Mutating on realtime and non-realtime

```
void addSource (const float* src) {
    if (! isRealtimeThread()) {
        realtimeThreadCaller.callAsync([src] () { addSource (src); });
        return;
    }
    RealtimeMutatable<SourceList>::ScopedAccess<true> sourceList (sharedSourceList);
    assert (sourceList->numSources < MAX_SOURCES);
    sourceList->buffers[sourceList->numSources++] = src;
}
```

75

Mutating on realtime and non-realtime

```
void addSource (const float* src) {
    if (! isRealtimeThread()) {
        realtimeThreadCaller.callAsync([src] () { addSource (src); });
        return;
    }
    RealtimeMutatable<SourceList>::ScopedAccess<true> sourceList (sharedSourceList);
    assert (sourceList->numSources < MAX_SOURCES);
    sourceList->buffers[sourceList->numSources++] = src;
}
```

Mutating on realtime and non-realtime

```
void mixAllSources (float* output, char* realtimeEventMessages, int n) {
    processRealtimeEvents(realtimeEventMessages); // may add and remove sources
    realtimeThreadCaller.process(); // process all the lambdas

    RealtimeMutatable<SourceList>::ScopedAccess<true> sourceList (sharedSourceList);
    for (int i = 0; i < sourceList->numSources; ++i)
        mixSource (output, sourceList->buffers[i]);
}
```

```
void printSources() {
    RealtimeMutatable<SourceList>::ScopedAccess<false> sourceList (sharedSourceList);
    for (int i = 0; i < sourceList->numSources; ++i)
        std::cout << (void*)sourceList->buffers[i] << std::endl;
}
```