Boolean Expressions & the ‘if’ Statement

September 22, 2006

Flow of Control

• Unless specified otherwise, the order of statement execution through a method is linear: one statement after another in sequence

• Some programming statements allow us to:
  • decide whether or not to execute a particular statement
  • execute a statement over and over, repetitively

• These decisions are based on boolean expressions (or conditions) that evaluate to true or false

• The order of statement execution is called the flow of control

Method Control Flow

• If the called method is in the same class, only the method name is needed

Method Control Flow

• The called method is often part of another class or object
Encapsulation

- An encapsulated object can be thought of as a **black box** -- its inner workings are hidden from the client.
- The client invokes the interface methods of the object, which manages the instance data.

Conditional Statements

- A conditional statement lets us choose which statement will be executed next.
- Therefore they are sometimes called **selection statements**.
- Conditional statements give us the power to make basic decisions.
- The Java conditional statements are:
  - if statement
  - if-else statement
  - switch statement

The if Statement

- The if statement has the following syntax:
  ```java
  if (condition) statement;
  ```
  - `if` is a Java reserved word.
  - The condition must be a boolean expression. It must evaluate to either true or false.
  - If the condition is true, the statement is executed. If it is false, the statement is skipped.

Boolean Expressions

- A condition often uses one of Java's equality operators or relational operators, which all return boolean results:
  ```java
  == equal to
  != not equal to
  < less than
  > greater than
  <= less than or equal to
  >= greater than or equal to
  ```
- Note the difference between the equality operator (==) and the assignment operator (=)
Example: Age.java (page 208)

Indentation

• The statement controlled by the if statement is indented to indicate that relationship

• The use of a consistent indentation style makes a program easier to read and understand

• Although it makes no difference to the compiler, proper indentation is crucial

  "Always code as if the person who ends up maintaining your code will be a violent psychopath who knows where you live."

  -- Martin Golding

The if Statement

• What do the following statements do?

  if (top >= MAXIMUM)
  top = 0;

  Sets top to zero if the current value of top is greater than or equal to the value of MAXIMUM

  if (total != stock + warehouse)
  inventoryError = true;

  Sets a flag to true if the value of total is not equal to the sum of stock and warehouse

• The precedence of the arithmetic operators is higher than the precedence of the equality and relational operators

Logical Operators

• Boolean expressions can also use the following logical operators:

  ! Logical NOT

  && Logical AND

  || Logical OR

• They all take boolean operands and produce boolean results

• Logical NOT is a unary operator (it operates on one operand)

• Logical AND and logical OR are binary operators (each operates on two operands)

Logical NOT

• The logical NOT operation is also called logical negation or logical complement

• If some boolean condition a is true, then !a is false; if a is false, then !a is true

• Logical expressions can be shown using a truth table

<table>
<thead>
<tr>
<th>a</th>
<th>!a</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Logical AND and Logical OR

• The logical AND expression

  a && b

  is true if both a and b are true, and false otherwise

• The logical OR expression

  a || b

  is true if a or b or both are true, and false otherwise
Logical Operators

- Expressions that use logical operators can form complex conditions
  
  ```java
  if (total < MAX+5 && !found)
      System.out.println("Processing...");
  ```

- All logical operators have lower precedence than the relational operators

- Logical NOT has higher precedence than logical AND and logical OR

---

Boolean Expressions

- Specific expressions can be evaluated using truth tables

```
<table>
<thead>
<tr>
<th>total &lt; MAX</th>
<th>found</th>
<th>!found</th>
<th>total &lt; MAX &amp;&amp; !found</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
```

---

Logical Operators

- A truth table shows all possible true-false combinations of the terms

- Since && and || each have two operands, there are four possible combinations of conditions a and b

| a   | b   | a && b | a || b |
|-----|-----|--------|--------|
| true| true| true   | true   |
| true| false| false  | true   |
| false| true| false  | true   |
| false| false| false  | false  |

---

Short-Circuited Operators

- The processing of logical AND and logical OR is “short-circuited”

- If the left operand is sufficient to determine the result, the right operand is not evaluated

  ```java
  if (count != 0 && total/count > MAX)
      System.out.println("Testing...");
  ```

- This type of processing must be used carefully

---

The if-else Statement

- An else clause can be added to an if statement to make an if-else statement

  ```java
  if (condition)
      statement1;
  else
      statement2;
  ```

- If the condition is true, statement1 is executed; if the condition is false, statement2 is executed

- One or the other will be executed, but not both

---

Example: Wages.java (page 211)
Logic of an if-else statement

- Condition evaluated
  - True
    - Statement1
  - False
    - Statement2

THE END