How am I better than a Scarecrow?
Avoid **break** and **continue**??

Savitch: "Because of the complications they introduce, break statements and continue statements within loops should be avoided. Any loop that includes either of these statements can be written without one."

```java
1) for (int i = 0; i<str.length(); i++)
2) {
3)    char c = str.charAt(i);
4)    if (Character.isDigit(c)) digitCount++;
5)    else if (Character.isAlphabetic(c)) letterCount++;
6)    else break;
7) }
```

```java
1) int i = 0; boolean good = true;
2) while ((i<str.length()) && good)
3) {
4)    char c = str.charAt(i);
5)    if (Character.isDigit(c)) digitCount++;
6)    else if (Character.isAlphabetic(c)) letterCount++;
7)    else good=false;
8)    i++;
9) }
```

**Loop Bugs**

Programs containing loops are more likely to have mistakes than simpler programs.

The two most common kinds of loop errors are

- **Unintended infinite loops:**
  - A loop might terminate for some input data values but repeat infinitely for other values.
  - Mitigation: Spend time making sets of test data.

- **Off-by-one errors:**
  - Mitigation: Test by printing the results of a loop.
Always Retest

Whenever you find a bug in a program and fix it, always retest the program with all test cases.

- Yet another bug might be waiting or
- The "fix" may have introduced a new bug.
- This process of retesting a modified program with a full set of test cases is called regression testing.

Tracing Variables and Control Flow

If your program misbehaves, but you cannot see what is wrong, your best bet is to trace some key variables and the trace the control flow.

```java
if (z < 10)
{
    System.out.println("Entered z<10");
    x = x + y * z
    System.out.println("Line 128, x=" + x);
}
```

**Control Flow Trace:**
Whenever your code enters a new block, print something that identifies the block.

**Variable Trace:**
Whenever a variable's value changes, print the code location, the variable's name and its value.
Eclipse Integrated Debugger

1. Add one or more **breakpoints** to your code. Note a breakpoint can only be added to an executable line of code.

2. Select the **Debug Perspective**.

3. Click the "**Run in Debug Mode**" Icon. This will run your code from the beginning up until control reaches the first breakpoint.

4. Add trace variables to the list of **Expressions**.

5. Click the "**Step Over**" icon to fully execute the current (highlighted) statement. If the current statement includes any method calls, then those methods are fully executed.

6. "**Step Into**" is the same as "Step Over" if there are no method calls in the current statement. If the current statement contains a method call, then execution pauses at the first statement within that method.

7. Click the "**Resume**" Icon to run your code from the current line through to the next breakpoint.

---

**Diagram:**

- **Run in Debug mode**
- **Step Into**
- **Step Over**
- **Debug Perspective**
- **Breakpoint**
- **Execution Pointer**
- **Trace Variables**
- **Continue to next Breakpoint**