SimpleNodePlayer::Process

```
void process (const Node::ProcessContext& pc)
{
    // Prepare all nodes for the next block
    for (auto node : orderedNodes)
        node->prepareForNextBlock (pc.referenceSampleRange);

    // Then process them all in sequence
    for (auto node : orderedNodes)
        node->process (pc.referenceSampleRange);

    // Finally copy the output from the root Node to our player buffers
    auto output = rootNode->getProcessedOutput();
    const size_t numAudioChannels = std::min (output.audio.getNumChannels(),
                                              pc.buffers.audio.getNumChannels());

    if (numAudioChannels > 0)
        pc.buffers.audio.getSubsetChannelBlock (0, numAudioChannels)
                        .add (output.audio.getSubsetChannelBlock (0, numAudioChannels));

    pc.buffers.midi.mergeFrom (output.midi);
}
```

SimpleNodePlayer::Process

```
void process (const Node::ProcessContext& pc)
{
    // Prepare all nodes for the next block
    for (auto node : orderedNodes)
        node->prepareForNextBlock (pc.referenceSampleRange);

    // Then process them all in sequence
    for (auto node : orderedNodes)
        node->process (pc.referenceSampleRange);

    // Finally copy the output from the root Node to our player buffers
    auto output = rootNode->getProcessedOutput();
    const size_t numAudioChannels = std::min (output.audio.getNumChannels(),
                                              pc.buffers.audio.getNumChannels());

    if (numAudioChannels > 0)
        pc.buffers.audio.getSubsetChannelBlock (0, numAudioChannels)
                        .add (output.audio.getSubsetChannelBlock (0, numAudioChannels));

    pc.buffers.midi.mergeFrom (output.midi);
}
```

Summary of `NodePlayer` Class