Lecture 3: Operators

Kenya 2005
Lecture Outline

• What operators are
• Arithmetic Operators such as +, -
• Assignment Operator
• Increment/Decrement Operators e.g i++
• Relational Operators
• Conditional Operators
What are Operators?

• Operators are special symbols used for:
  - mathematical functions
  - assignment statements
  - logical comparisons

• Examples of operators:
  3 + 5 // uses + operator
  14 + 5 – 4 * (5 – 3) // uses +, -, * operators

• *Expressions* can be combinations of variables, primitives and operators that result in a value
The Operator Groups

- There are 5 different groups of operators:
  - Arithmetic Operators
  - Assignment Operator
  - Increment / Decrement Operators
  - Relational Operators
  - Conditional Operators

- The following slides will explain the different groups in more detail.
Arithmetic Operators

• Java has 6 basic arithmetic operators:
  
  +     add
  -     subtract
  *     multiply
  /     divide
  %     modulo (remainder)
  ^     exponent (to the power of)

• The order of operations (or precedence) when evaluating an expression can be summarized in the acronym PEMDAS.
Order Of Operations

• Order of Operations (PEMDAS)
  1. Parentheses
  2. Exponents
  3. Multiplication and Division from left to right
  4. Addition and Subtraction from left to right

• An easy way to remember this is: “Please Excuse My Dear Aunt Sally”!
Order of Operations (Cont’d)

• Example: 10 + 15 / 5;

• The result is different depending on whether the addition or division is performed first

  \[(10 + 15) / 5 = 5\]
  \[10 + (15 / 5) = 13\]

Without parentheses, Java will choose the second case

• Note: you should be explicit and use parentheses to avoid confusion
Integer Division

- In the previous example, we were lucky that \((10 + 15) / 5\) gives an exact integer answer (5).

- But what if we divide 63 by 35?

- Depending on the data types of the variables that store the numbers, we will get different results.
• `int i = 63;
  int j = 35;
  System.out.println(i / j);
  Output: 1`

• `double x = 63;
  double y = 35;
  System.out.println(x / y);
  Output: 1.8`

• The result of integer division is just the integer part of the quotient!
Assignment Operator

• The basic assignment operator (=) assigns the value of `expr` to `var`

\[
\text{var} = \text{expr} ;
\]

• Java allows you to combine arithmetic and assignment operators into a single operator

• Examples:

\[
x = x + 5; \quad \text{is equivalent to} \quad x += 5;
\]
\[
y = y * 7; \quad \text{is equivalent to} \quad y *= 7;
\]
Increment/Decrement Operators

• `++` is called the increment operator. It is used to increase the value of a variable by 1.

  For example:
  
  $$i = i + 1;$$
  
  can be written as:
  
  ```
  ++i; or i++;
  ```

• `-->` is called the decrement operator. It is used to decrease the value of a variable by 1.

  $$i = i - 1;$$
  
  can be written as:
  
  ```
  --i; or i--;
  ```
Increment Operators (cont’d)

- The increment / decrement operator has two forms:

  - Prefix Form e.g. `++i;`  `--i;`
  - Postfix Form e.g. `i++;`  `i--;`
Prefix increment /decrement

• The prefix form first adds/ subtracts 1 from the variable and then continues to any other operator in the expression

• Example:

```java
int numOranges = 5;
int numApples = 10;
int numFruit;
numFruit = ++numOranges + numApples;
```

numFruit has value 16
numOranges has value 6
Postfix Increment/ Decrement

• The postfix form i++, i-- first evaluates the entire expression and then adds 1 to the variable

• Example:

```java
int numOranges = 5;
int numApples = 10;
int numFruit;
numFruit = numOranges++ + numApples;

numFruit has value 15
numOranges has value 6
```
Relational (Comparison) Operators

- Relational operators compare two values
- They produce a boolean value (true or false) depending on the relationship

<table>
<thead>
<tr>
<th>Operation</th>
<th>....Is true when</th>
</tr>
</thead>
<tbody>
<tr>
<td>a &gt; b</td>
<td>a is greater than b</td>
</tr>
<tr>
<td>a &gt;= b</td>
<td>a is greater than or equal to b</td>
</tr>
<tr>
<td>a == b</td>
<td>a is equal to b</td>
</tr>
<tr>
<td>a != b</td>
<td>a is not equal to b</td>
</tr>
<tr>
<td>a &lt;= b</td>
<td>a is less than or equal to b</td>
</tr>
<tr>
<td>a &lt; b</td>
<td>a is less than b</td>
</tr>
</tbody>
</table>

Note: == sign!
Examples of Relational Operations

```java
int x = 3;
int y = 5;
boolean result;

1) result = (x > y);
now result is assigned the value false because 3 is not greater than 5

2) result = (15 == x*y);
now result is assigned the value true because the product of 3 and 5 equals 15

3) result = (x != x*y);
now result is assigned the value true because the product of x and y (15) is not equal to x (3)
```
### Conditional Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>AND</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>NOT</td>
</tr>
</tbody>
</table>

- Conditional operators can be referred to as **boolean operators**, because they are only used to combine expressions that have a value of **true** or **false**.
# Truth Table for Conditional Operators

| x   | y    | x && y | x || y | !x |
|-----|------|--------|--------|----|
| True| True | True   | True   | False |
| True| False| False  | True   | False |
| False| True | False  | True   | True |
| False| False| False  | False  | True |
Examples of Conditional Operators

```java
boolean x = true;
boolean y = false;
boolean result;

1. Let result = (x && y);

   now result is assigned the value false (see truth table!)

2. Let result = ((x || y) && x);

   (x || y) evaluates to true
   (true && x) evaluates to true

   now result is assigned the value true
```
Using && and ||

• Java performs **short-circuit evaluation**: By this we mean that it evaluates && and || expressions from left to right and **once it finds the result, it stops**.

• Examples:
  
  \[(a \&\& (b++ > 3))\]
  
  \[(x \mid\mid y)\]

• Java will evaluate these expressions from left to right and so will evaluate
  
  \[a \text{ before } (b++ > 3)\]
  
  \[x \text{ before } y\]
Short-Circuit Evaluation

```java
(a && (b++ > 3));
```

What happens if `a` is `false`?

- Java will not evaluate the right-hand expression `(b++ > 3)` if the left-hand operator `a` is `false`, since the result is already determined in this case to be `false`. This means `b` will not be incremented!

```java
(x || y);
```

What happens if `x` is `true`?

- Similarly, Java will not evaluate the right-hand operator `y` if the left-hand operator `x` is `true`, since the result is already determined in this case to be `true`. 
POP QUIZ

1) What is the value of `number`?
   ```java
   int number = 5 * 3 - 3 / 6 - 9 * 3;
   ```

2) What is the value of `result`?
   ```java
   int x = 8;
   int y = 2;
   boolean result = (15 == x * y);
   ```

3) What is the value of `result`?
   ```java
   boolean x = 7;
   boolean result = (x < 8) && (x > 4);
   ```

4) What is the value of `numCars`?
   ```java
   int numBlueCars = 5;
   int numGreenCars = 10;
   int numCars = numGreenCars++ + numBlueCars + ++numGreenCars;
   ```
POP Quiz Solutions

1) What is the value of number?  -12
   int number = 5 * 3 - 3 / 6 - 9 * 3;

2) What is the value of result?  false
   int x = 8;
   int y = 2;
   boolean result = (15 == x * y);

3) What is the value of result?  true
   boolean x = 7;
   boolean result = (x < 8) && (x > 4);

4) What is the value of numCars?  27
   int numBlueCars = 5;
   int numGreenCars = 10;
   int numCars = numGreenCars++ + numBlueCars +
                 ++numGreeenCars;
Reference

• Summary of Java operators
This Lecture Covered....

• What Operators are
• The different types of operators
• The order of Operations for arithmetic operators
• Prefix and Postfix operators
• Short Circuit Evaluation
EC.S01 Internet Technology in Local and Global Communities
Spring 2005-Summer 2005

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.