Handout #3 - Data Basics (Part 2)

<table>
<thead>
<tr>
<th>Math Operators</th>
<th>The MODULUS Operator %</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Addition</td>
<td>- Subtraction</td>
</tr>
<tr>
<td>* Multiplication</td>
<td>/ Division</td>
</tr>
<tr>
<td>% Modulus (also called remainder)</td>
<td></td>
</tr>
</tbody>
</table>

While there is no exponentiation symbol in Java, there is a Math.pow() method which will be discussed later.

The modulus operator finds the modulus of its first operand with respect to the second. That is, it produces the remainder of dividing the first value by the second value. For example:

\[
22 \mod 6 = 4 \quad \text{because} \quad 22 / 6 = 3 \text{ with a remainder of 4}
\]

Confusing DIVISIONS: Be careful when performing integer division. When dividing an integer by an integer, the answer will be an integer (not rounded). Compare these divisions:

(5 is an integer while 5.0 is a double)

<table>
<thead>
<tr>
<th>Division Type</th>
<th>Example 1</th>
<th>Example 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer division</td>
<td>8 / 5 = 1</td>
<td></td>
</tr>
<tr>
<td>Double division</td>
<td>8.0 / 5.0 = 1.6</td>
<td>8.0 / 5 = 1.6</td>
</tr>
<tr>
<td>Mixed division</td>
<td>8.0 / 5 = 1.6</td>
<td></td>
</tr>
</tbody>
</table>

When an operation involves two types (as the mixed division shown here), the smaller type is converted to the larger type. In this case, the integer 5 was converted to a double type before the division.

Mixed Mode Operations: Consider these examples:

```java
int a;
double b, c;
a = 3;
b = 5.1;
c = a + b;
```

When adding an int to a double, the int is converted to a double for the purpose of adding. The memory space retains the int. The location of the sum, c, must be a double.

```java
int a, b;
double c;
b = 21;
a = 5;
c = b/a;
```

Integer division takes place and gives an answer of 4. This answer is stored as a double 4.0. But what if we wanted the correct division answer ...

```java
c = (double) b/a;
c = 21.0 / 5
```

It is possible to force the type you want by type casting. Be careful to force the double to either the numerator or denominator, not both.

```java
c = (double) (b/a);
gives an answer 4.0 since integer division is done first.
```

Order of Operations: The normal rules you learned in mathematics for order of operations also apply to programming. The answer to \(2 + 7 \times 3\) is 23 (not 27). In math you learned to use PEMDAS (Please Excuse My Dear Aunt Sally) for determining order of operations.
Handout #3 - Data Basics (Part 2)

Printing Variables

Remember: The basic output statement in Java is

\[ \text{System.out.println ( );} \]

System.out.println(" text "); will print what is between the double quotes " " and move the printing
cursor to the next line.

System.out.print ( " text "); will print what is between the double quotes and leave the printing cursor
on the same line.

When dealing with variables:

\[ \text{System.out.println(" The total pay is " + totalPay);} \]

what is surrounded by " " is referred to as a "literal print" and gets printed exactly. The "+" sign is the
concatenator operator and concatenates the string with the value that is stored in the variable
totalPay. totalPay is declared as a double, but is automatically converted to a String for the
purpose of printing out.

double totalPay = 1006.5;
System.out.println("The total pay is " + totalPay);

On the screen:
The total pay is 1006.5

You can print an arithmetic expression within a System.out.print statement. Use parentheses around
the arithmetic expression to avoid unexpected problems.

\[ \text{System.out.println("Adding ten to the total: " + (totalPay + 10));} \]

\[ \text{System.out.println("answer " + 3 + 4); // becomes answer 34} \]

\[ \text{System.out.println("answer " + (3 + 4)); // becomes answer 7} \]