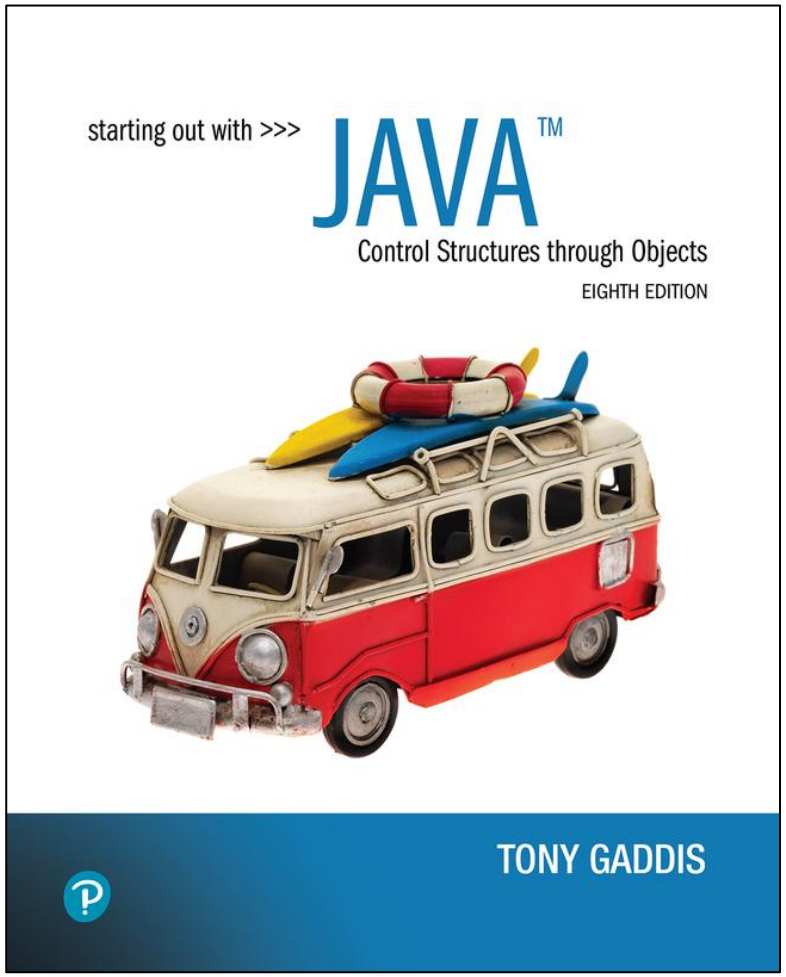
Starting Out with Java Control Structures Through Objects

Eighth Edition



Chapter 4

Loops and Files

Pearson

Chapter Topics (1 of 2)

- Chapter 4 discusses the following main topics:
  - The Increment and Decrement Operators
  - The `while` Loop
  - Using the `while` Loop for Input Validation
  - The `do-while` Loop
  - The `for` Loop
  - Running Totals and Sentinel Values

Chapter Topics (2 of 2)

- Chapter 4 discusses the following main topics:
  - Nested Loops
  - The `break` and `continue` Statements
  - Deciding Which Loop to Use
  - Introduction to File Input and Output
  - Generating Random Numbers with the `Random` class

The Increment and Decrement Operators

- There are numerous times where a variable must simply be incremented or decremented.

```
number = number + 1;
number = number - 1;
```

- Java provide shortened ways to increment and decrement a variable's value.
- Using the **++** or **--** unary operators, this task can be completed quickly.

```
number++;   or   ++number;
number--;   or   --number;
```

- Example: IncrementDecrement.java

Differences Between Prefix and Postfix

- When an increment or decrement are the only operations in a statement, there is no difference between prefix and postfix notation.
- When used in an expression:
  - prefix notation indicates that the variable will be incremented or decremented prior to the rest of the equation being evaluated.
  - postfix notation indicates that the variable will be incremented or decremented after the rest of the equation has been evaluated.
- Example: Prefix.java

The `while` Loop (1 of 2)

- Java provides three different looping structures.
- The `while` loop has the form:

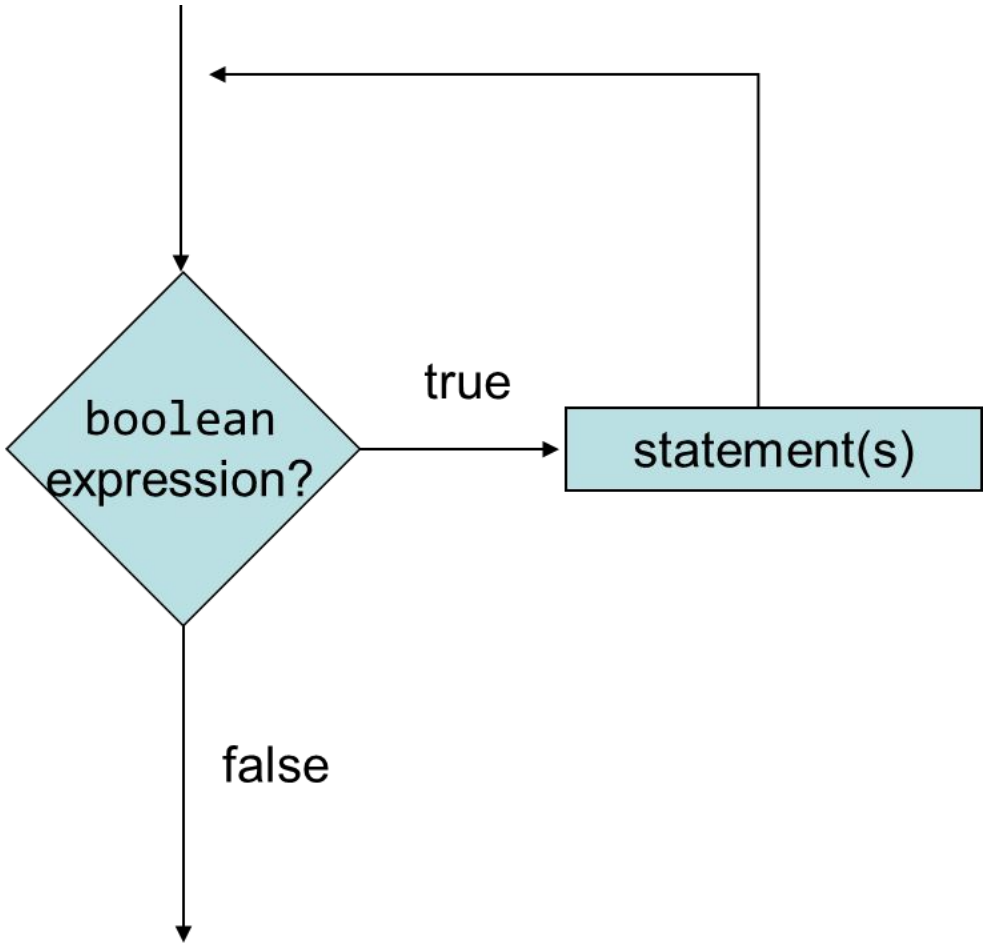
```
while(condition)
{
   statements;
}
```

- While the condition is true, the statements will execute repeatedly.
- The `while` loop is a **pretest** loop, which means that it will test the value of the condition prior to executing the loop.

The `while` Loop (2 of 2)

- Care must be taken to set the condition to false somewhere in the loop so the loop will end.
- Loops that do not end are called **infinite loops**.
- A `while` loop executes 0 or more times. If the condition is false, the loop will not execute.
- Example: WhileLoop.java

The `while` Loop Flowchart



Infinite Loops (1 of 2)

- In order for a `while` loop to end, the condition must become false. The following loop will not end:

```
int x = 20;
while(x > 0)
{
    System.out.println("x is greater than 0");
}
```

- The variable `x` never gets decremented so it will always be greater than 0.
- Adding the **`x--`** above fixes the problem.

Infinite Loops (2 of 2)

- This version of the loop decrements `x` during each iteration:

```
int x = 20;
while(x > 0)
{
   System.out.println("x is greater than 0");
   x--;
}
```

Block Statements in Loops

- Curly braces are required to enclose block statement while loops. (like block `if` statements)

```
while (condition)
{
    statement;
    statement;
    statement;
}
```

The `while` Loop for Input Validation

- **Input validation** is the process of ensuring that user input is valid.

```
System.out.print("Enter a number in the " +
                 "range of 1 through 100: ");
number = keyboard.nextInt();
// Validate the input.
while (number < 1 || number > 100)
{
  System.out.println("That number is invalid.");
  System.out.print("Enter a number in the " +
                   "range of 1 through 100: ");
  number = keyboard.nextInt();
}
```

- Example: SoccerTeams.java

The `do-while` Loop

- The `do-while` loop is a **post-test** loop, which means it will execute the loop prior to testing the condition.
- The `do-while` loop (sometimes called called a `do` loop) takes the form:

```
do
{
  statement(s);
} while (condition);
```

- Example: TestAverage1.java

The `do-while` Loop Flowchart