EXP51-J. Do not perform assignments in conditional expressions

Using the assignment operator in conditional expressions frequently indicates programmer error and can result in unexpected behavior. The assignment operator should not be used in the following contexts:

- `if` (controlling expression)
- `while` (controlling expression)
- `do ... while` (controlling expression)
- `for` (second operand)
- `switch` (controlling expression)
- `?:` (first operand)
- `&&` (either operand)
- `||` (either operand)
- `?:` (second or third operands) where the ternary expression is used in any of these contexts

Noncompliant Code Example

In this noncompliant code example, the controlling expression in the `if` statement is an assignment expression:

```
public void f(boolean a, boolean b) {
  if (a = b) {
    /* ... */
  }
}
```

Although the programmer's intent could have been to assign b to a and test the value of the result, this usage frequently occurs when the programmer mistakenly used the assignment operator = rather than the equality operator ==.

Compliant Solution

The conditional block shown in this compliant solution executes only when a is equal to b:

```
public void f(boolean a, boolean b) {
  if (a == b) {
    /* ... */
  }
}
```

Unintended assignment of b to a cannot occur.

Compliant Solution

When the assignment is intended, this compliant solution clarifies the programmer's intent:

```
public void f(boolean a, boolean b) {
  if ((a = b) == true) {
    /* ... */
  }
}
```

Compliant Solution

It may be clearer to express the logic as an explicit assignment followed by the `if` condition:

```
public void f(boolean a, boolean b) {
  a = b;
  if (a) {
    /* ... */
  }
}
```

Noncompliant Code Example

In this noncompliant code example, an assignment expression appears as an operand of the `&&` operator:

```
public void f(boolean a, boolean b, boolean flag) {
  while ( (a = b) && flag ) {
    /* ... */
  }
}
```

Because `&&` is not a comparison operator, assignment is an illegal operand. Again, this is frequently a case of the programmer mistakenly using the assignment operator = instead of the equals operator ==.

Compliant Solution

When the assignment of `b` to `a` is unintended, this conditional block is now executed only when `a` is equal to `b` and `flag` is `true`:

```
public void f(boolean a, boolean b, boolean flag) {
  while ( (a == b) && flag ) {
    /* ... */
  }
}
```

Applicability

The use of the assignment operator in controlling conditional expressions frequently indicates programmer error and can result in unexpected behavior.

As an exception to this guideline, it is permitted to use the assignment operator in conditional expressions when the assignment is not the controlling expression (that is, the assignment is a subexpression), as shown in the following compliant solution:

```
public void assignNocontrol(BufferedReader reader)
    throws IOException{
  String line;
  while ((line = reader.readLine()) != null) {
    // ... Work with line
  }
}
```

Automated Detection

Tool	Version	Checker	Description
SonarQube	6.7	**AssignmentInSubExpressionCheck**	

Bibliography

[Hatton 1995]	§2.7.2, "Errors of Omission and Addition"