Comparing Data & the ‘switch’ Statement

September 25, 2006

The if Statement
- The if statement has the following syntax:
  ```java
  if (condition)
  
  statement;
  ```
- 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.

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.
Logical NOT

- The logical NOT operation is also called logical negation or logical complement.
- If some boolean condition \(a\) is true, then \(\neg a\) is false; if \(a\) is false, then \(\neg a\) is true.
- Logical expressions can be shown using a truth table:

<table>
<thead>
<tr>
<th></th>
<th>(\neg a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<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 \(b\):

|   |   | \(a \&\& b\) | \(a || b\) |
|---|---|-------------|------------|
| true | true | true        | true       |
| true | false | false       | true       |
| false | true | false       | true       |
| false | false | false       | false      |

Boolean Expressions

- Specific expressions can be evaluated using truth tables:

<table>
<thead>
<tr>
<th>total &lt; MAX</th>
<th>found</th>
<th>(\neg)found</th>
<th>total &lt; MAX &amp;&amp; (\neg)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>

Other Stuff from Section 5.2

Indentation Revisited

- Remember that indentation is for the human reader, and is ignored by the computer.

```
if (total > MAX)
    System.out.println ("Error!!");
errorCount++;
```

Despite what is implied by the indentation, the increment will occur whether the condition is true or not.

Block Statements

- Several statements can be grouped together into a block statement delimited by braces:
- A block statement can be used wherever a statement is called for in the Java syntax rules:

```
if (total > MAX)
{
    System.out.println ("Error!!");
    errorCount++;
}
```
Block Statements

- In an if-else statement, the if portion, or the else portion, or both, could be block statements

```java
if (total > MAX)
{
    System.out.println("Error!!");
    errorCount++;
}
else
{
    System.out.println("Total: "+total);
    current = total*2;
}
```

- See Guessing.java (page 216)

The Conditional Operator

- Java has a conditional operator that uses a boolean condition to determine which of two expressions is evaluated

```java
condition ? expression1 : expression2
```

- If the condition is true, expression1 is evaluated; if it is false, expression2 is evaluated

- The value of the entire conditional operator is the value of the selected expression

The Conditional Operator

- Another example:

```java
System.out.println("Your change is "+count+
                    ((count == 1) ? "Dime" : "Dimes");
```

- If count equals 1, then "Dime" is printed

- If count is anything other than 1, then "Dimes" is printed

Nested if Statements

- The statement executed as a result of an if statement or else clause could be another if statement

- These are called nested if statements

- See MinOfThree.java (page 219)

- An else clause is matched to the last unmatched if (no matter what the indentation implies)

- Braces can be used to specify the if statement to which an else clause belongs

The Coin Class

- Let’s examine a class that represents a coin that can be flipped

- Instance data is used to indicate which face (heads or tails) is currently showing

- See CoinFlip.java (page 213)

- See Coin.java (page 214)
The switch Statement

- The \textit{switch} statement provides another way to decide which statement to execute next
- The \textit{switch} statement evaluates an expression, then attempts to match the result to one of several possible cases
- Each case contains a value and a list of statements
- The flow of control transfers to statement associated with the first case value that matches

- Often a \textit{break} statement is used as the last statement in each case's statement list
- A \textit{break} statement causes control to transfer to the end of the \textit{switch} statement
- If a \textit{break} statement is not used, the flow of control will continue into the next case
- Sometimes this may be appropriate, but often we want to execute only the statements associated with one case

Example: \texttt{Guessing.java} (page 216)

\begin{verbatim}
switch (option) {
    case 'A':
        aCount++;
        break;
    case 'B':
        bCount++;
        break;
    case 'C':
        cCount++;
        break;
}
\end{verbatim}
The switch Statement

- The expression of a switch statement must result in an integral type, meaning an integer (byte, short, int, long) or a char.
- It cannot be a boolean value or a floating point value (float or double).
- The implicit boolean condition in a switch statement is equality.
- You cannot perform relational checks with a switch statement.
- See GradeReport.java (page 225)

Example: GradeReport.java (page 225)