 0; j<4; j = j + 1) {
      System.out.println ( "i=" + i + " and j=" + j);
      if ( i == j ) break OuterLoop;
   }
}
```



```
i=3 and j=0
i=3 and j=1
i=3 and j=2
i=3 and j=3
```