Chapter 2: Java Fundamentals

Operators

Content

• Group of Operators
• Arithmetic Operators
• Assignment Operator
• Order of Precedence
• Increment/Decrement Operators
• Relational Operators
• Logical Operators
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 and operators that result in a value

Groups of Operators

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Java Operator</th>
<th>Example</th>
<th>Value (x = 10, y = 7, z = 2.5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
<td>x + y</td>
<td>17</td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
<td>x - y</td>
<td>3</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
<td>x * y</td>
<td>70</td>
</tr>
<tr>
<td>Division</td>
<td>/</td>
<td>x / y</td>
<td>1</td>
</tr>
<tr>
<td>Modulo division (remainder)</td>
<td>%</td>
<td>x % y</td>
<td>3</td>
</tr>
</tbody>
</table>

This is an integer division where the fractional part is truncated.
Example

Example of division issues:

10 / 3 gives 3
10.0 / 3 gives 3.33333

As we can see,

- if we divide two integers we get an integer result.
- if one or both operands is a floating-point value we get a floating-point result.

Modulus

- Generates the remainder when you divide two integer values.
  5%3 gives 2  5%4 gives 1
  5%5 gives 0  5%10 gives 5
- Modulus operator is most commonly used with integer operands. If we attempt to use the modulus operator on floating-point values we will garbage!
Order of Precedence

- ( ) evaluated first, inside-out
- *, /, or % evaluated second, left-to-right
- +, = evaluated last, left-to-right

Basic Assignment Operator

- We assign a value to a variable using the basic assignment operator (=).
- Assignment operator stores a value in memory.
- The syntax is

```plaintext
leftSide = rightSide;
```

Examples:

```plaintext
i = 1;
start = i;
sum = firstNumber + secondNumber;
avg = (one + two + three) / 3;
```
The Right Side of the Assignment Operator

- The Java assignment operator assigns the value on the right side of the operator to the variable appearing on the left side of the operator.

- The right side may be either:
  - **Literal**: ex. `i = 1;
  - **Variable identifier**: ex. `start = i;
  - **Expression**: ex. `sum = first + second;

Assigning Literals

- In this case, the literal is stored in the space memory allocated for the variable at the left side.

**Code**

```java
int firstNumber=1, secondNumber;
firstNumber = 234;
secondNumber = 87;
```

**State of Memory**

A. Variables are allocated in memory.

| firstNumber | 1 |
| secondNumber | ?? |

B. Literals are assigned to variables.

| firstNumber | 234 |
| secondNumber | 87 |
### Assigning Variables

In this case, the value of the variable at the right side is stored in the space memory allocated for the variable at the left side.

```c
int firstNumber=1, i;
firstNumber = 234;
i = firstNumber;
```

![State of Memory](image)

- **A.** Variables are allocated in memory.
  - `firstNumber` = 1
  - `i` = ???

- **B.** Values are assigned to variables.
  - `firstNumber` = 234
  - `i` = 234

### Assigning Expressions

In this case, the result of the evaluation of the expression is stored in the space memory allocated for variable at the left side.

```c
int first, second, sum;
first = 234;
second = 87;
Sum = first + second;
```

![State of Memory](image)

- **A.** Variables are allocated in memory.
  - `first` = 1
  - `second` = ???
  - `sum` = ???

- **B.** Values are assigned to variables.
  - `first` = 234
  - `second` = 87
  - `sum` = 321
**Updating Data**

A. The variable is allocated in memory.

B. The value 237 is assigned to number.

C. The value 35 overwrites the previous value 237.

---

**Code State of Memory**

```
int number;
number = 237;
number = 35;
```

---

**Example: Sum of two integer**

```java
public class Sum {
    // main method
    public static void main( String args[] ){
        int a, b, sum;
        a = 20;
        b = 10;
        sum = a + b;
        System.out.println(a + " + " + b + " = " + sum);
    } // end main
} // end class Sum
```
Java allows combining arithmetic and assignment operators into a single operator:

- Addition/assignment: `+=`
- Subtraction/assignment: `−=`
- Multiplication/assignment: `∗=`
- Division/assignment: `/=`
- Remainder/assignment: `%=`

The syntax is:
```
leftSide  Op= rightSide ;
```

This is equivalent to:
```
leftSide = leftSide  Op rightSide ;
```

- `x %= 5;`  `⇔  x = x % 5;`
- `x *= y+w*z;`  `⇔  x = x*(y+w*z);`

It is either a literal | a variable identifier | an expression.

It is an arithmetic operator.
Increment/Decrement Operators

Only use ++ or -- when a variable is being incremented/decremented as a statement by itself.

\[ x++; \text{ is equivalent to } x = x+1; \]

\[ x--; \text{ is equivalent to } x = x-1; \]

Relational 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>
Example

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

Logical 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>

<table>
<thead>
<tr>
<th>&amp;&amp;</th>
<th>T</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

| || | T | F |
|----|---|---|
| T  | T | F |
| T  | T | T |
| F  | T | F |
Example

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

result = (x && y);
result is assigned the value false

result = ((x || y) && x);
(x || y) evaluates to true
(true && x) evaluates to true
result is then assigned the value true
```

Operators Precedence

<table>
<thead>
<tr>
<th>Operators</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses</td>
<td>( ), inside-out</td>
</tr>
<tr>
<td>Increment/decrement</td>
<td>++, --, from left to right</td>
</tr>
<tr>
<td>Multiplicative</td>
<td>*, /, %, from left to right</td>
</tr>
<tr>
<td>Additive</td>
<td>+, - from left to right</td>
</tr>
<tr>
<td>Relational</td>
<td>&lt;, &gt;, &lt;=, &gt;=, from left to right</td>
</tr>
<tr>
<td>Equality</td>
<td>==, !=, from left to right</td>
</tr>
<tr>
<td>Logical AND</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>Logical OR</td>
<td></td>
</tr>
<tr>
<td>Assignment</td>
<td>=, +=, -=, *=, /=, %=</td>
</tr>
</tbody>
</table>