```cpp
/**
    Simple player for a Node.
    This iterates all the nodes attempting to process them in a single thread.
*/
class SimpleNodePlayer
{
public:
    /** Creates a player to play a Node. */
    SimpleNodePlayer (std::unique_ptr<Node> nodeToPlay)
        : rootNode (std::move (nodeToPlay))
    {
        assert (rootNode);
    }

    /** Prepares the Node to be played. */
    void prepareToPlay (double sampleRateToUse, int blockSizeToUse)
    {
        orderedNodes = node_player_utils::prepareToPlay (rootNode.get(), nullptr, sampleRateToUse, blockSizeToUse);
    }

    /** Processes a block of audio and MIDI data. */
    void process (const Node::ProcessContext&);

private:
    std::unique_ptr<Node> rootNode;
    std::vector<Node*> orderedNodes;
};
```

SimpleNodePlayer

SimpleNodePlayer

```cpp
/**
    Simple player for a Node.
    This iterates all the nodes attempting to process them in a single thread.
*/
class SimpleNodePlayer
{
public:
    /** Creates a player to play a Node. */
    SimpleNodePlayer (std::unique_ptr<Node> nodeToPlay)
        : rootNode (std::move (nodeToPlay))
    {
        assert (rootNode);
    }

    /** Prepares the Node to be played. */
    void prepareToPlay (double sampleRateToUse, int blockSizeToUse)
    {
        orderedNodes = node_player_utils::prepareToPlay (rootNode.get(), nullptr, sampleRateToUse, blockSizeToUse);
    }

    /** Processes a block of audio and MIDI data. */
    void process (const Node::ProcessContext&);

private:
    std::unique_ptr<Node> rootNode;
    std::vector<Node*> orderedNodes;
};
```

prepareToPlay

```cpp
namespace node_player_utils
{
    /** Prepares a specific Node to be played and returns all the Nodes. */
    static std::vector<Node*> prepareToPlay (Node* node, Node* oldNode,
                                             double sampleRate, int blockSize)
    {
        if (node == nullptr)
            return {};

        // First give the Nodes a chance to transform
        transformNodes (*node);

        // Then find all the nodes as it might have changed after initialisation
        auto orderedNodes = tracktion_graph::getNodes (*node, tracktion_graph::VertexOrdering::postordering);

        // Next, initialise all the nodes, this will call prepareToPlay on
        const PlaybackInitialisationInfo info { sampleRate, blockSize, *node, oldNode };

        for (auto node : orderedNodes)
            node->initialise (info);

        // Finally return the Nodes in playback order
        return orderedNodes;
    }
}
```