OBJ54-J. Do not attempt to help the garbage collector by setting local reference variables to null

Setting local reference variables to `null` to "help the garbage collector" is unnecessary. It adds clutter to the code and can make maintenance difficult. Java just-in-time compilers (JITs) can perform an equivalent liveness analysis, and most implementations do so.

A related bad practice is use of a finalizer to `null` out references. See MET12-J. Do not use finalizers for additional details.

This guideline applies specifically to local variables. For a case where explicitly erasing objects is useful, see OBJ55-J. Remove short-lived objects from long-lived container objects.

Noncompliant Code Example

In this noncompliant code example, `buffer` is a local variable that holds a reference to a temporary array. The programmer attempts to help the garbage collector by assigning `null` to the `buffer` array when it is no longer needed.

```
{ // Local scope
  int[] buffer = new int[100];
  doSomething(buffer);
  buffer = null;
}
```

Compliant Solution

Program logic occasionally requires tight control over the lifetime of an object referenced from a local variable. In the unusual cases where such control is necessary, use a lexical block to limit the scope of the variable because the garbage collector can collect the object immediately when it goes out of scope [Bloch 2008].

This compliant solution uses a lexical block to control the lifetime of the `buffer` object:

```
{ // Limit the scope of buffer
  int[] buffer = new int[100];
  doSomething(buffer);
}
```

Applicability

It is unnecessary to set local reference variables to `null` when they are no longer needed in a mistaken attempt to help the garbage collector reclaim the associated memory.

Bibliography

[Bloch 2008]	Item 6, "Eliminate Obsolete Object References"