George Boole (1815-1864) was a British mathematician who invented the Boolean algebra, which is the basis of computer arithmetic for all modern computers. Because of this, we now consider Boole one of the founders of computer science even though electronic computers did not exist when he was alive.

Boolean algebra involves “arithmetic” similar to that of numbers that you learned in school, except that a Boolean operation is either true or false. Java represents these two truth values using the keywords `true` and `false`, respectively.

A variable of the built-in data type `boolean` can hold either the value `true` or `false` stored as an 8-bit integer.

### Example

```java
boolean done = false;
boolean success = true;
```

A relational operator takes two non-Boolean primitive operands, compares them and yields `true` or `false`.

<table>
<thead>
<tr>
<th>Relational Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Symbol</strong></td>
</tr>
<tr>
<td>&lt;</td>
</tr>
<tr>
<td>&lt;=</td>
</tr>
<tr>
<td>&gt;</td>
</tr>
<tr>
<td>&gt;=</td>
</tr>
</tbody>
</table>

*See the topic Data Types and Operators → Introduction to Java Data Types and Operators*
**Examples**

Assume `x` is initialized to 12.5.

- `x < 12.0` is `false`
- `x > 12.0` is `true`
- `x <= 12.5` is `true`
- `x <= 12.0` is `false`
- `x >= 12.5` is `true`
- `x >= 12.0` is `true`

A relational operator is evaluated after all arithmetic operators and before any assignment operator.

**Example**

The follow statement determines if a student got 75% or better on a test

```java
boolean passedTest = (total-numberWrong) / total >= 0.75;
```

Relational operators cannot be cascaded.

**Examples**

| 0 < x < 100 | Error! |

**Beginner Errors with Relational Operators**

The most common errors beginners make with relational operators, besides trying to cascade them as shown in the previous example, are to separate the symbol pair with a space or typing them in the wrong order.

**Examples**

| `x <= 20.0` | Error! |
| `x =< 20.0` | Error! |
Comparing Characters
You can compare character data using the relational and equality operators. Characters use alphabetical order with all the digits preceding the upper-case letters, which precede the lower-case ones.

<table>
<thead>
<tr>
<th>if c is</th>
<th>then ...</th>
<th>is...</th>
<th>Why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>'1'</td>
<td>c &lt; '9'</td>
<td>true</td>
<td>The decimal digits (‘0’, ‘1’, etc.) are in order.</td>
</tr>
<tr>
<td>'9'</td>
<td>c &lt; 'A'</td>
<td>true</td>
<td>The decimal digits precede the upper-case letters.</td>
</tr>
<tr>
<td>'A'</td>
<td>c &lt; 'Z'</td>
<td>true</td>
<td>The upper-case letters are in order.</td>
</tr>
<tr>
<td>'Z'</td>
<td>c &lt; 'a'</td>
<td>true</td>
<td>The upper-case letters precede the lower-case ones.</td>
</tr>
<tr>
<td>'a'</td>
<td>c &lt; 'z'</td>
<td>true</td>
<td>The lower-case letters are in order.</td>
</tr>
</tbody>
</table>
**Exercises**

1. Write the Java statement to declare **done a boolean** variable initialized to **false**.

Assume **x** is a **double** variable. For each of the following expressions, circle what is wrong and explain. None of them is correct.

2. \[ x \leq 12.5 \]
3. \[ x \leq "12.5" \]
4. \[ x \Rightarrow 12.5 \]
5. \[ 0 \leq x \leq 100 \]

Give the output of each of the following.

6. ```java
double x = 100;
System.out.println( x <= 100 );
```
7. ```java
double x = 100;
System.out.println( x < 100 );
```
8. ```java
double x = 100;
System.out.println( x > 100 );
```
9. ```java
double x = 100;
System.out.println( x >= 100 );
```
10. ```java
char c = 'n';
System.out.println( c > 'N' );
```
11. ```java
char c = '1';
System.out.println( c < '9' );
```
12. ```java
char c = '1';
System.out.println( c < 9 );
```
13. ```java
char c = '1';
System.out.println( 'c' < 9 );
```
14. ```java
char c = '1';
System.out.println( 'c' < '9' );
```