

Counter Controlled Repetition

- Elements of a loop
 - Control variable, initial value, increment or decrement, final condition
- Figure 3.1 - **While** loop to control **counter**
 - **counter** should be an integer, floating point is more imprecise
 - **yPos** is a parameter and is changed with each iteration of the loop
 - The boolean (or conditional) expression is checked before each iteration of the loop, until it becomes false
- Concise Java notation may obscure the code (page 124) and doesn't even work unless **counter** is initialized to 0
- Note for loop in Figure 3.2 - initialization, condition, and increment are all contained in the control statement
 - "Off by one" errors are common in loops - test them!
 - The expressions in a for statement are not mandatory
 - Each expression can actually be a comma separated list of statements, but do this with caution
 - If the loop condition is false, the body will not be executed.
- Every for loop can be written as a while loop - see page 126
- Examples - page 128
- Figure 3.6 - compound interest (**Interest.java**)
 - Main calculation: $a = p(1+r)^n$ - line 16
 - Note **Math.pow(number, power)**
 - Check out **Math** in the help
 - Both arguments are **double**
 - **Math.pow(1.0 + r, n)**
- Floats are not good for doing monetary calculations – page 131
 - The **DecimalFormat** class is introduced in the source code

Switch statement for multiple selection

- Figure 3.7, **SwitchTest.java**
 - **val.charAt(loc)** accesses a specific character in a string, starting at 0
 - **e.getActionCommand()** is equivalent to **input.getText()**
 - Only the first letter is used, the rest of the **TextField** is discarded
- Each **switch** has a controlling expression and case labels
- Without a **break** statement the flow will continue sequentially to the next case
- Braces are not required around multiple actions
- Several case labels can be listed together
 - In figure 3.7, this is used to process upper and lower case.
- The expression must yield a constant integer or character expression
 - Any combination of character constants and integer constants that evaluates to a constant integer value.
 - Remember, characters are stored as integers!

- Conceptually similar to a nested `if ... else if` but much more efficient

Do while

- The block is always performed at least once.
- The condition is checked after the block is executed, not before as in the `while`

Break and Continue

- `break` and `continue` alter the flow of control
- `break` causes an immediate exit from a loop or switch
- `continue` skips the remaining statements in the body of a loop
- Examples - Fig 3.11 and Fig 3.12 (`BreakTest.java` and `ContinueTest.java`)
- A labeled `break` is used to break out of a nested structure
 - Execution resumes with the first statement after the enclosing labeled compound statement
 - A compound statement is a group of statements enclosed in braces
 - Any compound statement can be labeled
- Figure 3.13 and Figure 3.14 - It's a good idea to comment the compound statement
 - `xPos` controls the character position (`column`)
 - `yPos` controls the line position (`row`)
- A labeled `continue` requires a labeled repetition structure (Fig 3.14)

Logical Operators

- `&&` logical and, used to combine logical expressions or conditions; both conditions must be true
- `||` logical or, at least one expression must be true
- `&&` has higher precedence than `||`
 - When in doubt, use parenthesis
- `&&` and `||` will not evaluate unnecessary expressions once the overall expression value is determined. "Short circuit" evaluation

```
if ( (count != 0) && ( sum/count > 60) pass = true;
```

 - `&` and `|` evaluate every expression, even if it's not necessary
 - good example on page 146

```
(birthday == true | ++age >= 65)
```
- `^` or condition where exactly one expression is true; logical exclusive or
- `!` is logical negation
 - `!` has higher precedence than the equal operator
 - again, use parenthesis
- Note operator precedence on page 149
- Fig 3.19 actually outputs the truth tables