# Session 04: Loops and Control Statements

Dr Tomasz Kurowski
t.j.kurowski@cranfield.ac.uk

## Module engagement QR code

If you are unable to scan this code, please contact SAS Admin –

**seeaadmin@cranfield.ac.uk**

- Selection Statements
    - Using `if` and `if...else`
    - Nested `if` Statements
    - Using `switch` Statements
    - Conditional Operator

- Repetition Statements
    - Looping: `while`, `do`, and `for`
    - Nested loops
    - Using `break` and `continue`

## Selection Statements

- `if` Statements
- `switch` Statements
- Conditional Operators

## `if` Statements

```
if (booleanExpression) {
  statement(s);
}
```

Example:

```
if ((i > 0) && (i < 10)) {
  System.out.println("i is an " +
    "integer between 0 and 10");
}
```

## if - Caution

Adding a semicolon at the end of an if clause is a common mistake.

```
if (radius >= 0);   ←——— Wrong
{
  area = radius*radius*PI;
  System.out.println(
    "The area for the circle of radius " +
    radius + " is " + area);
}
```

This mistake is hard to find, because it is not a compilation error or a runtime error, it is a logic error.

This error often occurs when you use the next-line block style.

## The `if...else` Statement

```
if (booleanExpression) {
  statement(s)-for-the-true-case;
}
else {
  statement(s)-for-the-false-case;
}
```

## `if...else` Example

```
if (radius >= 0) {
  area = radius*radius*PI;
    System.out.println("The area for the “
    + “circle of radius " + radius +
    " is " + area);
}
else {
  System.out.println("Negative input");
}
```

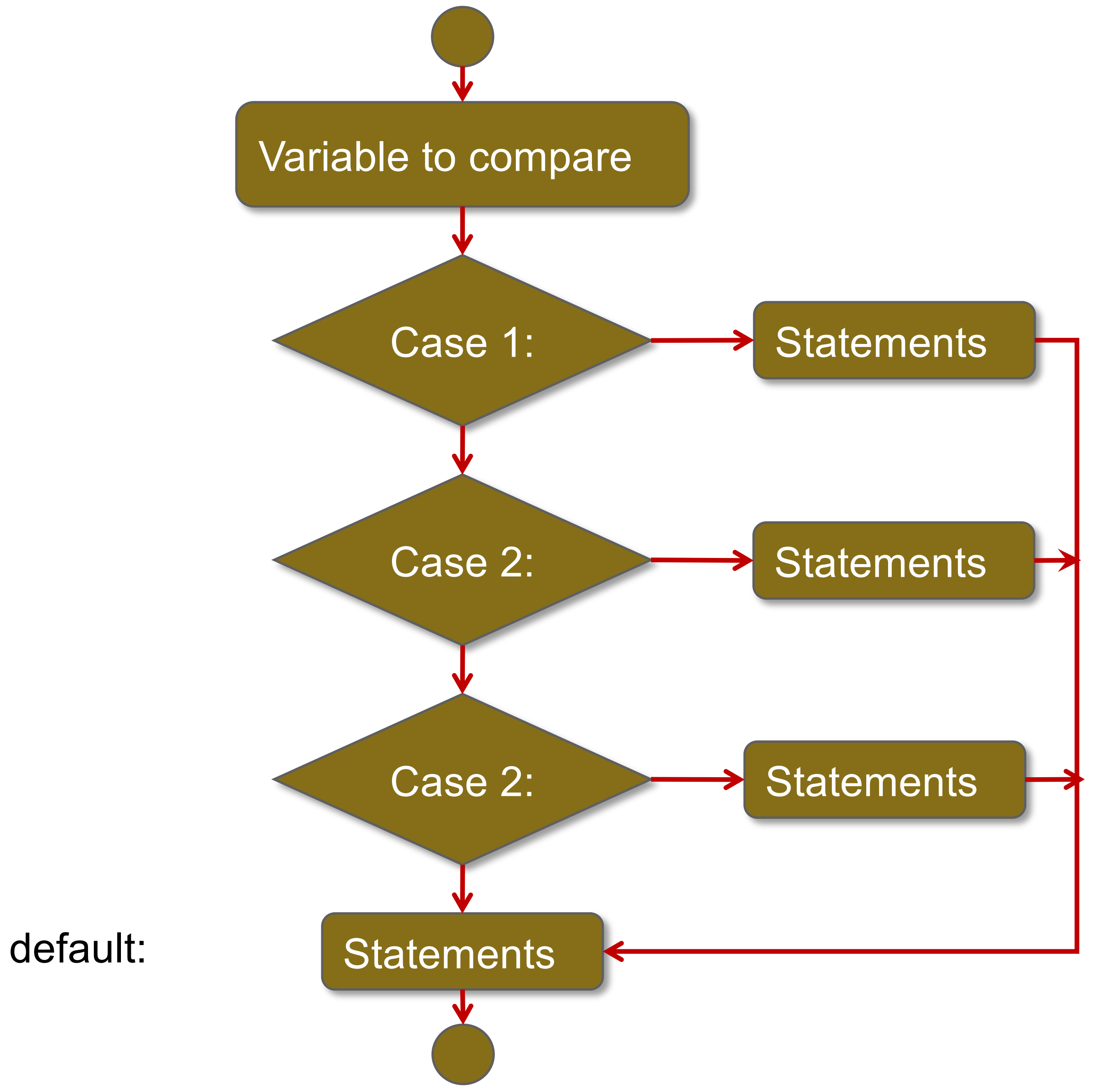
## Multiple Alternative if Statements

```
if (score >= 90) {
           grade = 'A';
       } else if (score >= 80) {
           grade = 'B';
       } else if (score >= 70) {
           grade = 'C';
       } else if (score >= 60) {
           grade = 'D';
       } else {
           grade = 'F';
       }
```

## `switch` Statements

```
switch (year) {
  case 7:  annualInterestRate = 7.25;
           break;
  case 15: annualInterestRate = 8.50;
           break;
  case 30: annualInterestRate = 9.0;
           break;
  default: System.out.println(
      "Wrong number, enter 7, 15, or 30");
}
```

Another way of writing multi-way ‘if’ statements


Variable to compare
Case 1:
Statements
Case 2:
Statements
Case 2:
Statements
default:
Statements

## `switch` Statement Rules

```
switch (year) {
  case 7:  annualInterestRate = 7.25;
           break;
  case 15: annualInterestRate = 8.50;
           break;
  case 30: annualInterestRate = 9.0;
           break;
  default: System.out.println(
      "Wrong number, enter 7, 15, or 30");
}
```

1. The switch-expression must yield a value of char, byte, short, or int type and must always be enclosed in parentheses.

2. The Value that we want to compare must have the same data type as the value of the switch-expression.

**The resulting statements in the case statement are executed when the value in the case statement matches the value of the switch-expression. (The case statements are executed in sequential order.)

## `switch` Statement Rules (cont.)

```
switch (year) {
  case 7:  annualInterestRate = 7.25;
           break;
  case 15: annualInterestRate = 8.50;
           break;
  case 30: annualInterestRate = 9.0;
           break;
  default: System.out.println(
      "Wrong number, enter 7, 15, or 30");
}
```

3. The keyword break is optional, but it should be used at the end of each case in order to terminate the remainder of the switch statement. If the break statement is not present, the next case statement will be checked.

## `switch` Statement Rules, cont.

```
switch (year) {
  case 7:  annualInterestRate = 7.25;
           break;
  case 15: annualInterestRate = 8.50;
           break;
  case 30: annualInterestRate = 9.0;
           break;
  default: System.out.println(
      "Wrong number, enter 7, 15, or 30");
}
```

4. The default case, which is optional, can be used to perform actions when none of the specified cases is true.

5. The order of the cases (including the default case) does not matter. However, it is a good programming style to follow the logical sequence of the cases and place the default case at the end.

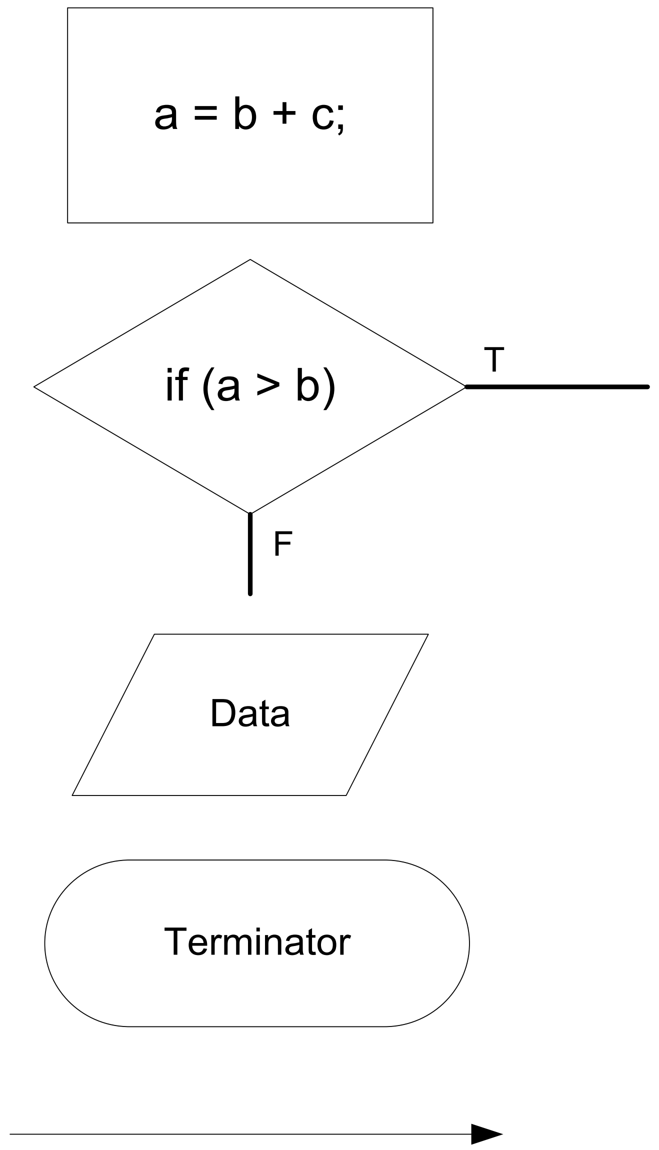
## switch - Caution

Do not forget to use a break statement when one is needed. For example, the following code always displays Wrong number of years regardless of what numOfYears is.

Suppose the numOfYears is 15. The statement annualInterestRate = 8.50 is executed, then the statement annualInterestRate = 9.0, and finally the statement System.out.println ("Wrong number of years").

```
switch (numOfYears) {
  case 7:  annualInterestRate = 7.25;
  case 15: annualInterestRate = 8.50;
  case 30: annualInterestRate = 9.0;
  default: System.out.println("Wrong number of years");
}
```

## Flowchart symbols



A Process

A Decision

Data

Start / Finish

Flow of control

## Conditional Operator

```
if (x > 0)
    y = 1
else
    y = -1;
```

**is equivalent to**

```
y = (x > 0) ? 1 : -1;
```

**Format of the conditional operator is**

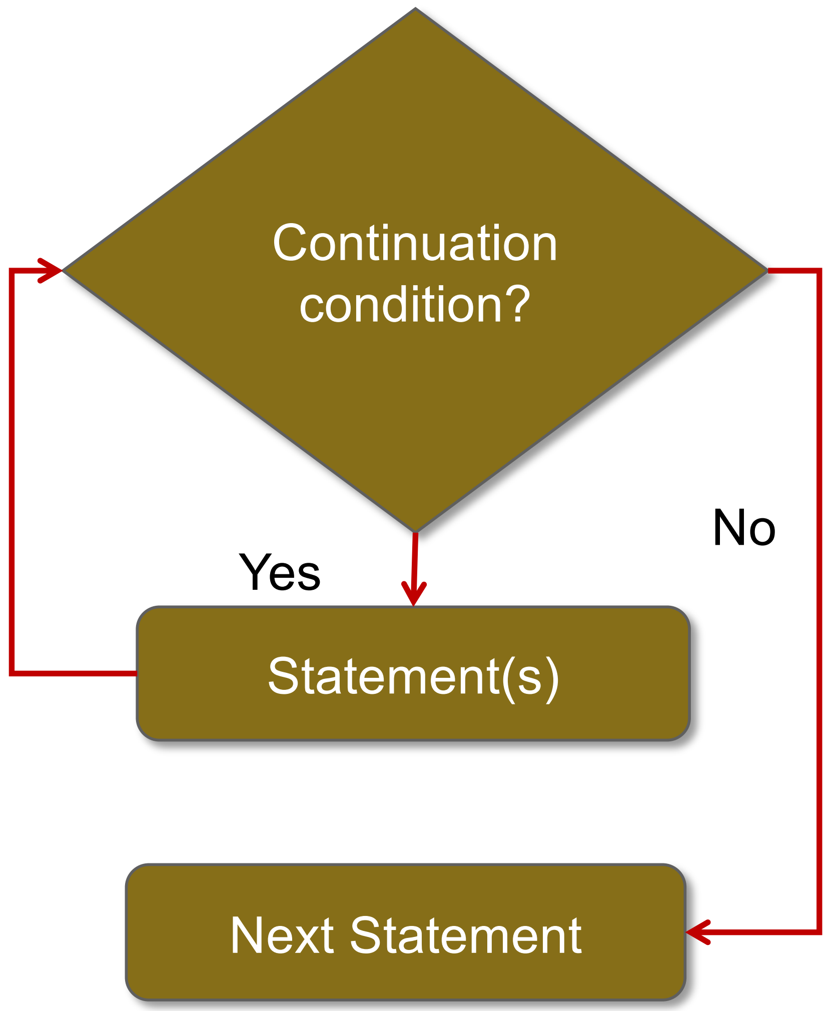
```
(booleanExp) ? exp1 : exp2
```

## Repetitions

- `while` Loops
- `do-while` Loops
- `for` Loops
- `break` and `continue`

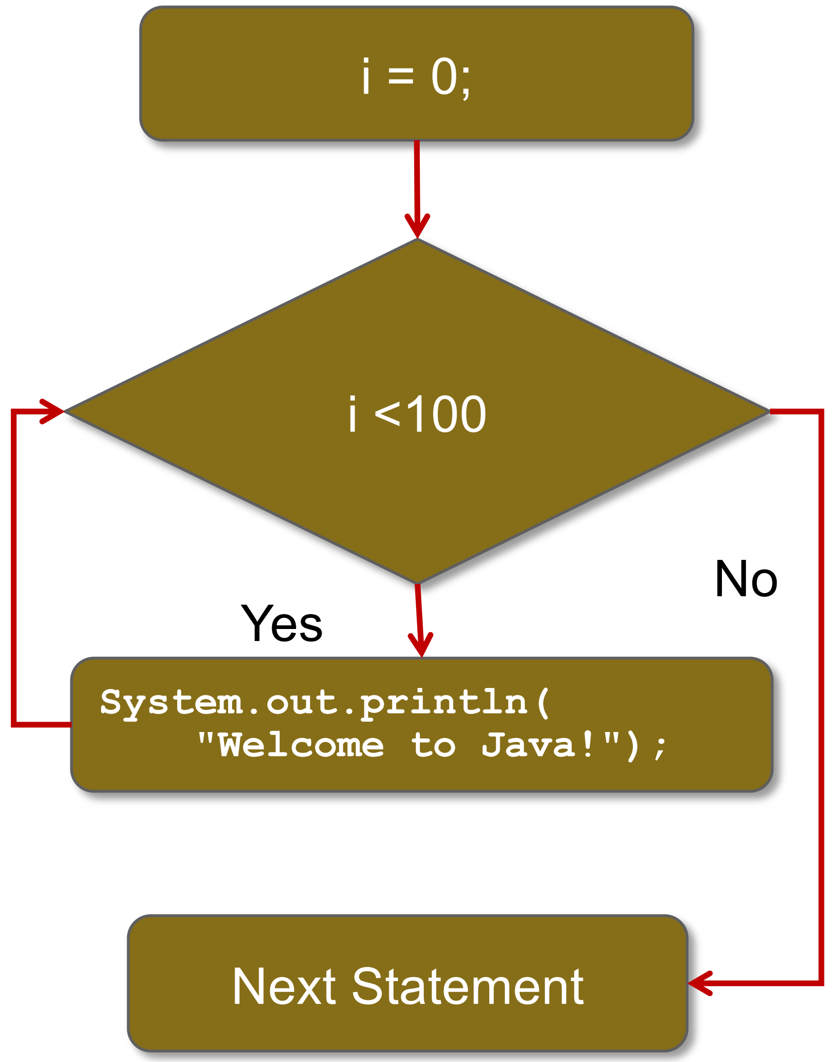
## `while` Loop Flow Chart

```
while (continuation-condition) {
 // loop-body;
}
```



## `while` Loop Flow Chart Example

```
int i = 0;
while (i < 100) {
  System.out.println(
    "Welcome to Java!");
  i++;
}
```



## Using while Loops

This small program accepts a series of integer values from the user, via an Input Dialog Box.

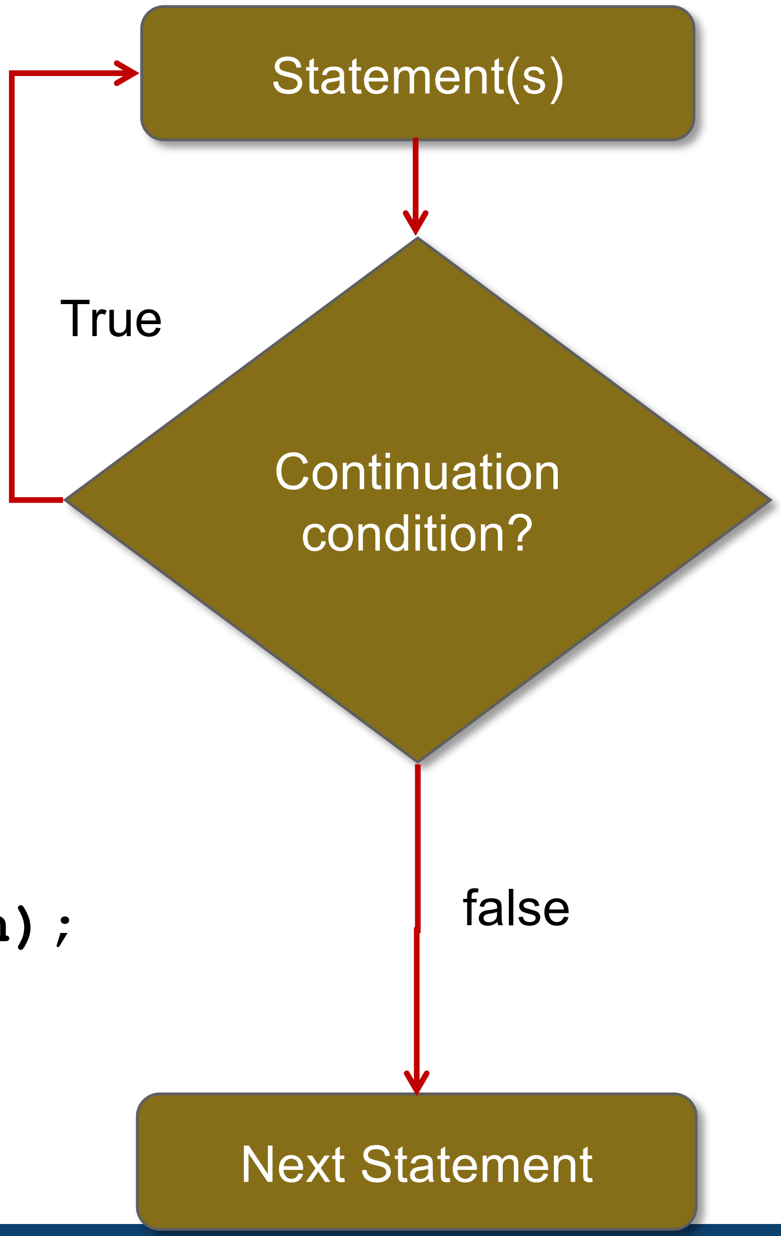
The values are summed until the user enters a value of 0, whereupon the program display the current total and finishes.

(Netbeans TestWhile.java)

TestWhile

Run

## `do-while` Loop



```
do {
  // Loop body;
} while (continue-condition);
```

## `for` Loops

```
for (initial-action; loop-continuation-condition; action-
 after-each-iteration) {
   //loop body;
}
```

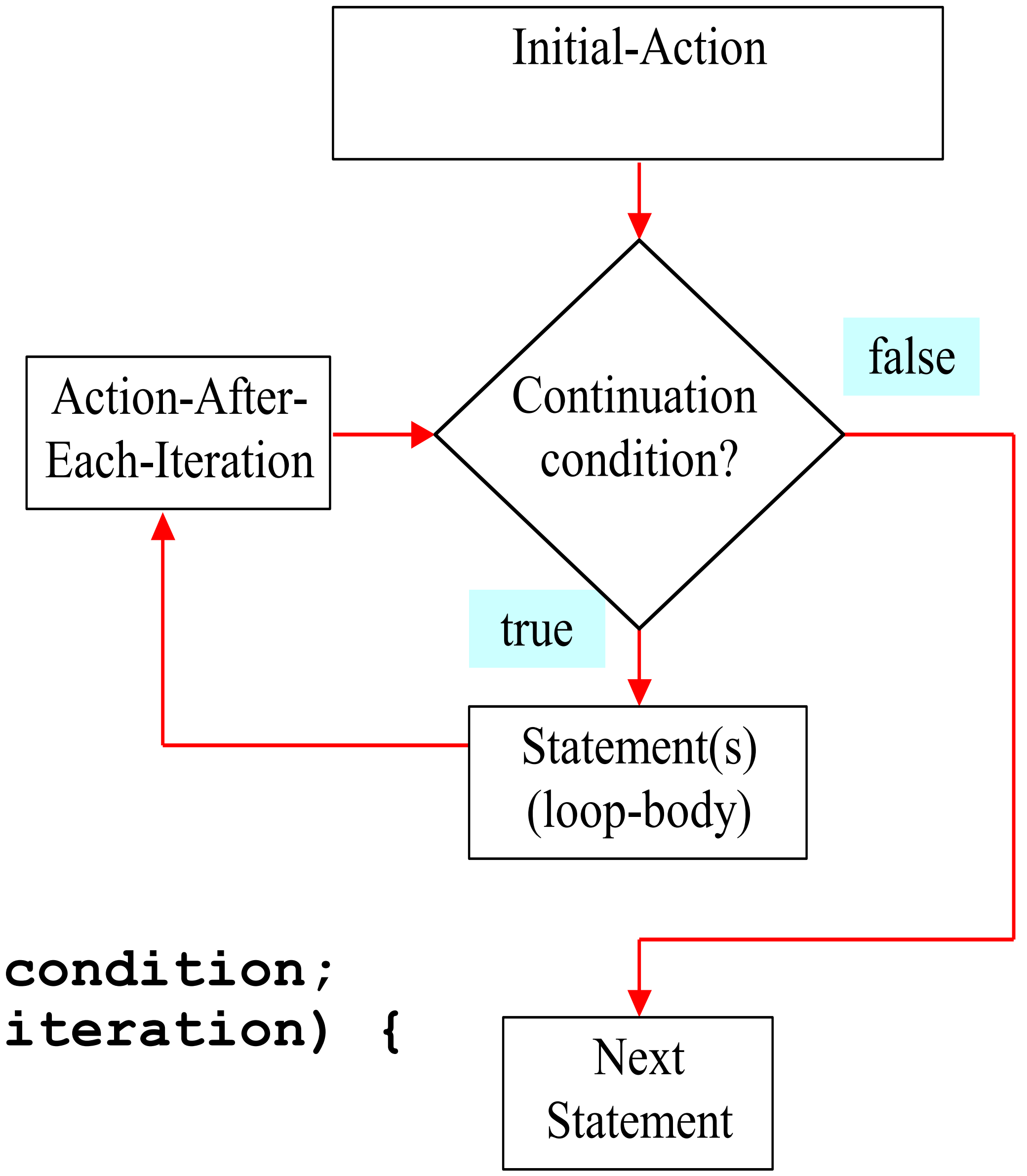
**Example:**

```
int i;
for (i = 0; i<100; i++) {
  System.out.println("Welcome to Java! ” + i);
}
```

**Equivalent to :**

```
int i = 0;
while (i < 100) {
  System.out.println("Welcome to Java! ” + i);
  i++;
}
```

## `for` Loop Flow Chart



```
for (int i=0; i<5; i++) {
    ….statements
}
…next statement
```

```
for (initial-action;
  loop-continuation-condition;
  action-after-each-iteration) {
   //loop body;
}
```

## `for` Loop Examples

Examples for using the `for` loop:

Example 3.3: Using for Loops

[TestSum] [Run]

## for loop - Caution

Adding a semicolon at the end of the for clause before the loop body is a common mistake, as shown below:

```
for (int i=0; i<10; i++);
{
  System.out.println("i is " + i);
}
```

$\leftarrow$ Wrong

## loops - caution, cont.

Similarly, the following loop is also wrong:

```
int i=0;
while (i<10);       ← Wrong
{
  System.out.println("i is " + i);
  i++;
}
```

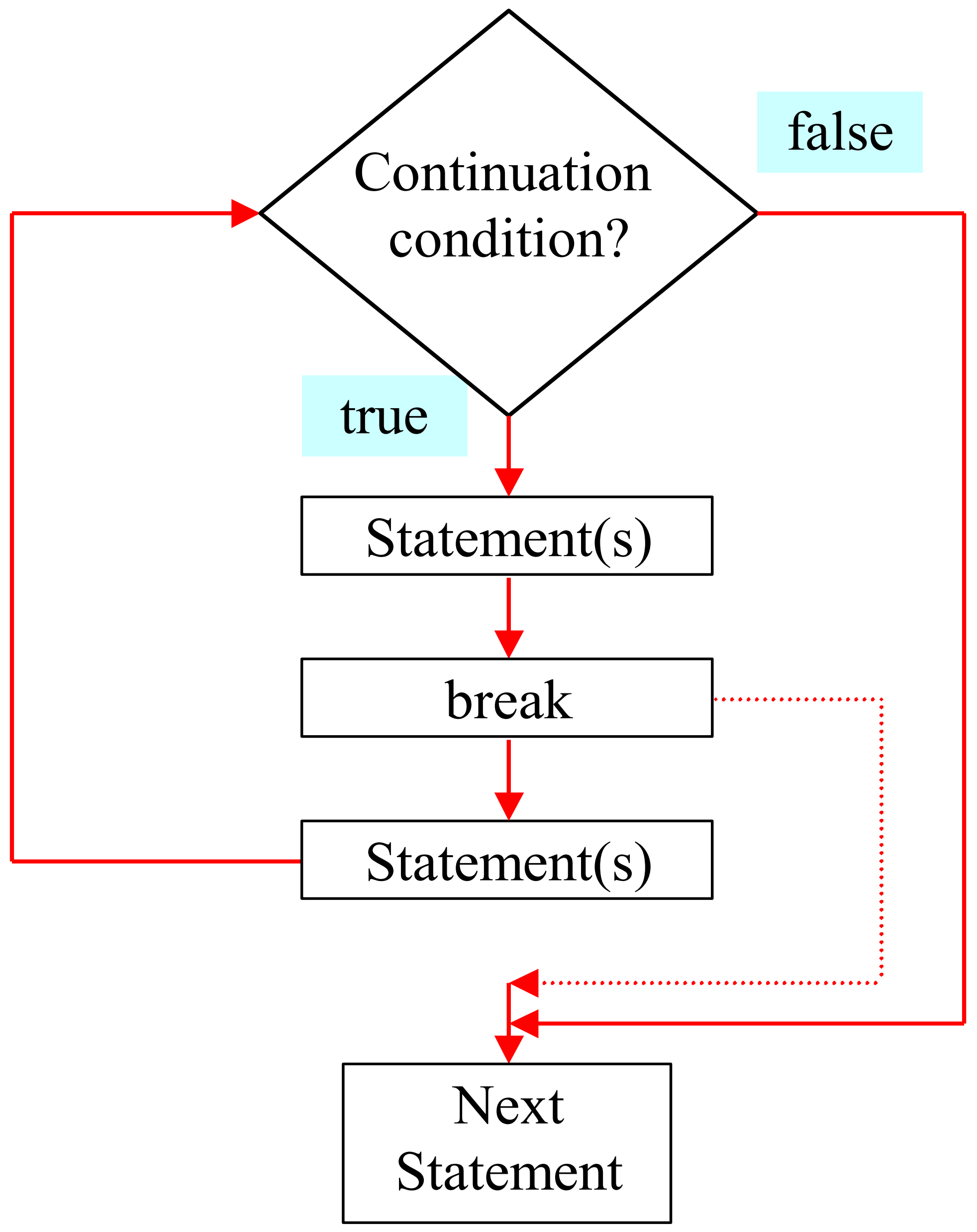
In the case of the do loop, the following semicolon is needed to end the loop.

```
int i=0;
do {
  System.out.println("i is " + i);
  i++;
} while (i<10);       ← Correct
```

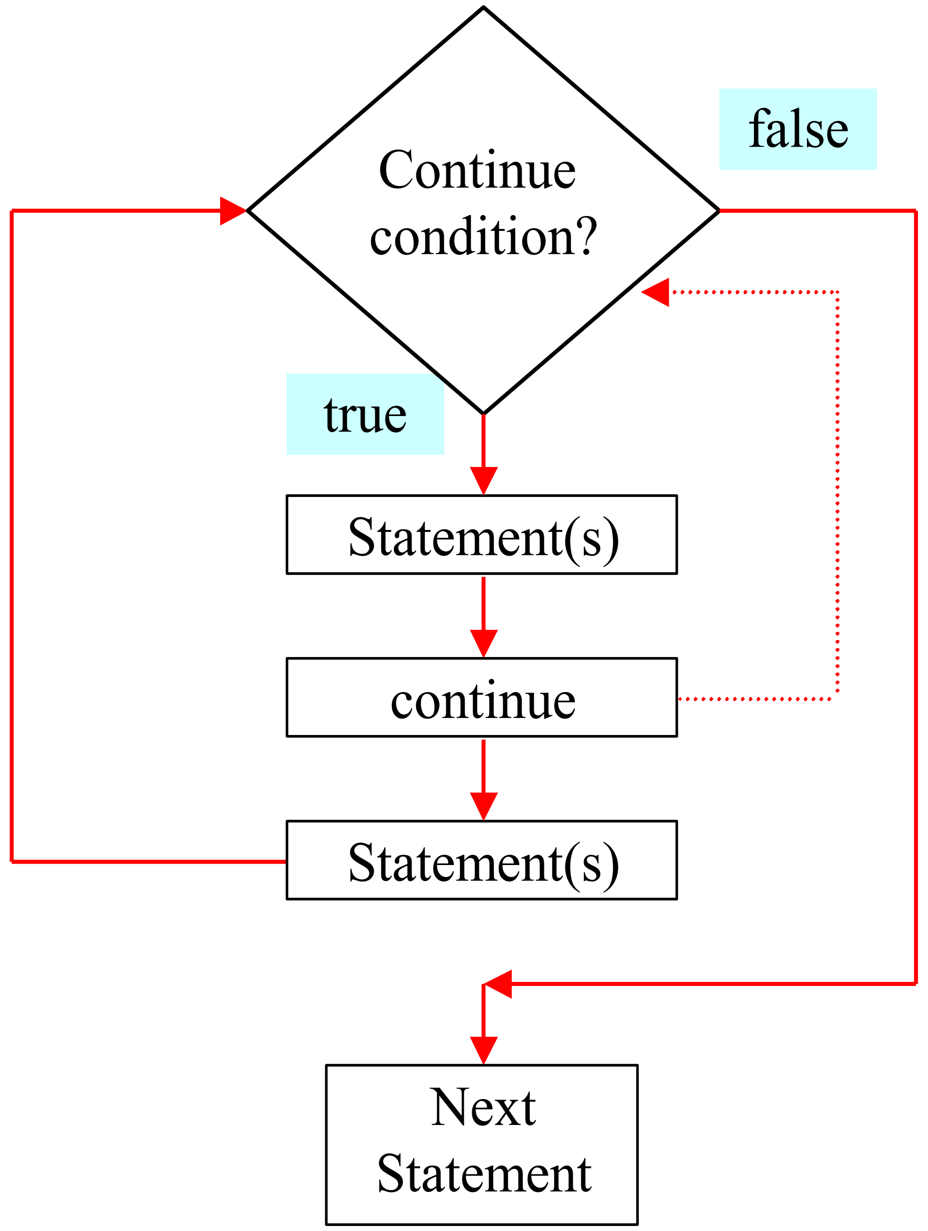
## Which Loop to Use?

- The three forms of loop statements, while, do, and for, are expressively equivalent; that is, you can write a loop in any of these three forms.
- In general, a for loop is used if the number of repetitions is known, as, for example, when you need to print a message 100 times.
- A while loop may be used if the number of repetitions is not known, as in the case of reading the numbers until the input is 0. It may be executed 0 times if the initial condition is false.
- A do-while loop is very similar to the while loop and can be used to replace a while loop if the loop body has to be executed at least once before testing the continuation condition.

## The `break` Keyword



## The `continue` Keyword



## break Example

```
public class TestBreak {
  /** Main method */
  public static void main(String[] args) {
       int sum = 0;
       int item = 0;
       while (item < 5)   {
            item ++;
                    sum += item;
                    if (sum >= 6) break;      // break out from loop completely
            }
            System.out.println("The sum is " + sum);
      }
}
```

## continue Example

```
public class TestContinue {
  /** Main method */
  public static void main(String[] args) {
        int sum = 0;
        int item = 0;
        while (item < 5) {
            item++;
            if (item == 2) continue;  // Skip rest of this iteration
                 sum += item;
             }
             System.out.println("The sum is " + sum);
       }
}
```