Chapter 4: Selection Control Structures

- Relational and logical operators.
- Logical (Boolean) expressions.
- The selection control structures `if`, `if...else`, and `switch`.

Control Structures

- Three methods of processing a program:
  - In sequence
  - Branching
  - Looping
- Branch: Altering the flow of program execution by making a selection or choice.
- Loop: Altering the flow of program execution by repeating statements.
Control Structures

![Flow of execution diagram](image)

**Figure 4-1** Flow of execution

Relational Operators

- Relational operator:
  - Allows you to make comparisons in a program.
  - Binary operator.
- Condition is represented by a logical expression in Java.
- Logical expression: An expression that has a value of either true or false.
Relational Operators

Table 4-1  Relational Operators in Java

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>equal to</td>
</tr>
<tr>
<td>!=</td>
<td>not equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>=&gt;</td>
<td>greater than or equal to</td>
</tr>
</tbody>
</table>

Relational Operators and Primitive Data Types

- Can be used with integral and floating-point data types.
- Can be used with the char data type.
- Unicode Collating Sequence.
Relational Operators and Primitive Data Types

Comparing Strings

- class String
- Method `compareTo`
- Method `equals`

Given string `str1` and `str2`

\[
\begin{align*}
\text{str1.compareTo(str2)} &= \begin{cases} 
\text{an integer} < 0 & \text{if string str1 < str2} \\
0 & \text{if string str1 equals string str2} \\
\text{an integer} > 0 & \text{if string str1 > str2}
\end{cases}
\end{align*}
\]
Comparing Strings

```java
String str1 = "Hello";
String str2 = "Hi";
String str3 = "Air";
String str4 = "Bill";
String str5 = "Bigger";
```

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>str1.compareTo(str2)</code></td>
<td>&lt; 0</td>
<td><code>str1 = &quot;Hello&quot;</code> and <code>str2 = &quot;Hi&quot;</code>. The first character of <code>str1</code> and <code>str2</code> are the same, but the second character 'e' of <code>str1</code> is less than the second character 'i' of <code>str2</code>. Therefore, <code>str1.compareTo(str2) &lt; 0</code></td>
</tr>
<tr>
<td><code>str1.compareTo(&quot;Hen&quot;)</code></td>
<td>&lt; 0</td>
<td><code>str1 = &quot;Hello&quot;</code>. The first two characters of <code>str1</code> and &quot;Hen&quot; are the same, but the third character '1' of <code>str1</code> is less than the third character 'n' of &quot; Hen&quot;. Therefore, <code>str1.compareTo(&quot;Hen&quot;) &lt; 0</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>str4.compareTo(str3)</code></td>
<td>&gt; 0</td>
<td><code>str4 = &quot;Bill&quot;</code> and <code>str3 = &quot;Air&quot;</code>. The first character 'B' of <code>str4</code> is greater than the first character 'A' of <code>str3</code>. Therefore, <code>str4.compareTo(str3) &gt; 0</code></td>
</tr>
<tr>
<td><code>str1.compareTo(&quot;hello&quot;)</code></td>
<td>&lt; 0</td>
<td><code>str1 = &quot;Hello&quot;</code>. The first character 'H' of <code>str1</code> is less than the first character 'h' of &quot;hello&quot; because the Unicode value of 'H' is 72, and the Unicode value of 'h' is 104. Therefore, <code>str1.compareTo(&quot;hello&quot;) &lt; 0</code></td>
</tr>
<tr>
<td><code>str2.compareTo(&quot;Hi&quot;)</code></td>
<td>= 0</td>
<td><code>str2 = &quot;Hi&quot;</code>. The strings <code>str2</code> and &quot;Hi&quot; are of the same length and their corresponding characters are the same. Therefore, <code>str2.compareTo(&quot;Hi&quot;) = 0</code></td>
</tr>
<tr>
<td><code>str4.compareTo(&quot;Billy&quot;)</code></td>
<td>&lt; 0</td>
<td><code>str4 = &quot;Bill&quot;</code> has four characters and &quot;Billy&quot; has five characters. Therefore, <code>str4</code> is the shorter string. All four characters of <code>str4</code> are the same as the corresponding first four characters of &quot;Billy&quot; and &quot;Billy&quot; is the larger string. Therefore, <code>str4.compareTo(&quot;Billy&quot;) &lt; 0</code></td>
</tr>
<tr>
<td><code>str5.compareTo(&quot;Big&quot;)</code></td>
<td>&gt; 0</td>
<td><code>str5 = &quot;Bigger&quot;</code> has six characters and &quot;Big&quot; has three characters. Therefore, <code>str5</code> is the larger string. The first three characters of <code>str5</code> are the same as the corresponding first three characters of &quot;Big&quot;. Therefore, <code>str5.compareTo(&quot;Big&quot;) &gt; 0</code></td>
</tr>
</tbody>
</table>
Comparing Strings

Table 4-4 Logical (Boolean) Operators in Java

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>not</td>
</tr>
<tr>
<td>&amp; &amp;</td>
<td>and</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4-5 ! (not) Operator

<table>
<thead>
<tr>
<th>Expression</th>
<th>!(Expression)</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Table 4-6 & & (and) Operator

<table>
<thead>
<tr>
<th>Expression1</th>
<th>Expression2</th>
<th>Expression1 &amp; &amp; Expression2</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

Table 4-7 | (or) Operator

| Expression1 | Expression2 | Expression1 | | Expression2 |
|-------------|-------------|-------------|-----------------------------|
| true        | true        | true        | true                        |
| true        | false       | true        | false                       |
| false       | true        | true        | false                       |
| false       | false       | false        | false                       |
Comparing Strings

Table 4-8  Precedence of Operators

<table>
<thead>
<tr>
<th>Operators</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>!, +, − (unary operators)</td>
<td>first</td>
</tr>
<tr>
<td>*, /, %</td>
<td>second</td>
</tr>
<tr>
<td>+, −</td>
<td>third</td>
</tr>
<tr>
<td>&lt;, &lt;=, &gt;=, &gt;</td>
<td>fourth</td>
</tr>
<tr>
<td>==, !=</td>
<td>fifth</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>sixth</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>= (assignment operator)</td>
<td>last</td>
</tr>
</tbody>
</table>

Short-Circuit Evaluation

>A process in which the computer evaluates a logical expression from left to right and stops as soon as the value of the expression is known.
Selection

- One-way selection
- Two-way selection
- Compound (block of) statements
- Multiple selections (nested if)
- Conditional operator
- `switch` structures

One-Way Selection

- Syntax:
  
  ```java
  if (expression)
  statement
  ```

  Expression referred to as decision maker.
  Statement referred to as action statement.
Example 4-11

// Determine the absolute value of an integer
import javax.swing.JOptionPane;
public class AbsoluteValue
{
    public static void main(String[] args)
    {
        int number;
        int temp;
        String numString;
        numString = JOptionPane.showInputDialog
            ("Enter an integer:"); // Line 1
        number = Integer.parseInt(numString); // Line 2
        temp = number; // Line 3

        if (number < 0) // Line 4
            number = -number; // Line 5

        JOptionPane.showMessageDialog(null,
            "The absolute value of " + temp
            + " is " + number,
            "Absolute Value",
            JOptionPane.INFORMATION_MESSAGE); // Line 6

        System.exit(0);
    }
}
Two-Way Selection

- Syntax:

```java
if (expression)
statement1
else
statement2
```

- `else` statement must be paired with an `if`.

*Figure 4-6*  Two-way selection
Two-Way Selection

Example 4-14

```java
if (hours > 40.0)
    wages = 40.0 * rate +
            1.5 * rate * (hours - 40.0);
else
    wages = hours * rate;
```

Two-Way Selection

Example 4-15

```java
if (hours > 40.0);          //Line 1
    wages = 40.0 * rate +
            1.5 * rate * (hours - 40.0); //Line 2
else                          //Line 3
    wages = hours * rate;       //Line 4
```

Because a semicolon follows the closing parenthesis of the if statement (Line 1), the `else` statement stands alone. The semicolon at the end of the if statement (see Line 1) ends the if statement, so the statement at Line 2 separates the else clause from the if statement. That is, else is by itself. Because there is no separate else statement in Java, this code generates a syntax error.
Compound (Block of) Statements

Syntax:
{
    statement1
    statement2
    ...
    ...
    statementn
}

if (age > 18)
{
    System.out.println("Eligible to vote.");
    System.out.println("No longer a minor.");
}
else
{
    System.out.println("Not eligible to vote.");
    System.out.println("Still a minor.");
}
Conditional (?:) Operator

- Ternary operator
- Syntax:
  \[ \text{expression1} \ ? \ \text{expression2} \ : \ \text{expression3} \]
- If \( \text{expression1} = \text{true} \), then the result of the condition is \( \text{expression2} \).
  Otherwise, the result of the condition is \( \text{expression3} \).

Multiple Selection: Nested if

- Syntax:
  ```java
  if (expression1)
  statement1
else
  if (expression2)
  statement2
else
  statement3
  ```
- Else is associated with the most recent incomplete if.
- Multiple if statements can be used in place of if...else statements.
- May take longer to evaluate.
switch Structures

switch (expression)
{
    case value1: statements1
        break;
    case value2: statements2
        break;
    ...
    case valuen: statementsn
        break;
    default: statements
}

- Expression is also known as selector.
- Expression can be an identifier.
- Value can only be integral.
switch Structures

Example 4-24

```
switch (grade)
{
    case 'A': System.out.println("The grade is A.");
             break;
    case 'B': System.out.println("The grade is B.");
             break;
    case 'C': System.out.println("The grade is C.");
             break;
    case 'D': System.out.println("The grade is D.");
             break;
    case 'F': System.out.println("The grade is F.");
             break;
    default: System.out.println("The grade is invalid.");
}
```

Programming Example:
Cable Company Billing

◊ Input: Customer’s account number, customer code, number of premium channels to which customer subscribes, number of basic service connections (in the case of business customers).

◊ Output: Customer’s account number and the billing amount.
Programming Example: Cable Company Billing

Solution:
1. Prompt user for information.
2. Use switch statements based on customer’s type.
3. Use an if statement nested within a switch statement to determine the amount due by each customer.

Chapter Summary

- Control structures are used to process programs.
- Logical expressions and order of precedence of operators are used in expressions.
- Compare strings.
- If statements.
- if...else statements.
- switch structures.
- Proper syntax for using control statements.