Short-circuit Evaluation

- Sometimes only part of a boolean expression needs to be evaluated to determine the value of the entire expression.
  - If the first operand associated with an `||` is `true`, the expression is `true`.
  - If the first operand associated with an `&&` is `false`, the expression is `false`.
- This is called short-circuit or lazy evaluation.

Short-circuit Evaluation

- Short-circuit evaluation is not only efficient, sometimes it is essential!
- A run-time error can result, for example, from an attempt to divide by zero.

```
if ((number != 0) && (sum/number > 5))
```

- Complete evaluation can be achieved by substituting `&` for `&&` or `|` for `||`

The `switch` Statement

- The `switch` statement is a multiway branch that makes a decision based on an integral(integer or character) expression.
  - Java 7 allows String expressions
- The `switch` statement begins with the keyword `switch` followed by an integral expression in parentheses and called the controlling expression.

The `switch` Statement

- A list of cases follows, enclosed in braces.
- Each case consists of the keyword **`case`** followed by
  - A constant called the case label
  - A colon
  - A list of statements.
- The list is searched for a case label matching the controlling expression.

The `switch` Statement

- The action associated with a matching case label is executed.
- If no match is found, the case labeled `default` is executed.
  - The `default` case is optional, but recommended, even if it simply prints a message.
- Repeated case labels are not allowed.

The `switch` Statement

- Syntax

```
switch (Controlling_Expression)
{
  case Case_Label:
      Statement(s);
      break;
  case Case_Label:
  …
  default:
  …
}
```

The `switch` Statement

- The action for each case typically ends with the word `break`.
- The optional `break` statement prevents the consideration of other cases.
- The controlling expression can be anything that evaluates to an integral type.

Enumerations

- Consider a need to restrict contents of a variable to certain values
- An enumeration lists the values a variable can have
- Example

```
enum MovieRating {E, A, B}
MovieRating rating;
rating = MovieRating.A;
```

Enumerations

- Now possible to use in a **`switch`** statement

```
switch (rating)
{
   case E: //Excellent
      System.out.println("You must see this movie!");
      break;
   case A: //Average
      System.out.println("This movie is OK, but not great.");
      break;
   case B: // Bad
      System.out.println("Skip it!");
      break;
   default:
      System.out.println("Something is wrong.");
}
```

Enumerations

- An even better choice of descriptive identifiers for the constants

```
enum MovieRating
       {EXCELLENT, AVERAGE, BAD}
rating = MovieRating.AVERAGE;

case EXCELLENT:  ...
```