**Single Json Object Transformation**

Traversal Types:- topdown, r-topdown, bottomup, r-bottomup, prefix, infix, postfix

outputNode processNode(inputNode, parentNode, configData, processNodeContextData)

outputObject prepareOutput(outputNode, parentOutputNode, currentOutputObject, configData, prepareOutputContextData)

Array<ChildNode> skipTraversalList(inputNode, outputNode, parentNode, configData, skipContextData)

outputObject transform (inputObject, initialOutputObject, configData, traversalType, processNode, prepareOutput, skipTraversalList)

*Algo:*

Step 1: Create empty output object if initial output is not available

Step 2: Traverse to appropriate node using selected traversal type

Step 3: Process Node

Step 4: Prepare Output

Step 5: filter/skip some nodes in Input object using skipTraversalList

Step 6: Repeat from step 2 until traversal complete.

*Future Directions*: How to make it parallel execution & how to handle stateful context data.

**2 Json objects merge transformation**

outputNode processNodes(inputNode1, parentNode1, inputNode2, parentNode2, configData, processNodeContextData)

void postprocessOutputArrayNode (arrayOutputNode, configData, postprocessArrayContextData)

outputObject prepareOutput(outputNode1, parentOutputNode1, outputNode2, parentOutputNode2, currentOutputObject, configData, prepareOutputContextData)

Array<ChildNode> skipTraversalList1(inputNode1, outputNode, parentNode1, configData, skipContextData)

Array<ChildNode> skipTraversalList2(inputNode2, outputNode, parentNode2, configData, skipContextData)

outputObject transform (inputObject1, inputObject2, initialOutputObject, traversalType, processNodes, postprocessOutputArrayNode, prepareOutput, skipTraversalList1, skipTraversalList2)