Chapter 6: Selection Statements

- boolean expressions
  - constants: `true` and `false`
  - method return values: if a method returns `true` or `false`, then sending the message counts as a boolean expression
  - relational operators: $>$, $>=$, $<$, $<=$, $==$ and $!=$. These are all *binary* operators.
    - ∗ $>$, $>=$, $<$ and $<=$ require a numeric primitive types or `char` for operands, and behave as expected
    - ∗ $==$ and $!=$ can have numeric primitive types or `char` as operands, or any `Object` reference.

- boolean operators: && (AND), ‖ (OR), !(NOT). These may be applied to boolean expressions to yield another boolean expression.
  - ∗ && and ‖ both take two boolean expressions as operands.
  - ∗ ! takes one boolean expression as an operand.

- `if` statments
  - format: `if (<boolean expression>)` { `// ‘‘true’’ code` } `else` { `// ‘‘false’’ code` }
  - `‘‘false’’ code` optional
  - braces are optional if code body is only one command
  - ***see the Style Guide for rules about indentation!***
  - *any* boolean expression is allowed, so may be quite complex using boolan operators!

- `switch` statements
  - format: `switch(<numeric type>)` { `case <value>: ...; default: ...` }
  - may have multiple `case <value>` statements
  - note that `<value>` is *not* a boolean expression, but rather a constant!
  - `default` statement is optional
  - `break;` statement needed between `case` statements if you want code to skip to end of `switch` block.
  - ***see the Style Guide for rules about indentation!***

`ListBox` from Javabook

- used to offer discrete choices for user to select from
- constructor requires a `MainWindow` object as an argument, just like `InputBox` and `OutputBox`
- method `addItem(String)` adds a choice to the `ListBox`
- method `getSelectedIndex()` makes the `ListBox` appear, and waits until the user selects an item
- a *zero-based index* is returned from `getSelectedIndex()`, so `switch` is a good way to handle the response
- two class constants are defined:
  - `ListBox.NO_SELECTION` - when the user closes the `ListBox` window without selecting anything (using the X at the top)
  - `ListBox.CANCEL` - when the user clicks the `CANCEL` button rather than the `OK` button in the window