Selections - Branches

Programs can decide among different execution options, *branches*, based on conditions.

Basic selection statement:

```java
if ( <condition> )
  <statement>
else
  <statement>
```

The `<condition>` of a selection statement is a *boolean expression* that evaluates to *true* or *false*. To compute a boolean expression we use boolean operators:

- `<`  less than
- `<=`  less or equal than
- `>`  greater than
- `>=`  greater or equal than
- `!=`  not equal to
- `==`  equal to

**Note:** in two character operators there is no space in the middle.
The `<statement>` is either a single statement terminated by a semicolon or a block of statements enclosed in brackets.

Common pitfalls of selections:

- `=` is the assignment operator, `==` is the equality operator

```java
if ( f == true ) // execute if f is true
if ( f = true ) // set f to true, always true
```

- appending a semicolon to the end of the if or else before block

```java
if ( ... ) ;
{
}
else ;
{
}
```

- adding an empty block at the end of the conditional

```java
if ( ... ) { }
{
}
```
- Incorrect nesting of selection statement
  
  ```java
  if ( ... )
  {
    if ( ... )
  }
  else
  {
  }
  ```
  
  Not the same as:

  ```java
  if ( ... )
  {
    if ( ... )
  } else
  ```

- Dangling else

  ```java
  if ( ... )
  {
    if ( ... )
    else
  }
  ```

- When using real numbers, only use `<` and `>`. Because of the loss of precision in the representation of real numbers, equality is not guaranteed.
**Boolean Operators**

Compound boolean expressions created with boolean operators:

\[ A \text{ AND } B \Rightarrow \text{ both } A \text{ and } B \text{ true} \]

\[ A \text{ OR } B \Rightarrow \text{ either } A \text{ or } B \text{ true} \]

\[ \neg A \Rightarrow \text{ A is false} \]

\[ A \text{ XOR } B \Rightarrow \text{ A or B false, but not both} \]

**Note:** two character operators have no space. & and | are bitwise operators.

See the effect of boolean operators using *truth tables*:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A &amp;&amp; B</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

| A | B | A || B |
|---|---|--------|
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A ^ B</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>!A</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
The switch statement

The switch statement is a multi-way branching that will execute a statement or a series of statements based on the value of a variable or an expression:

```java
switch ( <switch expression> )
{
    case <value1>:
        <statement(s)>
        break;
    case <value2>:
        <statement(s)>
        break;
    ...
    default:
        <statement(s)>
        break;
}
```

The values in the `case` statements are constant expressions and have to be of the same type as the switch expression.

The type of the switch expression can be `byte`, `char`, `short` and `int`. With Java SDK 7 it can also be `String`. 
When the program encounters a `switch` statement, it evaluates the switch expression and jumps to the `case` with a matching value. If there is no matching value it will jump to the optional `default` case, if present.

After jumping to a `case` (or to the `default`) label the program executes the statements that follow until it reaches a `break` or the switch closing bracket.

Two or more case labels can be combined:

```java
switch ( expression )
{
  case 1:
    statement1; // Executed if expression == 1
  case 2:
  case 3:
    statement2; // Executed if expression == 1,2,3
    break;
  case 4:
    statement4; // Executed if expression == 4
    break;
}
```
Conditional Expression

Evaluate an expression based on a boolean condition.

\[
\text{double rate} = \text{taxable} > 50000 \ ? \ 0.28 \ : \ 0.15;
\]

equivalent to

\[
\text{double rate};
\text{if ( taxable > 50000 )}
\quad \text{rate} = 0.28;
\text{else}
\quad \text{rate} = 0.15;
\]