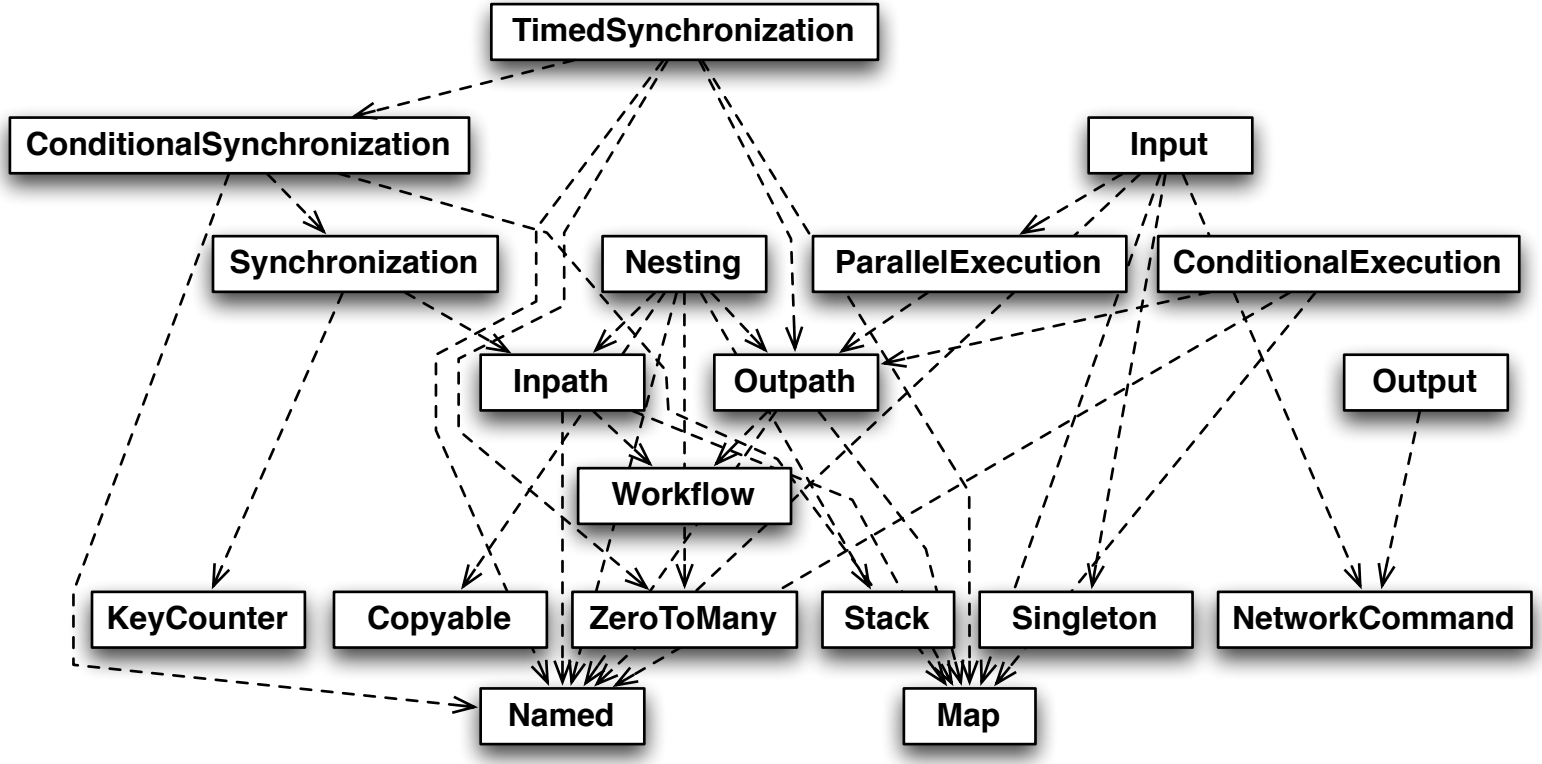
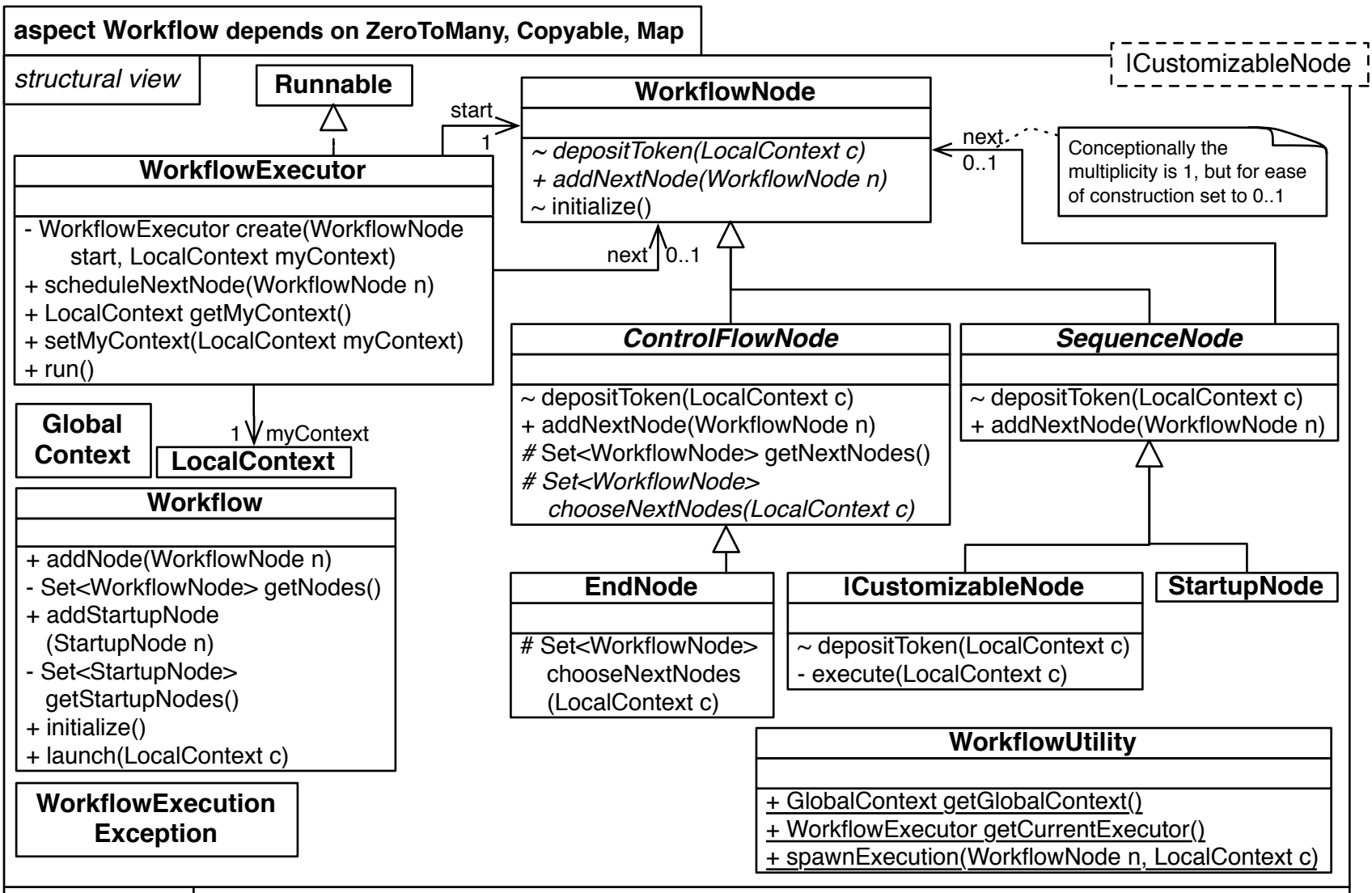

TimedSynchronization
ConditionalSynchronization
Input
Synchronization
Nesting
ParallelExecution
ConditionalExecution
Inpath
Outpath
Output
Workflow
KeyCounter
Copyable
ZeroToMany
Stack
Singleton
NetworkCommand
Named
Map

aspect Workflow depends on ZeroToMany, Copyable, Map

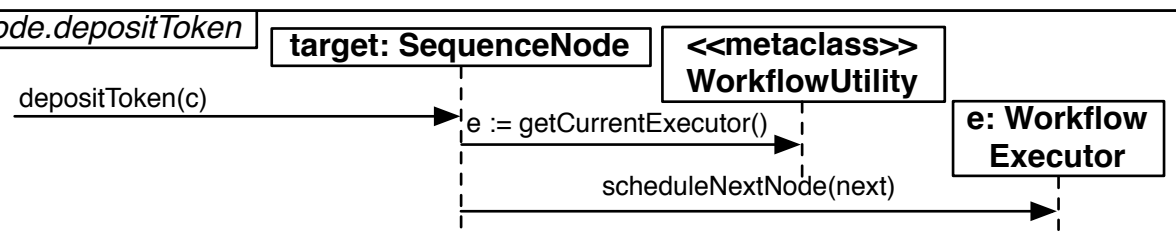
structural view



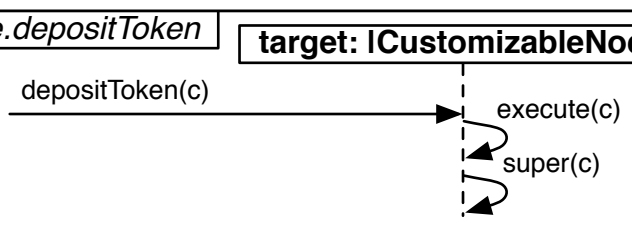
Instantiations:	
Copyable:	**Copyable → LocalContext**
ZeroToMany:	**IData → ControlFlowNode; IAssociated → WorkflowNode; getAssociated → getNextNodes; add → addNextNode**
ZeroToMany:	**IData → Workflow; IAssociated → WorkflowNode; getAssociated → getNodes; add → addNode**
ZeroToMany:	**IData → Workflow; IAssociated → StartupNode; getAssociated → getStartupNodes; add → addStartupNode**
Map:	**IData → WorkflowUtility; IKey → Thread; IValue → WorkflowExecutor; getValue → getWorkflowExecutor; add → addWorkflowExecutor**

Implementation:	
Thread:	java.lang.Thread

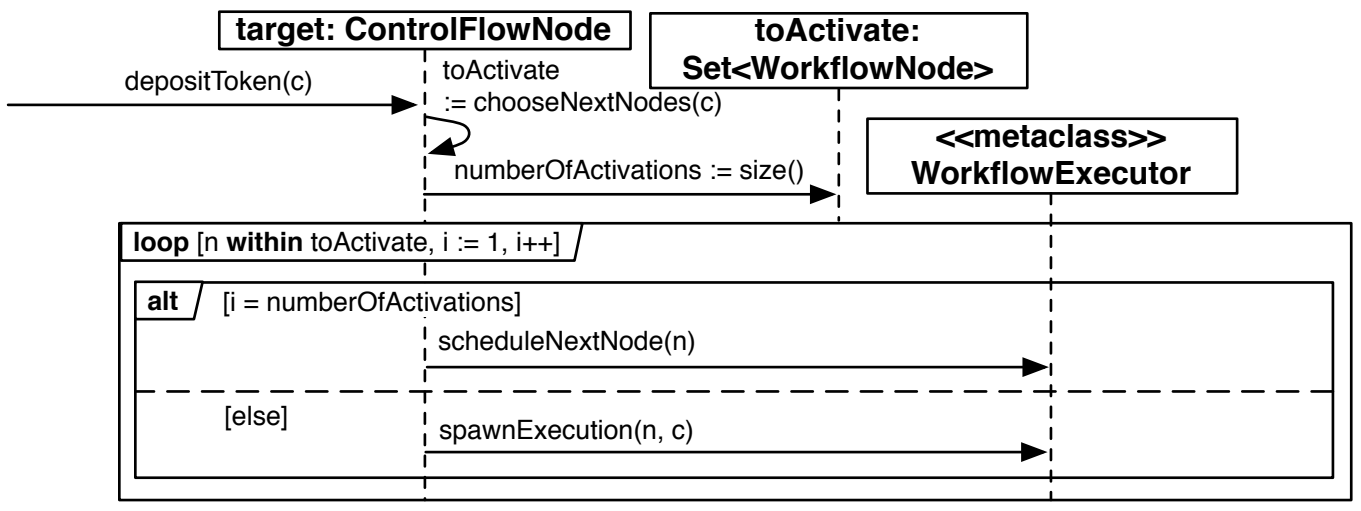
message view SequenceNode.depositToken



message view ICustomizableNode.depositToken



message view ControlFlowNode.depositToken



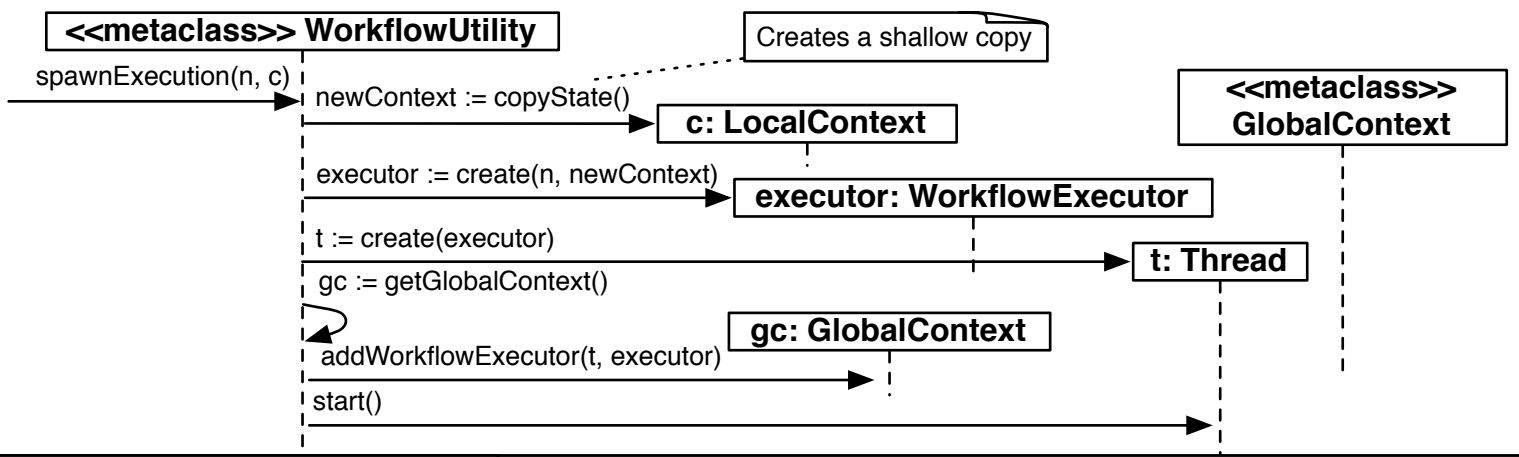
message view SequenceNode.addNextNode is Setter<next>

message view getMyContext is Getter<myContext>

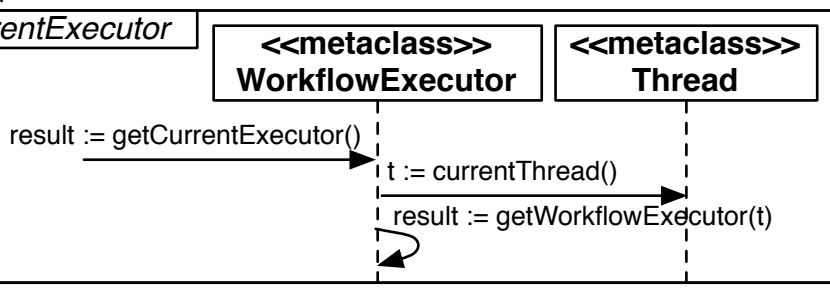
message view setMyContext is Setter<myContext>

message view scheduleNextNode is Setter<next>

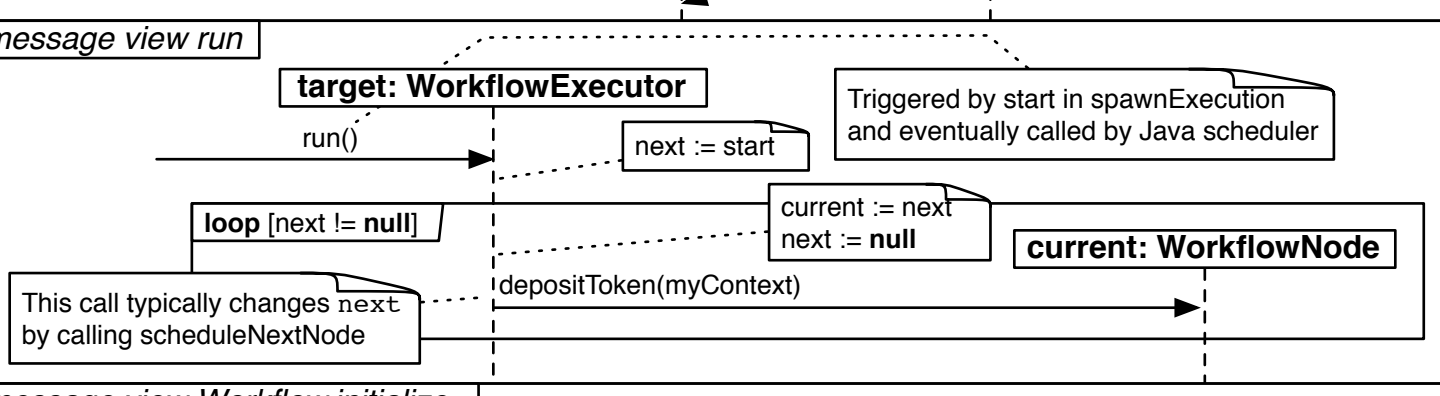
message view spawnExecution



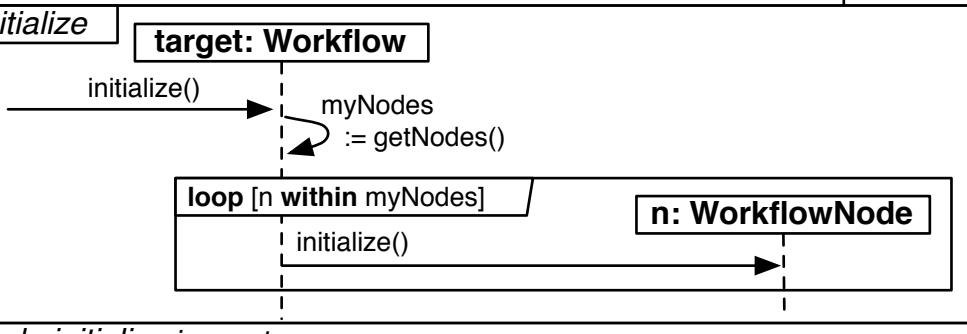
message view getCurrentExecutor



message view run

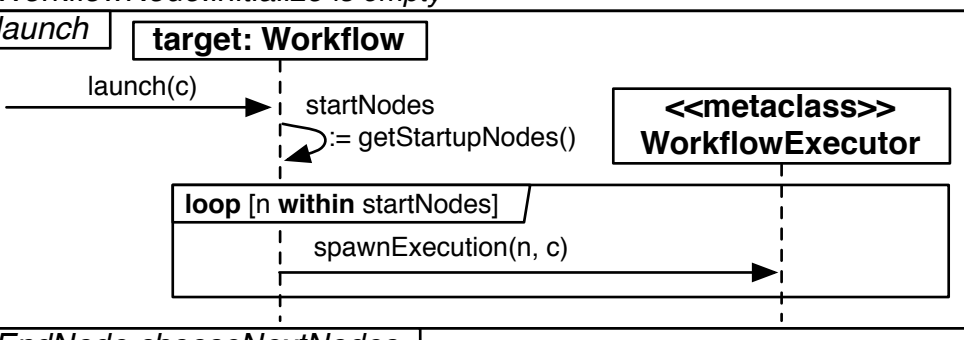


message view Workflow.initialize

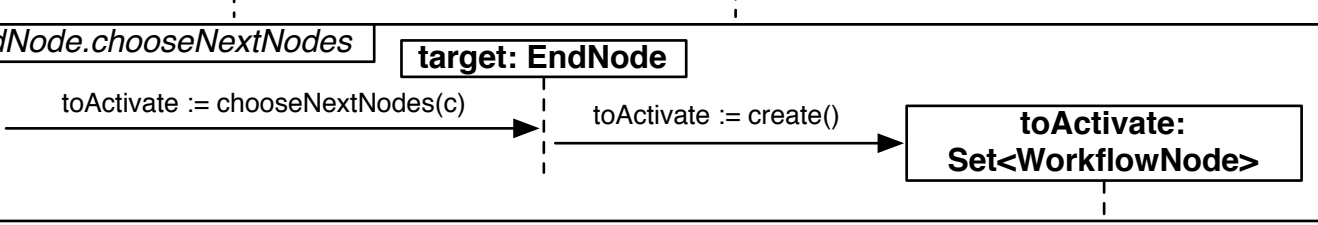


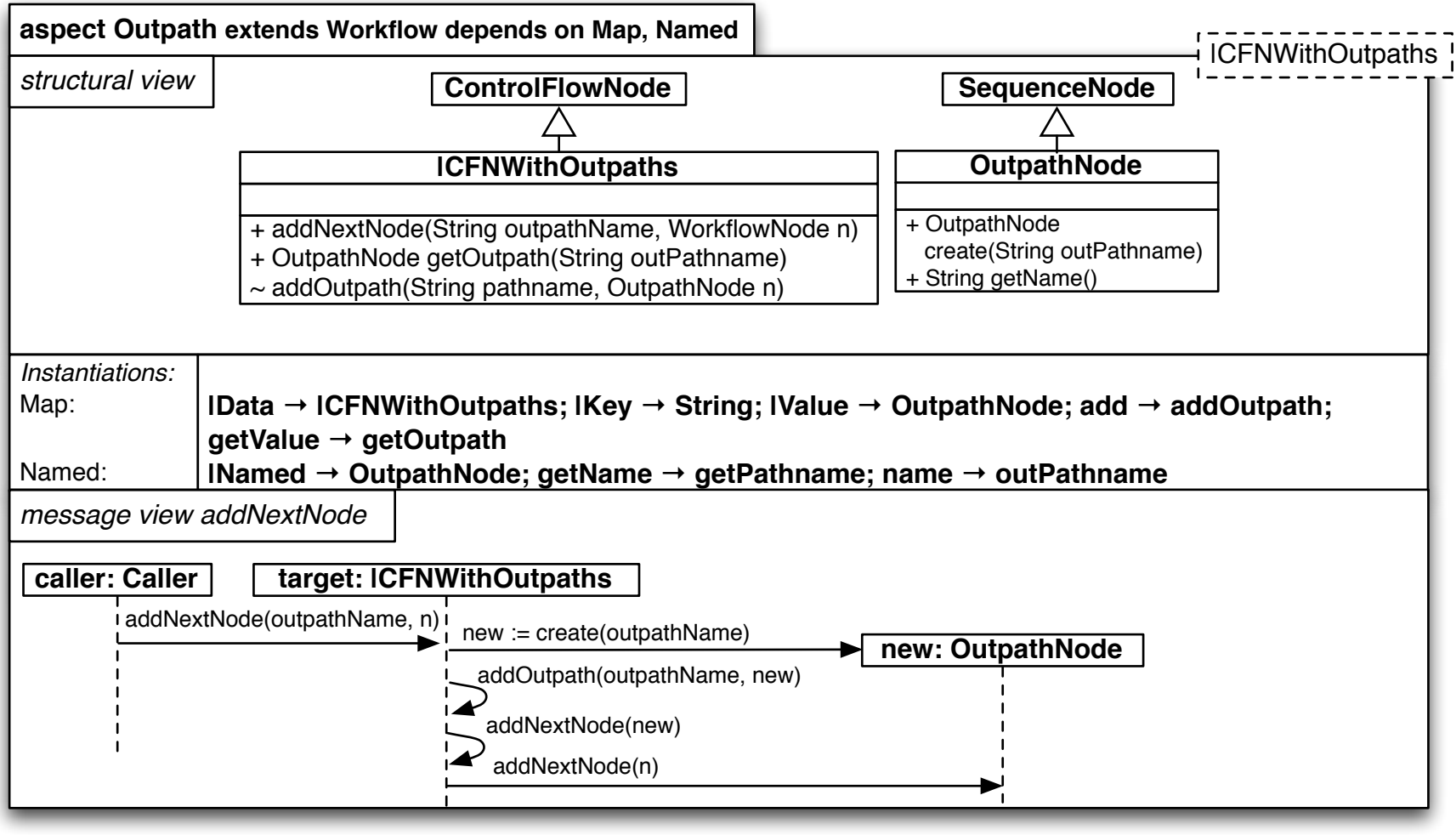
message view WorkflowNode.initialize is empty

message view launch



message view EndNode.chooseNextNodes




aspect Outpath extends Workflow depends on Map, Named
ICFNWithOutpaths
structural view
ControlFlowNode
SequenceNode
ICFNWithOutpaths
+ addNextNode(String outpathName, WorkflowNode n)
+ OutpathNode getOutpath(String outPathname)
~ addOutpath(String pathname, OutpathNode n)
OutpathNode
+ OutpathNode
create(String outPathname)
+ String getName()
Instantiations:
Map:
IData → ICFNWithOutpaths; IKey → String; IValue → OutpathNode; add → addOutpath;
getValue → getOutpath
Named:
INamed → OutpathNode; getName → getPathname; name → outPathname
message view addNextNode
caller: Caller
target: ICFNWithOutpaths
addNextNode(outpathName, n)
new := create(outpathName)
new: OutpathNode
addOutpath(outpathName, new)
addNextNode(new)
addNextNode(n)

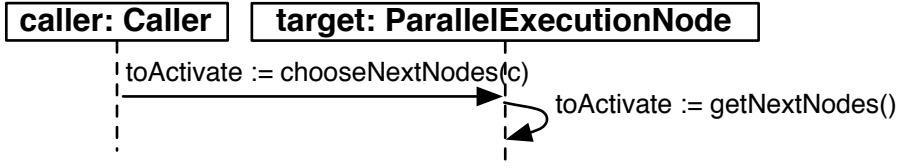
aspect ParallelExecution extends Workflow depends on Outpath

structural view

ParallelExecutionNode
Set<WorkflowNode> chooseNextNodes(LocalContext c)

Instantiations:
Outpath: **ICFNWithOutpaths → ParallelExecutionNode**

message view chooseSuccs



aspect ConditionalExecution extends Workflow depends on Outpath, Named, Map

structural view

WorkflowExecutor

+ setCurrentChosenPath (String ceNode, String chosenPath)
- setChosenOutpath(String ceNode, String outpath)
~ String getChosenOutpath(String ceNode)
~ void resetChosenOutpath(String ceNode)

ConditionalExecutionNode

Set<WorkflowNode> chooseNextNodes(LocalContext c)

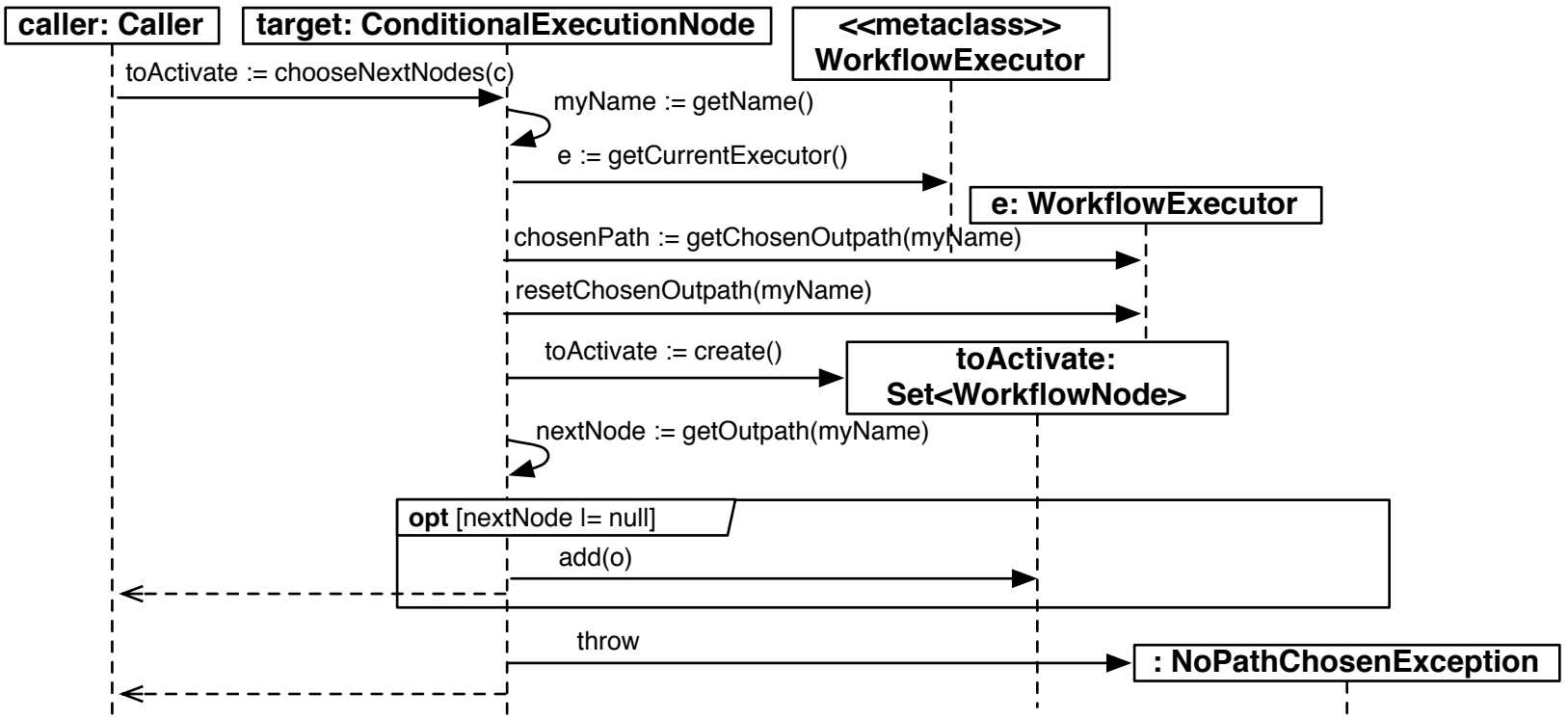
WorkflowExecutionException

NoPathChosenException

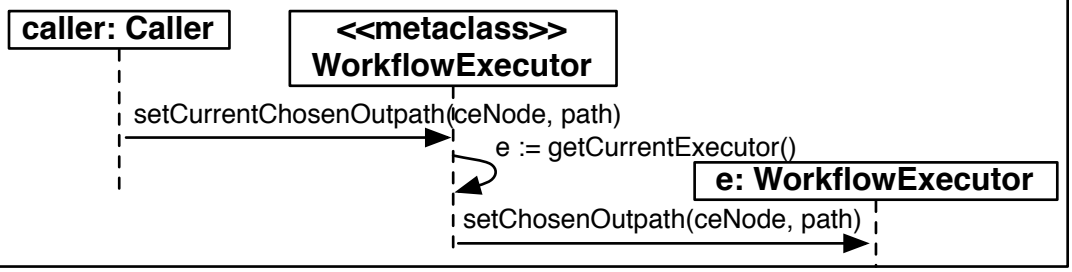
Instantiations:

Outpath: **ICFNWithOutpaths → ConditionalExecutionNode; OutpathNode → OutpathNode**
Named: **INamed → ConditionalExecutionNode**
Map: **IData → WorkflowExecutor; IKey → String; IValue → String; getValue → getChosenOutpath; add → setOutpath; remove → resetChosenOutpath;**

message view chooseNextNodes



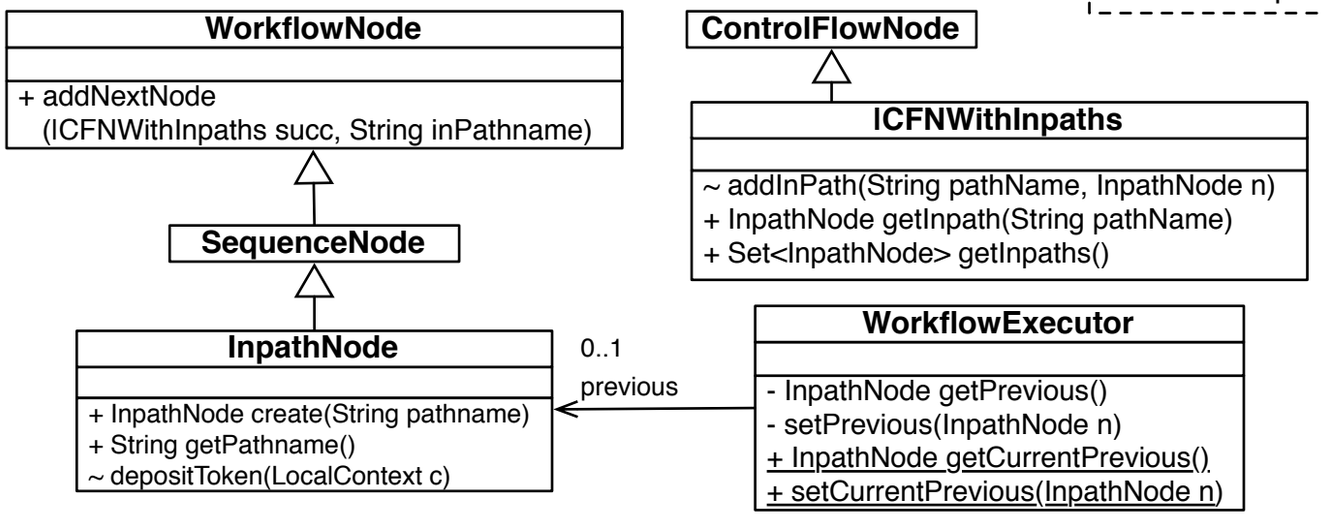
message view setCurrentChosenPath



aspect Inpath extends Workflow depends on Map, Named

ICFNWithInpaths

structural view

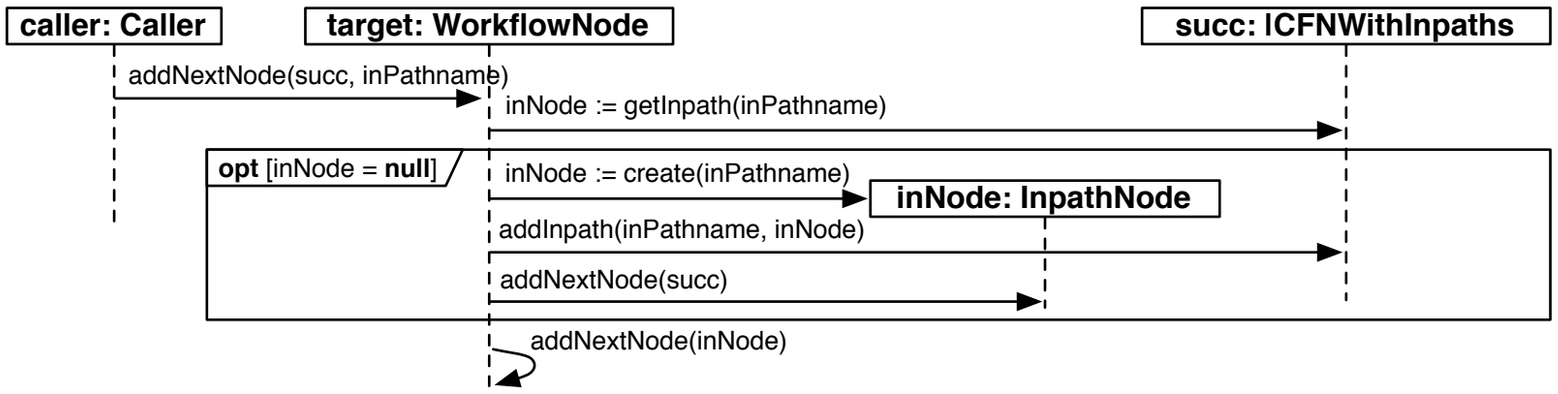


Instantiations:

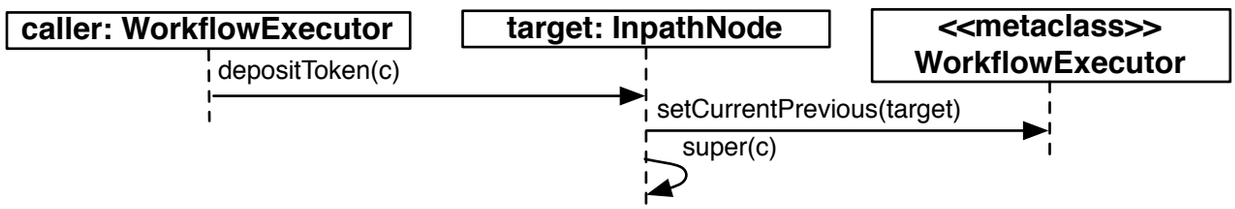
Map: **IData → ICFNWithInpaths; IKey → String; IValue → InpathNode; add → addInpath; getValue → getInpath; getValues → getInpaths**

Named: **INamed → InpathNode; getName → getPathname; name → pathname**

message view connectToInPath



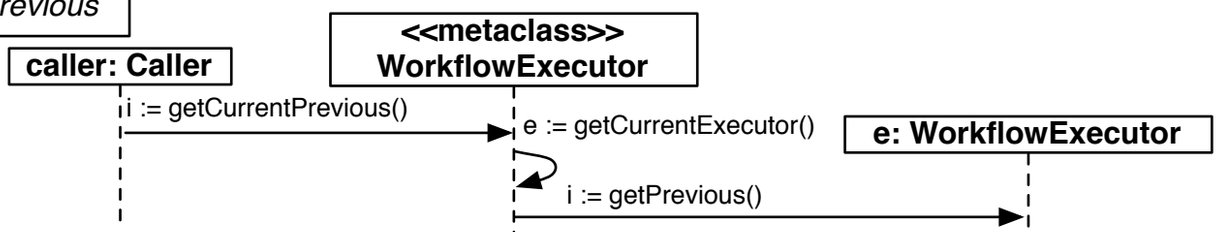
message view depositToken



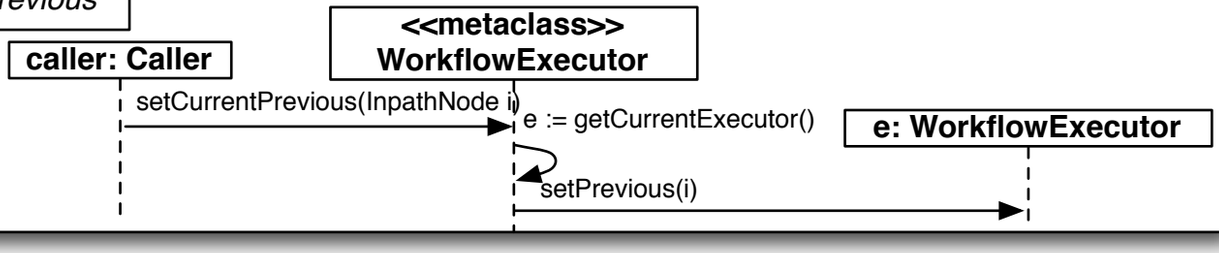
message view getPrevious is Getter<previous>

message view setPrevious is Setter<previous>

message view getCurrentPrevious



message view setCurrentPrevious



aspect Synchronization extends Workflow depends on Inpath, KeyCounter, ZeroToMany-Ordered

structural view

LocalContext
~ mergeContext(LocalContext other)

SynchronizationNode
Set<WorkflowNode> chooseNextNodes(LocalContext c) ~ initialize() - LocalContext getMergedContext(int index) - addMergedContext(int index, LocalContext c) - removeMergedContext(int index)

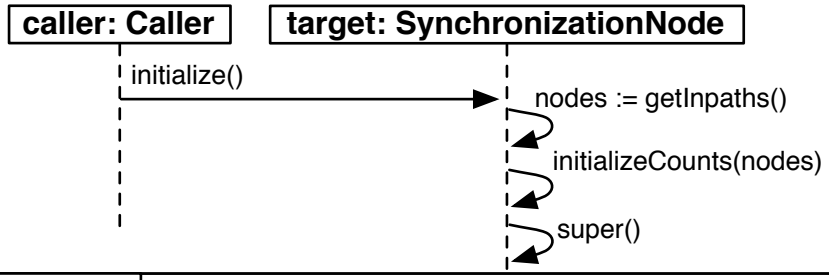
Instantiations:

Inpath: **ICFNWithInpaths → SynchronizationNode**

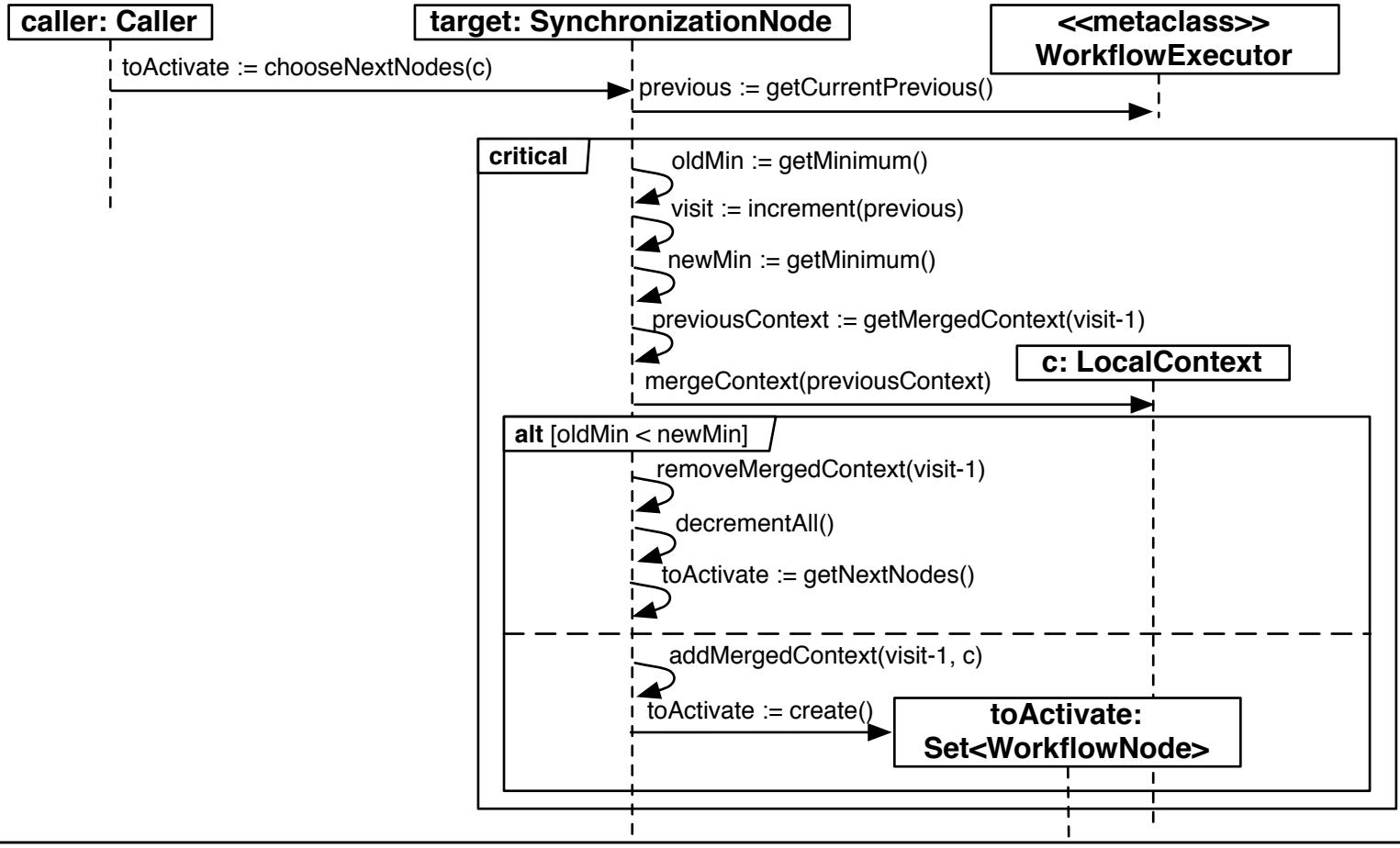
KeyCounter: **KeyCounter → SynchronizationNode; IElement → InpathNode**

ZeroToMany-Ordered: **IData → SynchronizationNode; IAssociated → LocalContext; getAssociated → getMergedContext; add → addMergedContext; remove → removeMergedContext**

message view initialize



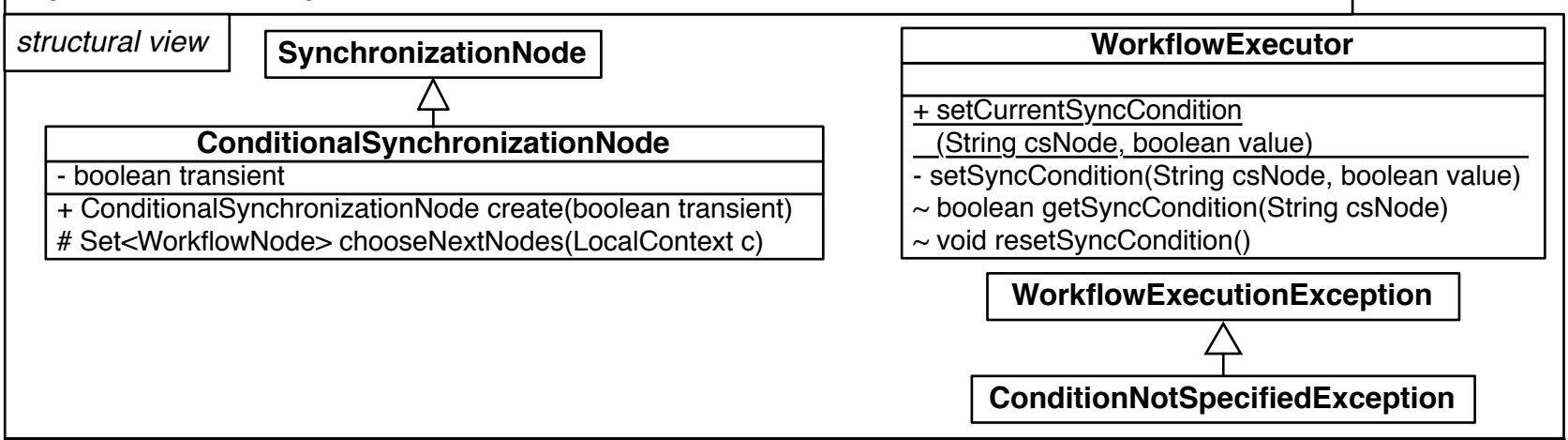
message view chooseNextNodes



message view mergeContext is empty

aspect ConditionalSynchronization extends Workflow depends on Synchronization, Named, Map

structural view



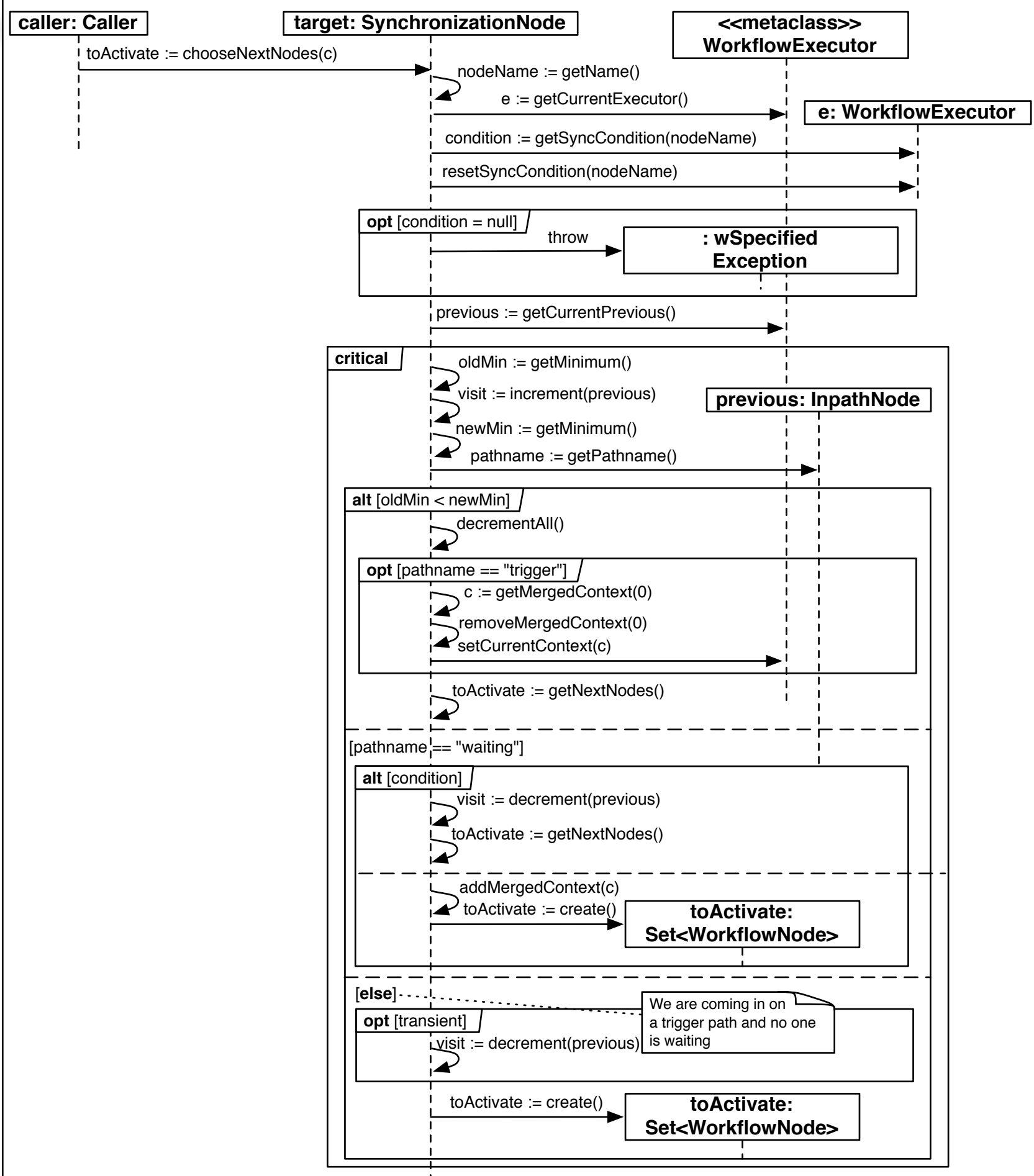
Instantiations:

Synchronization: **SynchronizationNode → SynchronizationNode**

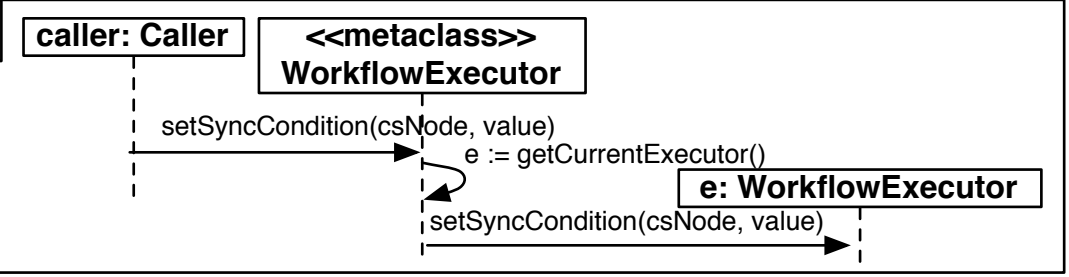
Named: **lNamed → ConditionalSynchronizationNode**

Map: **lData → WorkflowExecutor; lKey → String; lValue → Boolean; getValue → getSyncCondition; add → setSyncCondition; remove → resetSyncCondition**

message view chooseNextNodes

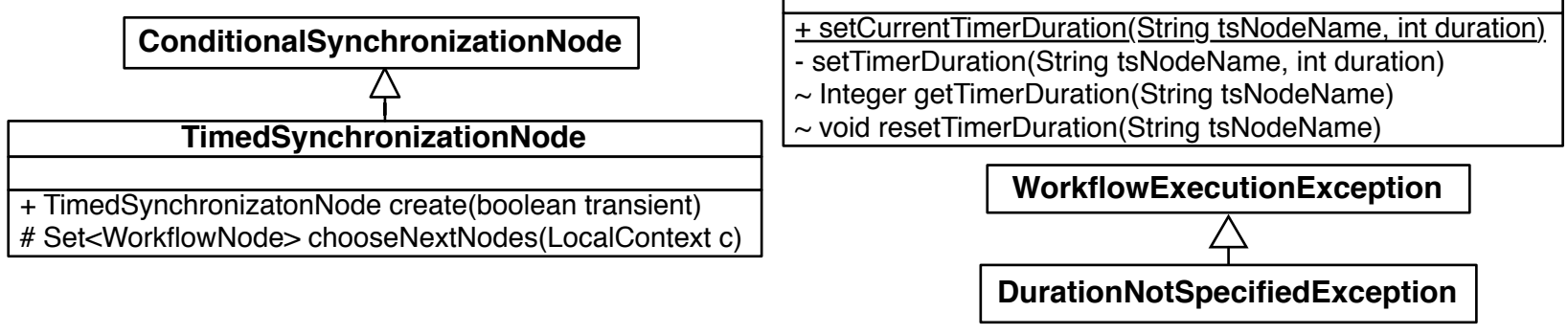


message view setCurrentSyncCondition



aspect TimedSynchronization extends Workflow
depends on ConditionalSynchronization, Outpath, ZeroToMany-Ordered, Map

structural view



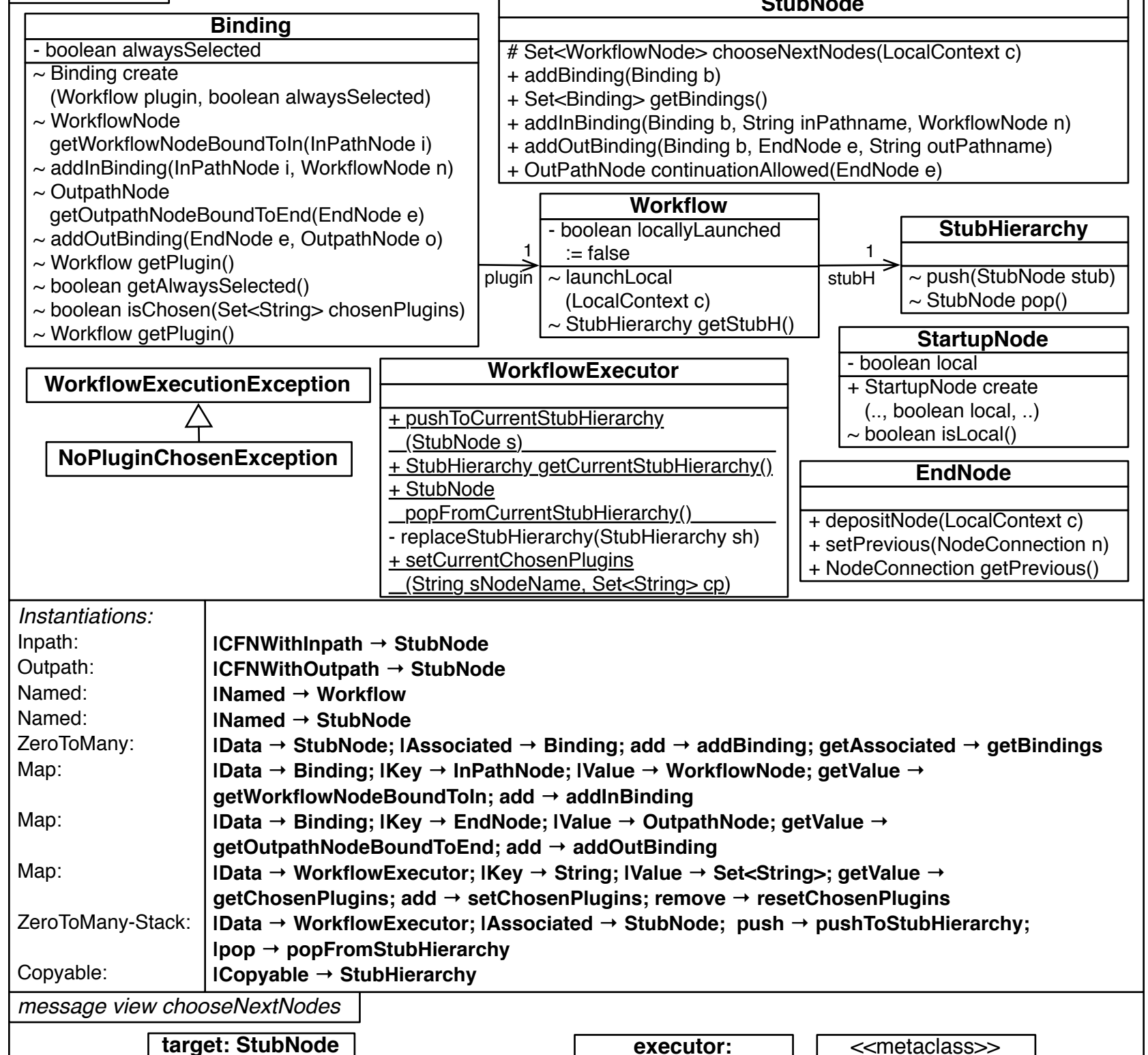
Instantiations:	
ConditionalSynchronization:	**ConditionalSynchronizationNode → TimedSynchronizationNode**
ZeroToMany-Ordered:	**IData → TimedSynchronizationNode; IAssociated → WorkflowExecutor; add → addExecutor; remove → removeExecutor; contains → containsExecutor**
Outpath:	**ICFNWithOutpaths → TimedSynchronizationNode**
Map:	**IData → WorkflowExecutor; IKey → String; IValue → Integer; getValue → getTimerDuration; add → setTimerDuration; remove → resetTimerDuration**

message view chooseNextNodes

message view setCurrentTimerCondition

aspect HierarchicalWorkflow extends Workflow
depends on Inpath, Outpath, Named, ZeroToMany, Map, ZeroToMany-Stack, Copyable

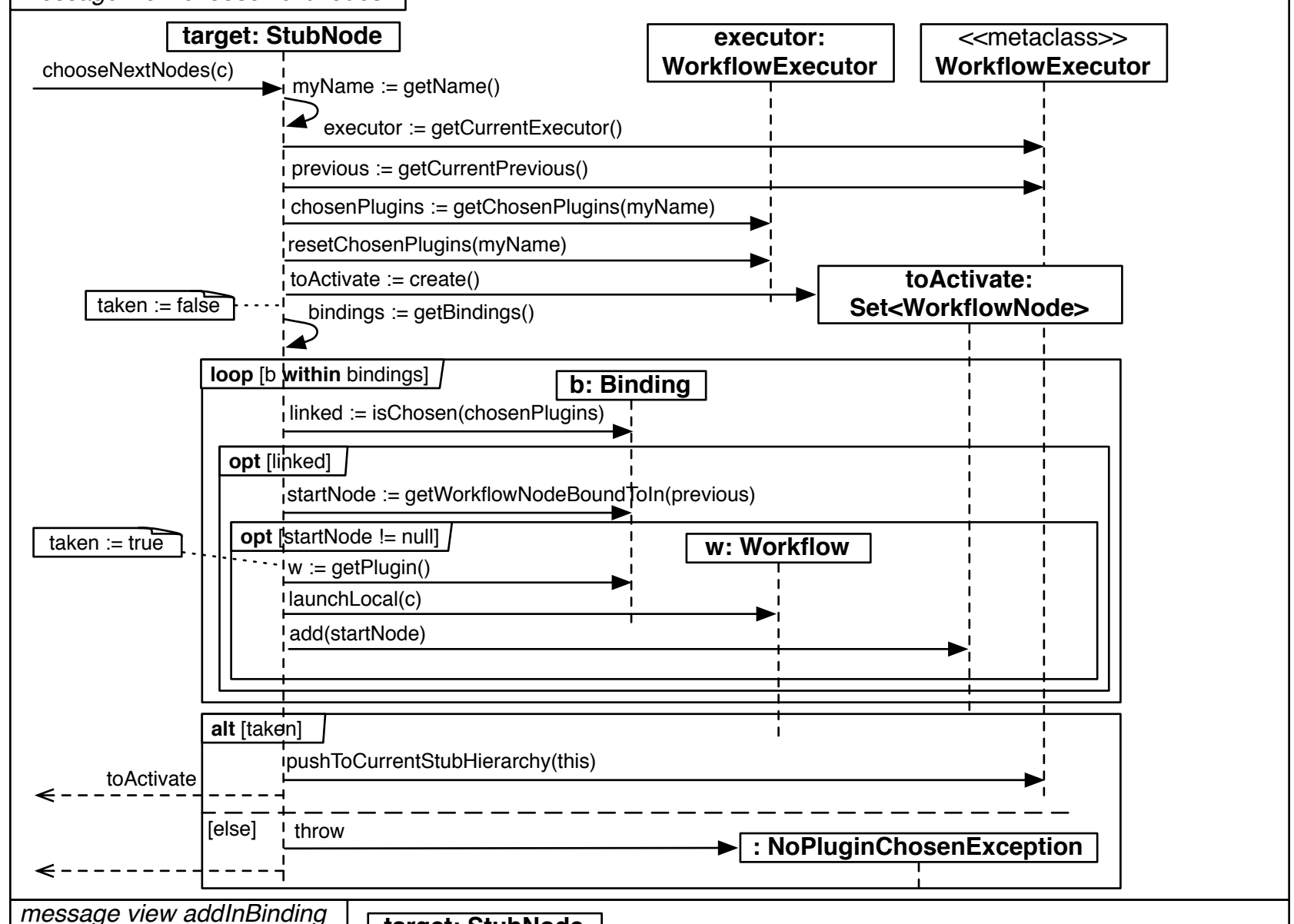
structural view



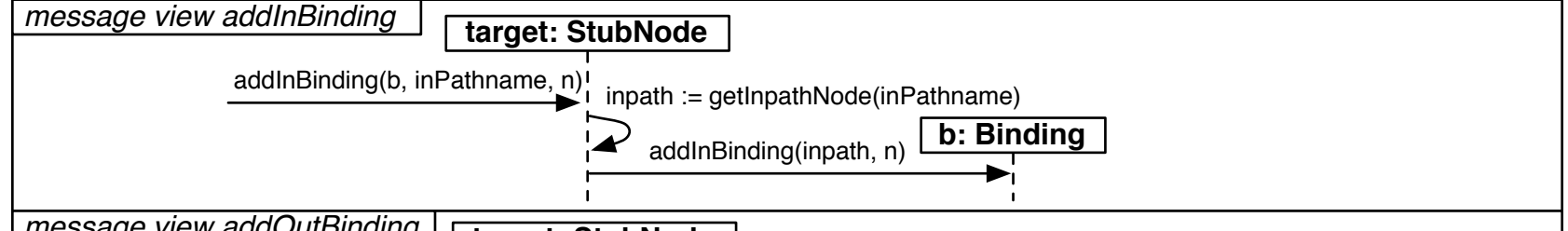
Instantiations:

Inpath:	**ICFNWithInpath → StubNode**
Outpath:	**ICFNWithOutpath → StubNode**
Named:	**INamed → Workflow**
Named:	**INamed → StubNode**
ZeroToMany:	**IData → StubNode; IAssociated → Binding; add → addBinding; getAssociated → getBindings**
Map:	**IData → Binding; IKey → InPathNode; IValue → WorkflowNode; getValue → getWorkflowNodeBoundToIn; add → addInBinding**
Map:	**IData → Binding; IKey → EndNode; IValue → OutpathNode; getValue → getOutpathNodeBoundToEnd; add → addOutBinding**
Map:	**IData → WorkflowExecutor; IKey → String; IValue → Set<String>; getValue → getChosenPlugins; add → setChosenPlugins; remove → resetChosenPlugins**
ZeroToMany-Stack:	**IData → WorkflowExecutor; IAssociated → StubNode; push → pushToStubHierarchy; Ipop → popFromStubHierarchy**
Copyable:	**ICopyable → StubHierarchy**

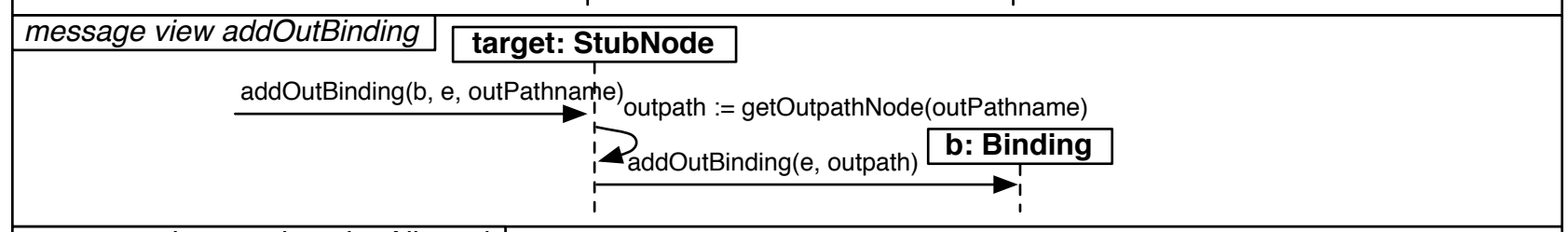
message view chooseNextNodes



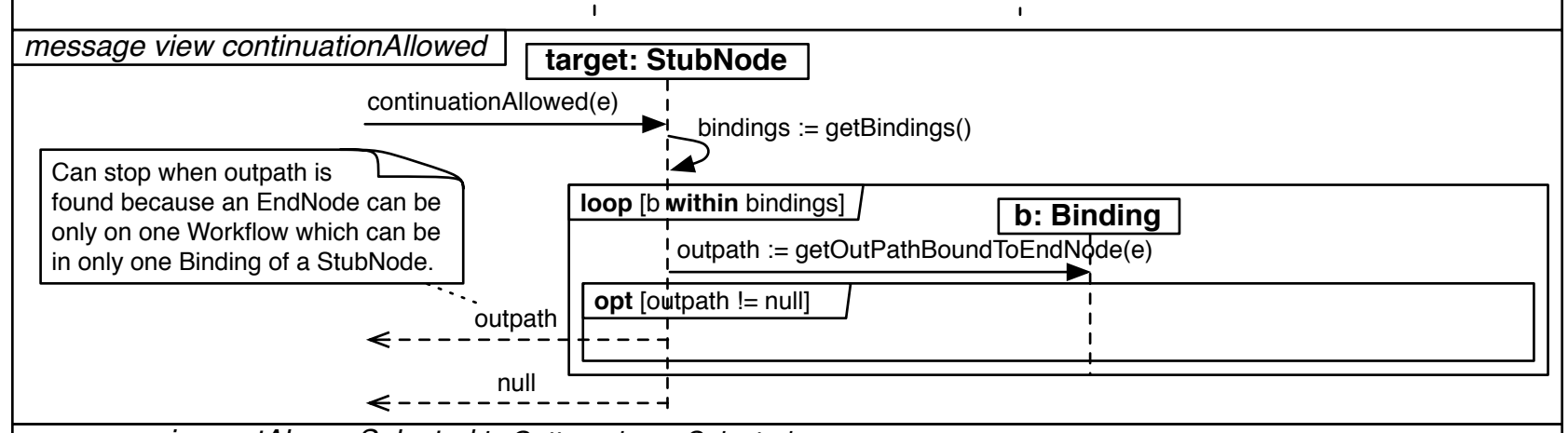
message view addInBinding



message view addOutBinding



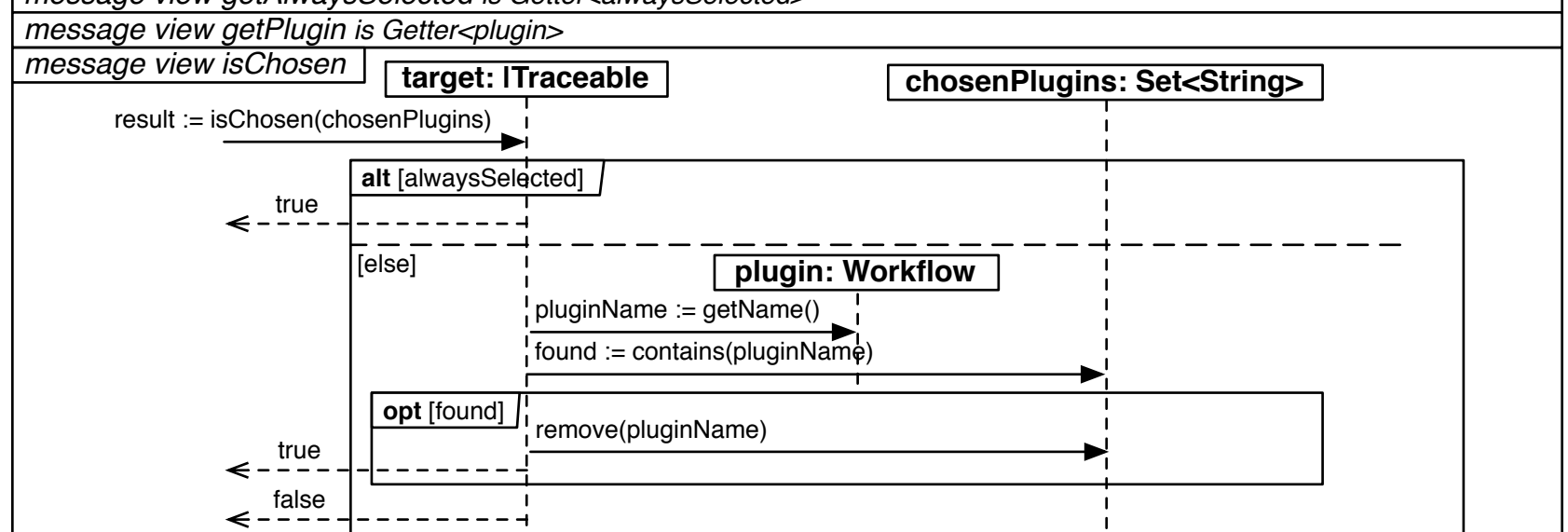
message view continuationAllowed



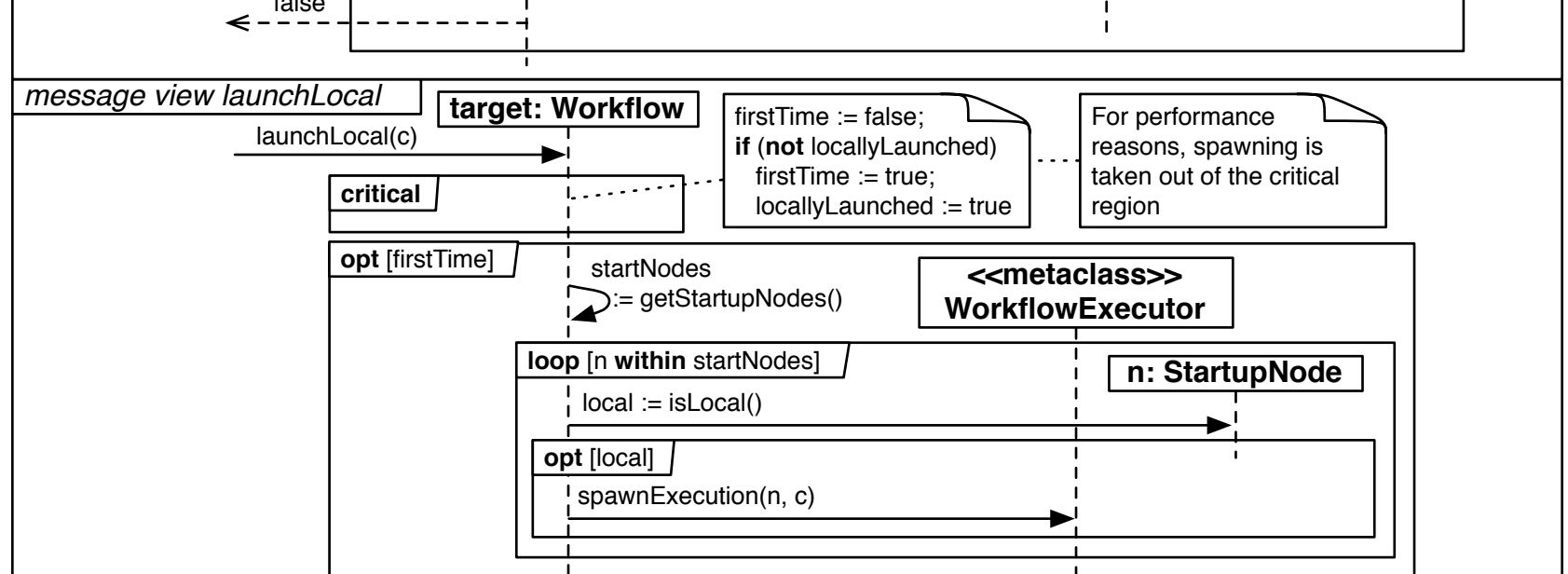
message view getAlwaysSelected is Getter<alwaysSelected>

message view getPlugin is Getter<plugin>

message view isChosen

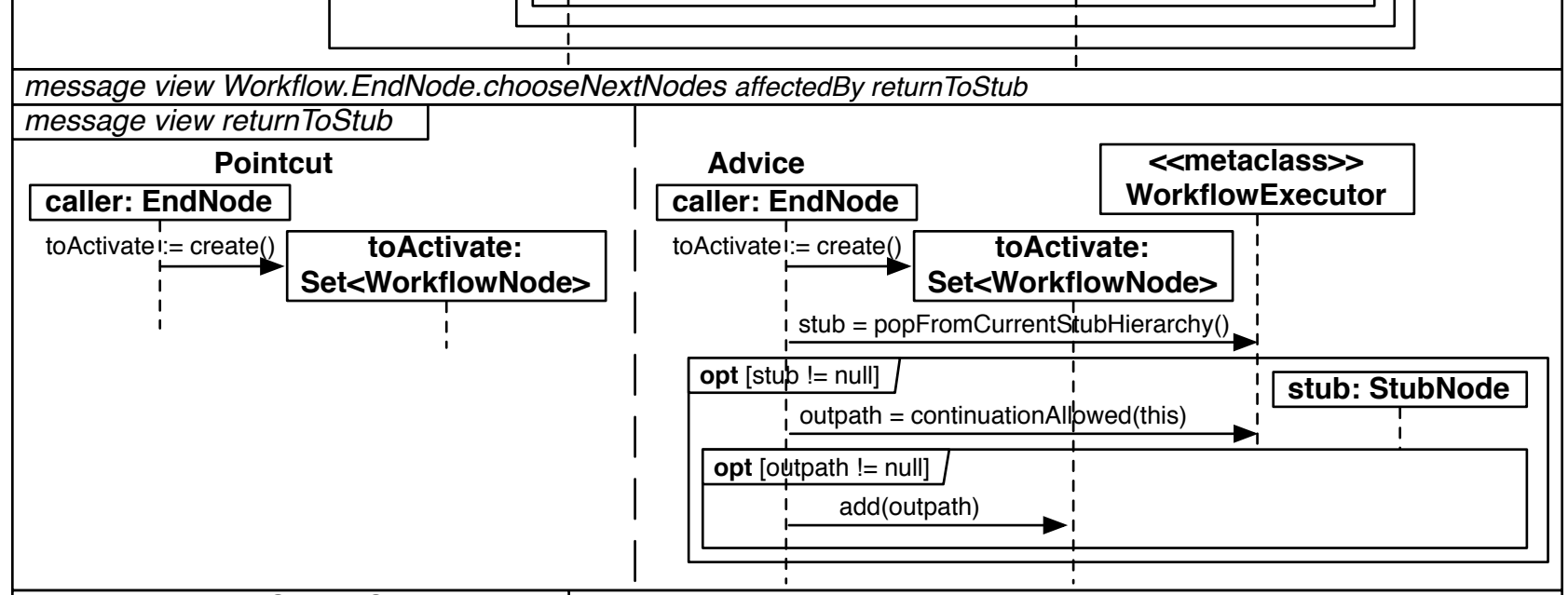


message view launchLocal

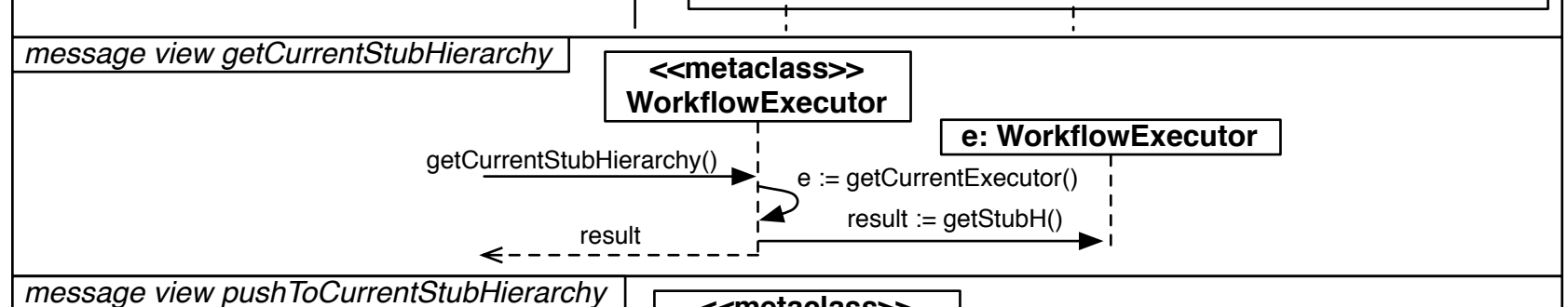


message view Workflow.EndNode.chooseNextNodes affectedBy returnToStub

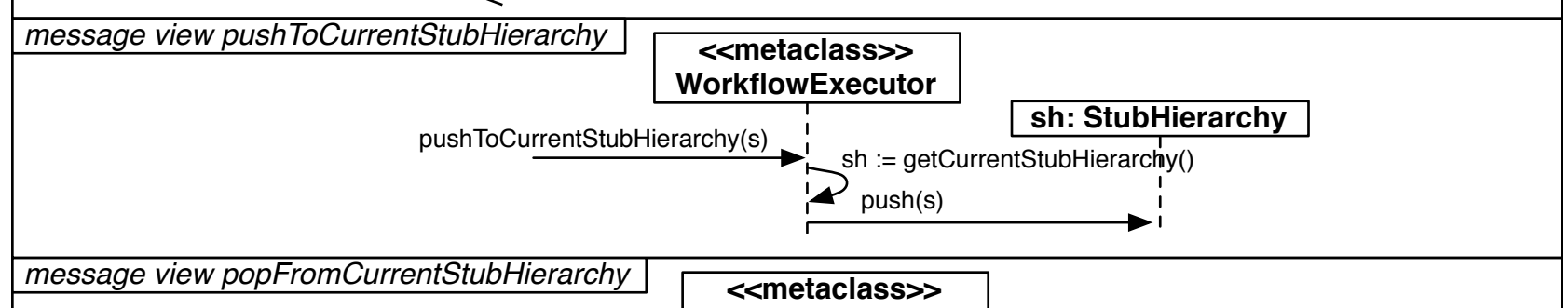
message view returnToStub



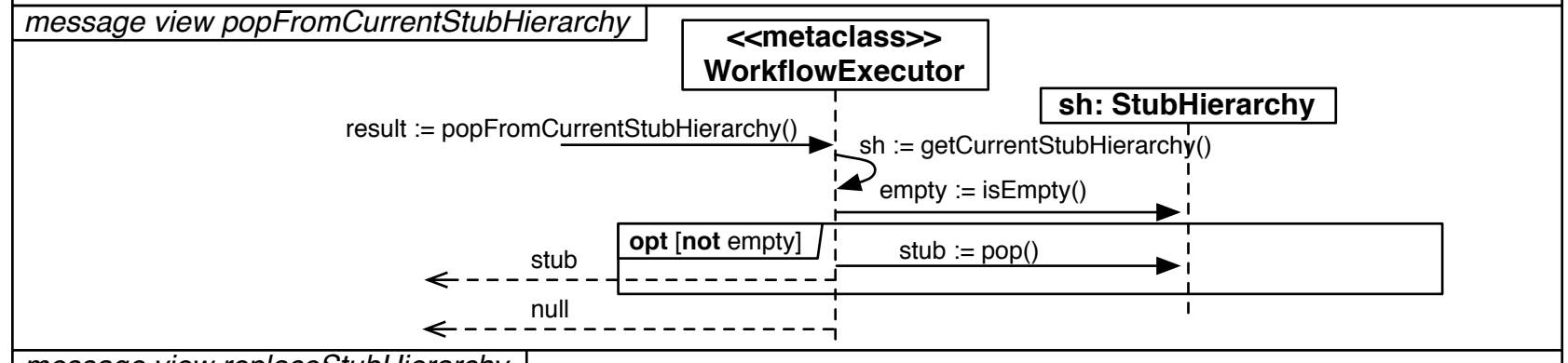
message view getCurrentStubHierarchy



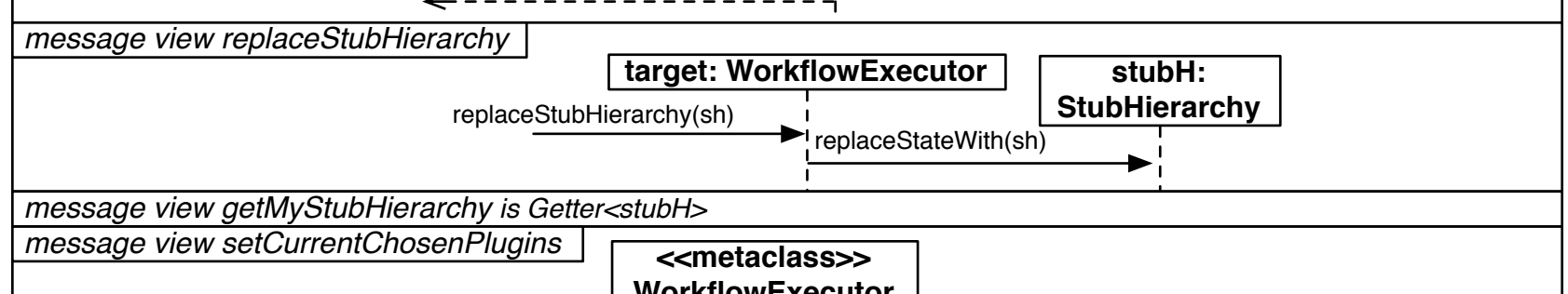
message view pushToCurrentStubHierarchy



message view popFromCurrentStubHierarchy

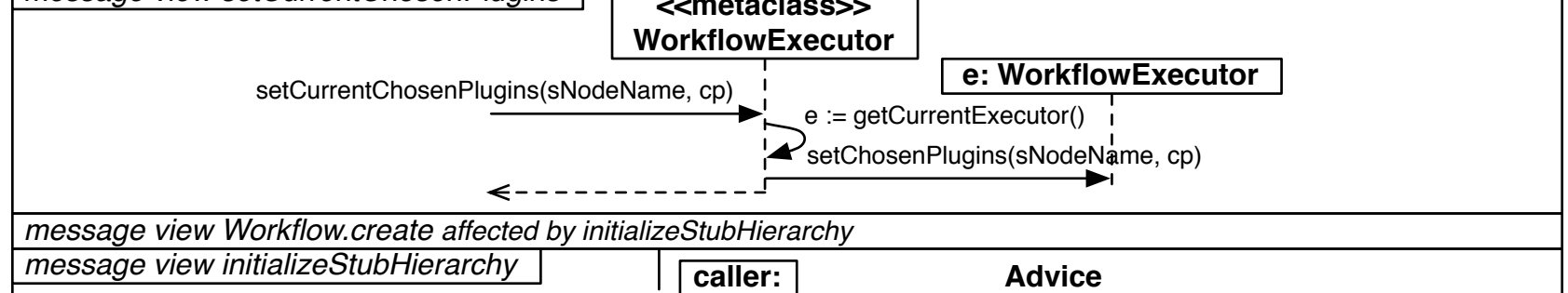


message view replaceStubHierarchy



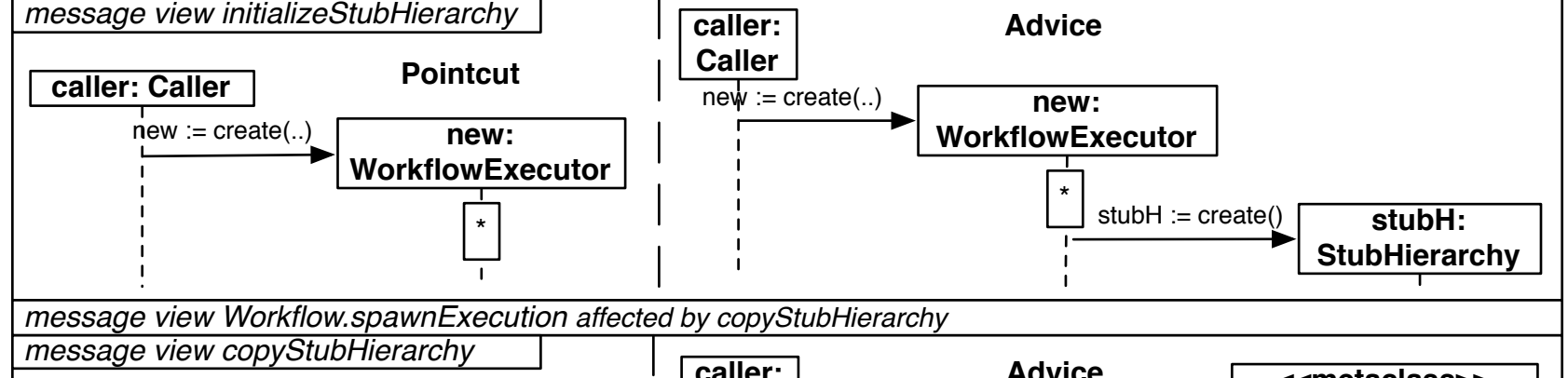
message view getMyStubHierarchy is Getter<stubH>

message view setCurrentChosenPlugins



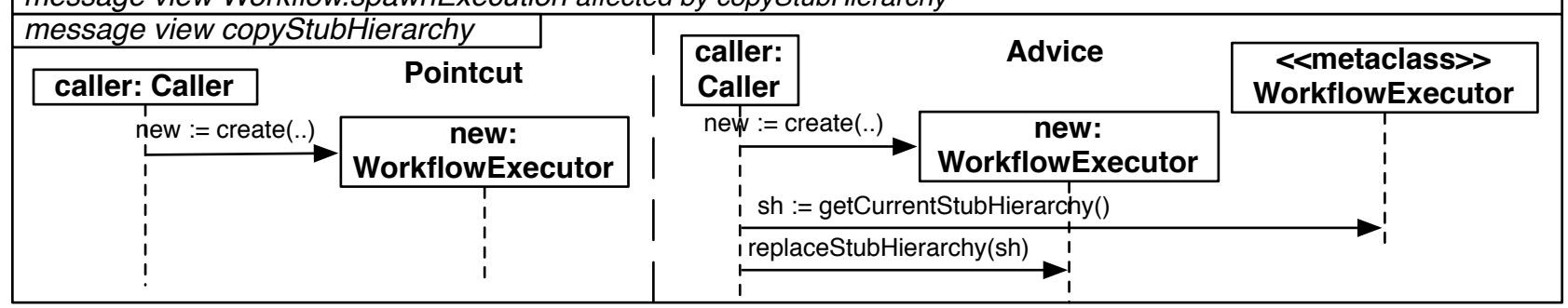
message view Workflow.create affected by initializeStubHierarchy

message view initializeStubHierarchy



message view Workflow.spawnExecution affected by copyStubHierarchy

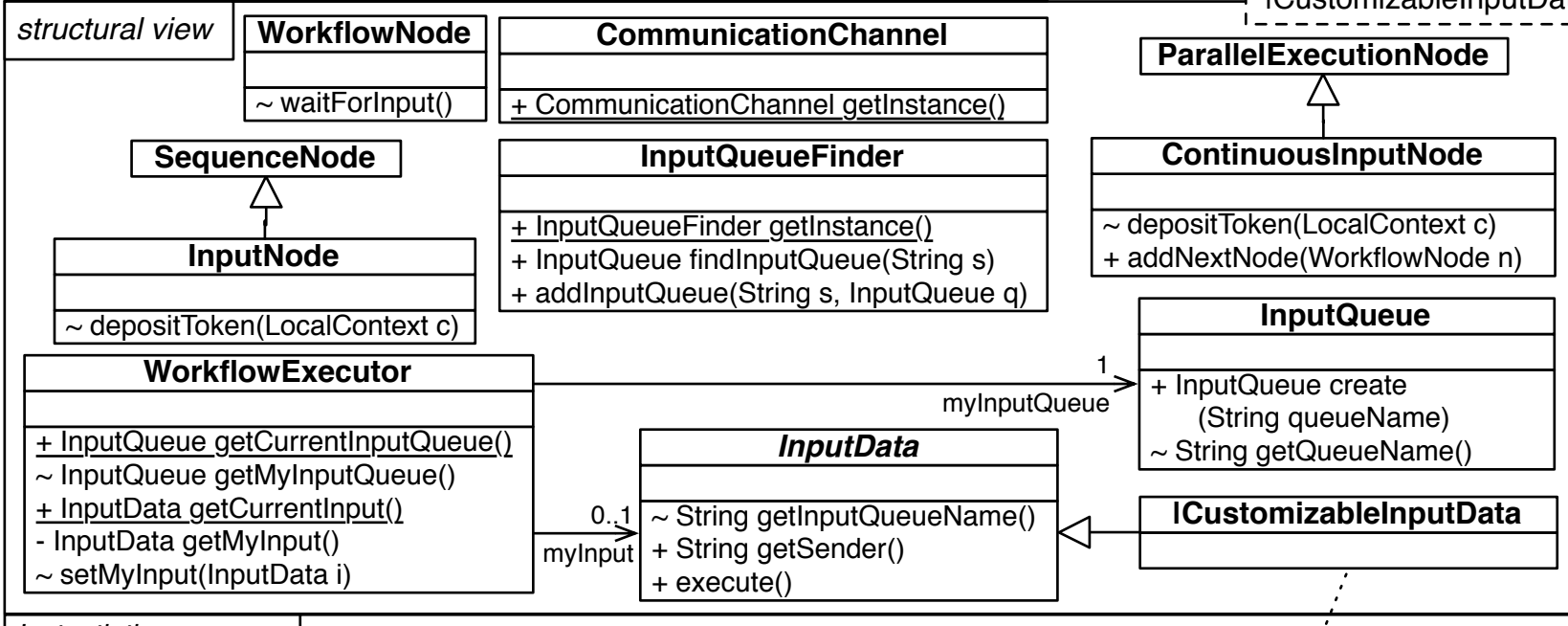
message view copyStubHierarchy



aspect Input extends Workflow
depends on ParallelExecution, NetworkCommand, Map, Named, Singleton

ICustomizableInputData

structural view



Instantiations:

ParallelExecution: **ParallelExecutionNode → ParallelExecutionNode**

NetworkCommand: **IRemoteCommand → InputData; CommandChannel → CommunicationChannel**

Map: **IData → InputQueueFinder; IKey → String; IValue → InputQueue; getValue → findInputQueue; add → addInputQueue**

This class should have a constructor, which ~~is used~~ only by the "client" node of our reactive system. We will worry about this once distribution roles are added

Singleton: **ISingleton → InputQueueFinder**

Singleton: **ISingleton → CommunicationChannel**

Named: **INamed → InputData; getName → getInputQueueName; name → inputQueueName**

Named: **INamed → InputData; getName → getSender; name → senderName**

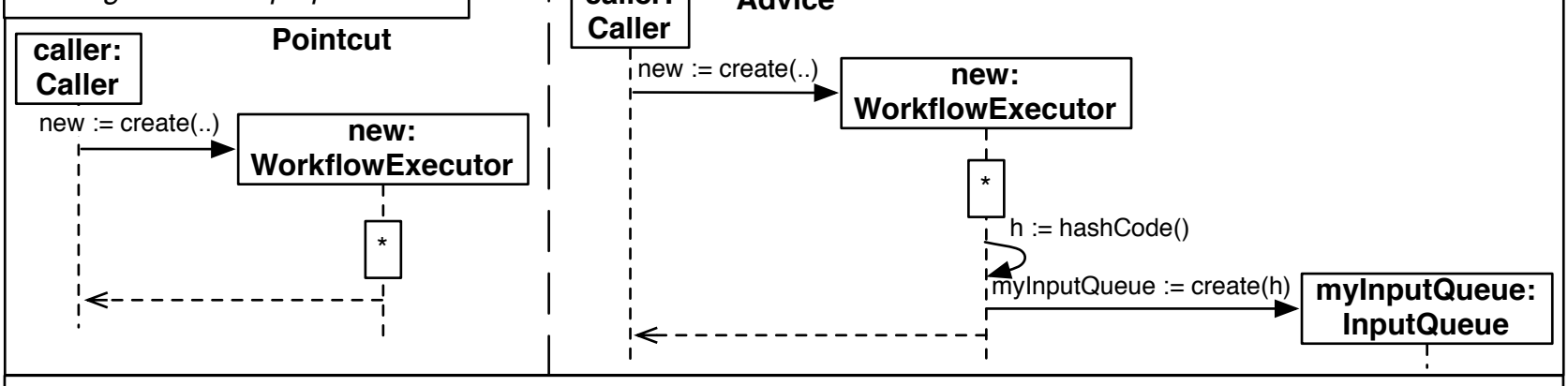
Named: **INamed → InputQueue; getName → getQueueName; name → queueName**

Implementation:

BlockingQueue<InputData>: java.util.concurrent.BlockingQueue<InputData>

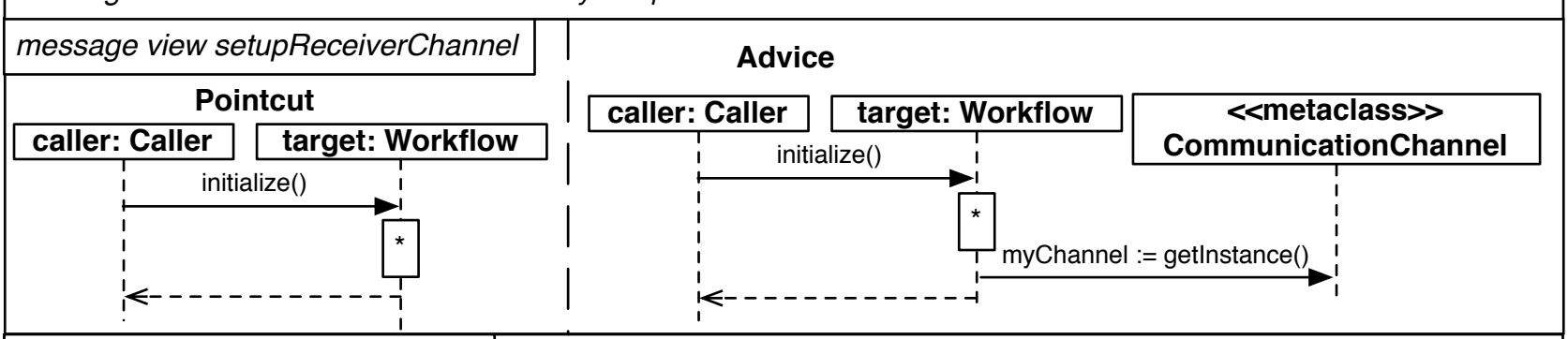
message view WorkflowExecutor.create affected by setupInputQueue

message view setupInputQueue

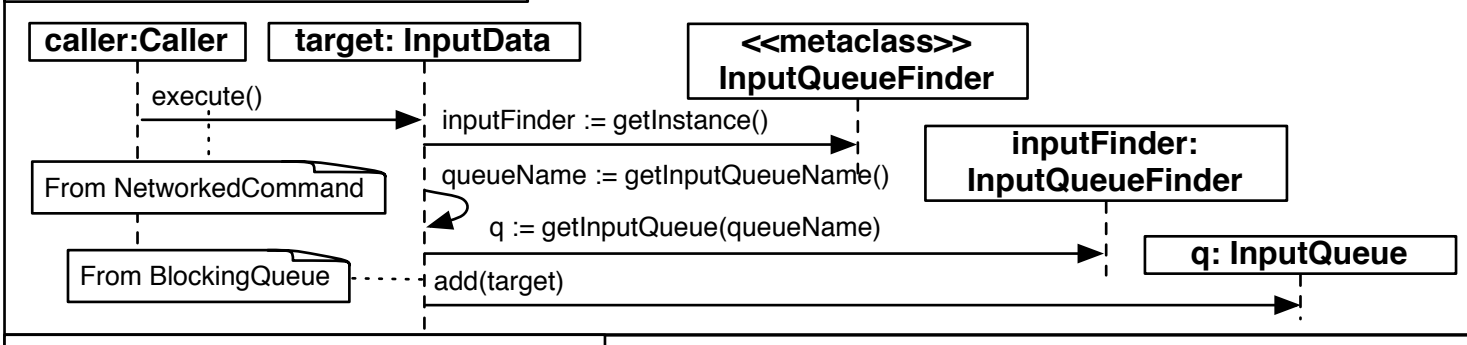


message view Workflow.initialize affected by setupReceiverChannel

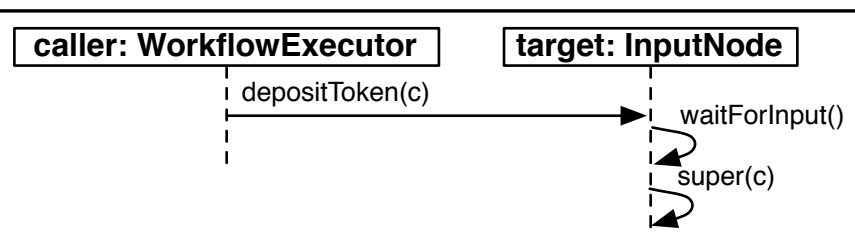
message view setupReceiverChannel



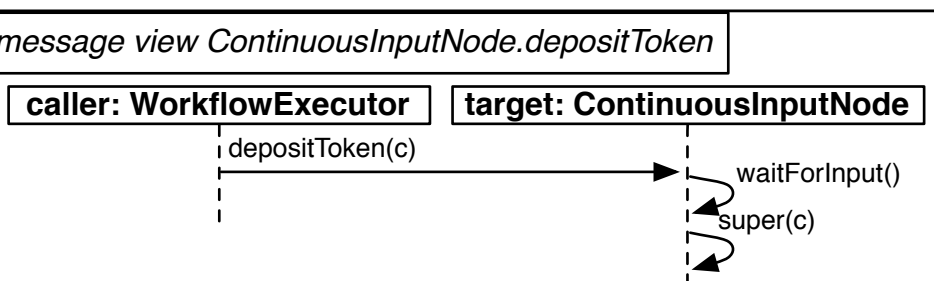
message view InputData.execute



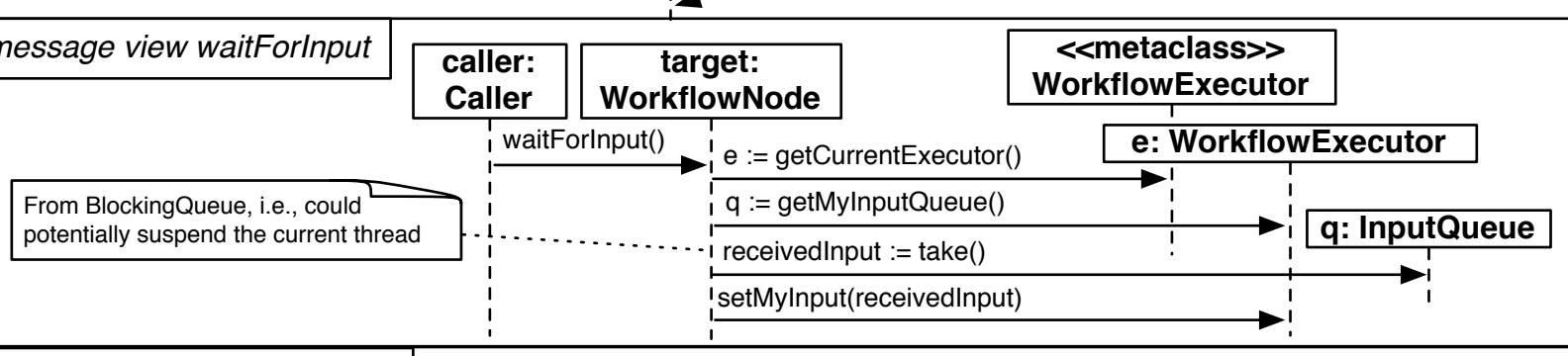
message view InputNode.depositToken



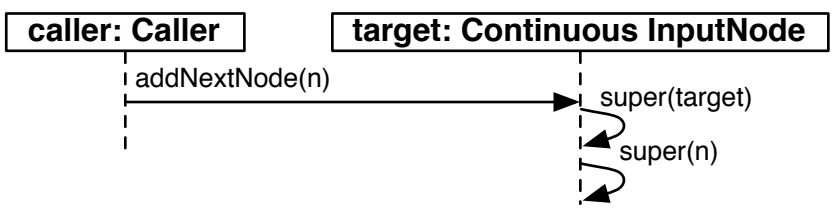
message view ContinuousInputNode.depositToken



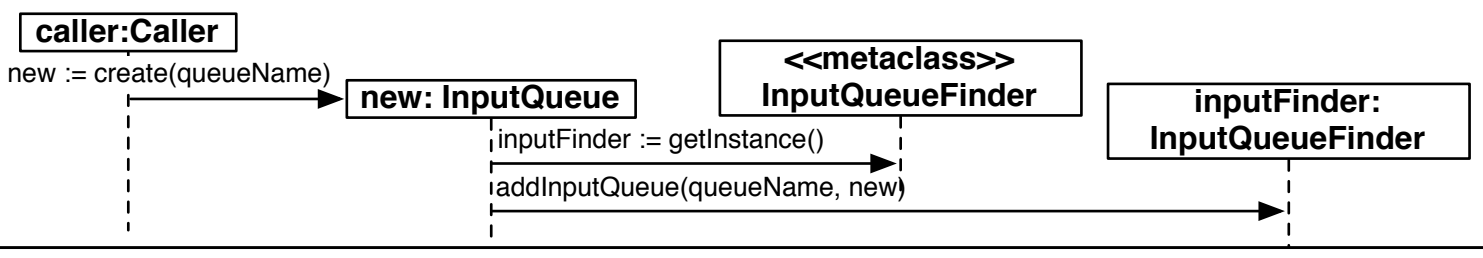
message view waitForInput



message view addNextNode



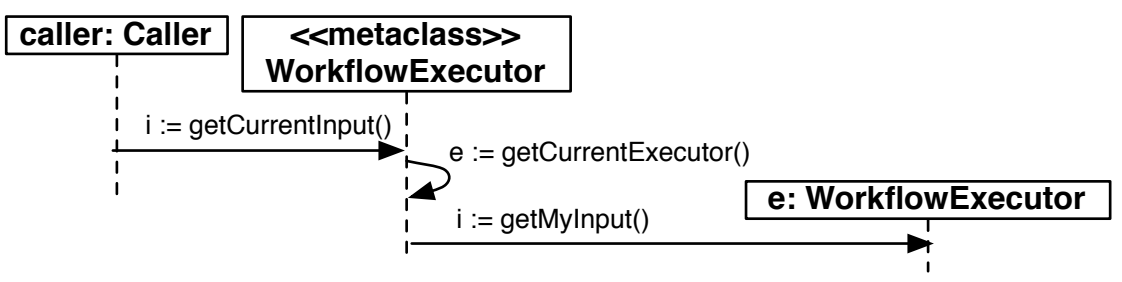
message view InputQueue.create



message view setMyInput is Setter<myInput>

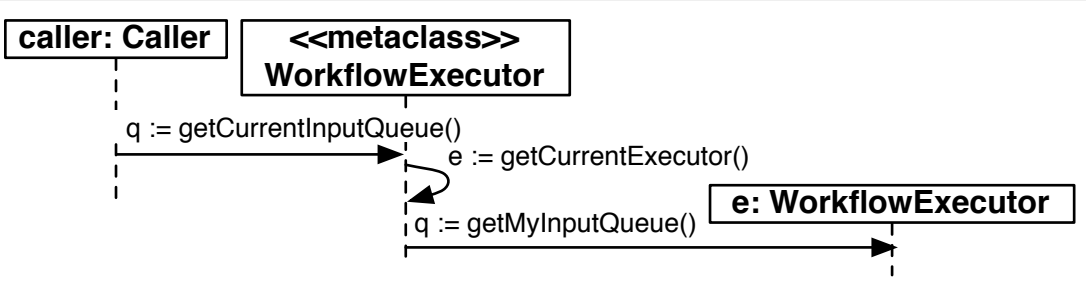
message view getMyInput is Getter<myInput>

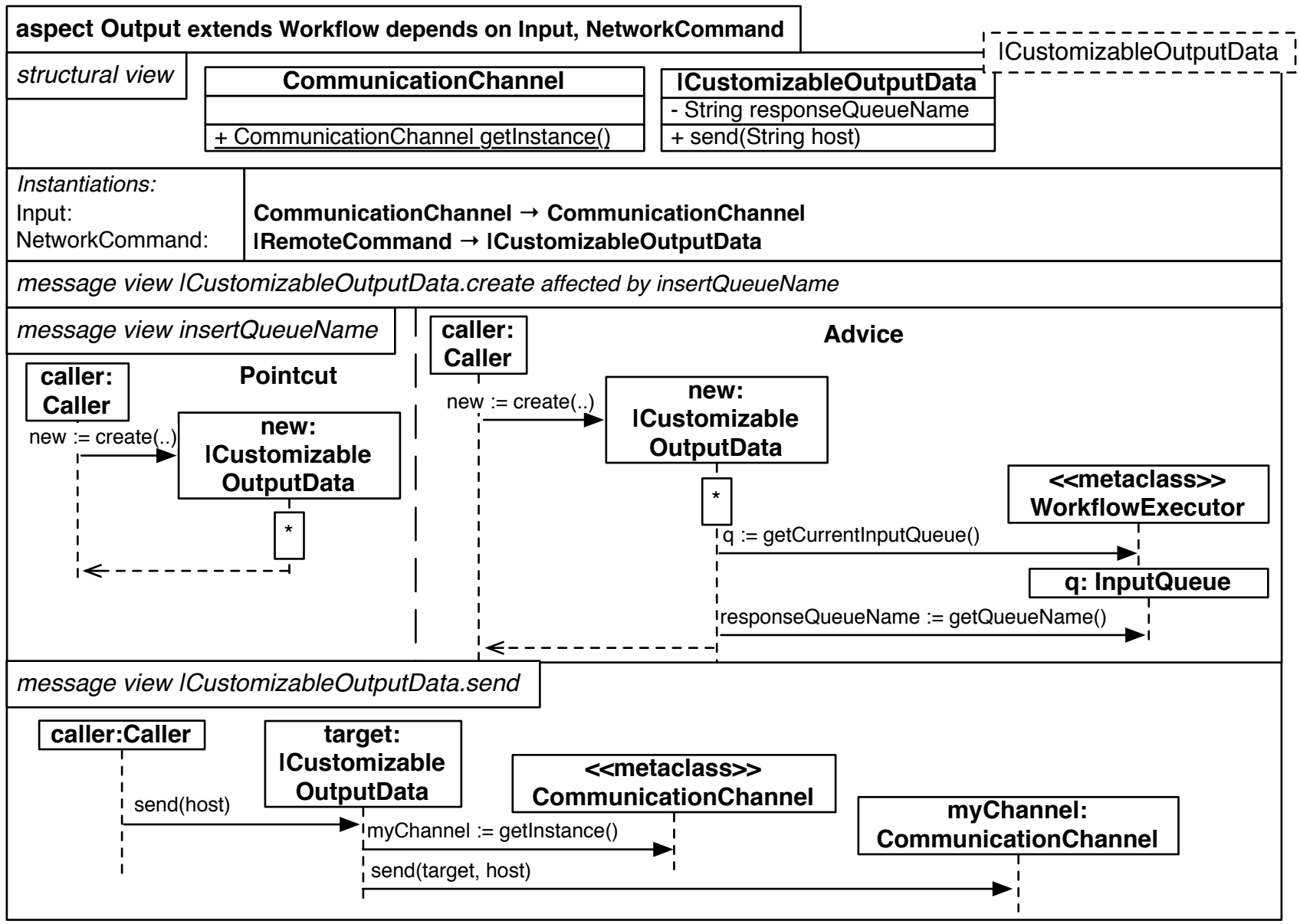
message view getCurrentInput



message view getMyInputQueue is Getter<MyInputQueue>

message view getCurrentInputQueue




aspect Output extends Workflow depends on Input, NetworkCommand
ICustomizableOutputData
structural view
CommunicationChannel
+ CommunicationChannel getInstance()
ICustomizableOutputData
- String responseQueueName
+ send(String host)
Instantiations:
Input:
NetworkCommand:
CommunicationChannel → CommunicationChannel
IRemoteCommand → ICustomizableOutputData
message view ICustomizableOutputData.create affected by insertQueueName
message view insertQueueName
caller: Caller
Pointcut
new := create(..)
new: ICustomizable OutputData
*
caller: Caller
Advice
new := create(..)
new: ICustomizable OutputData
*
<<metaclass>> WorkflowExecutor
q := getCurrentInputQueue()
q: InputQueue
responseQueueName := getQueueName()
message view ICustomizableOutputData.send
caller:Caller
target: ICustomizable OutputData
<<metaclass>> CommunicationChannel
myChannel: CommunicationChannel
send(host)
myChannel := getInstance()
send(target, host)

aspect CountingInWorkflows depends on Map, KeyCounter, Workflow

ICounterInWorkflow
IOccurrence

structural view

ICounterInWorkflow

+ count(IOccurrence o)
+ int getMinimum()
+ int getMaximum()

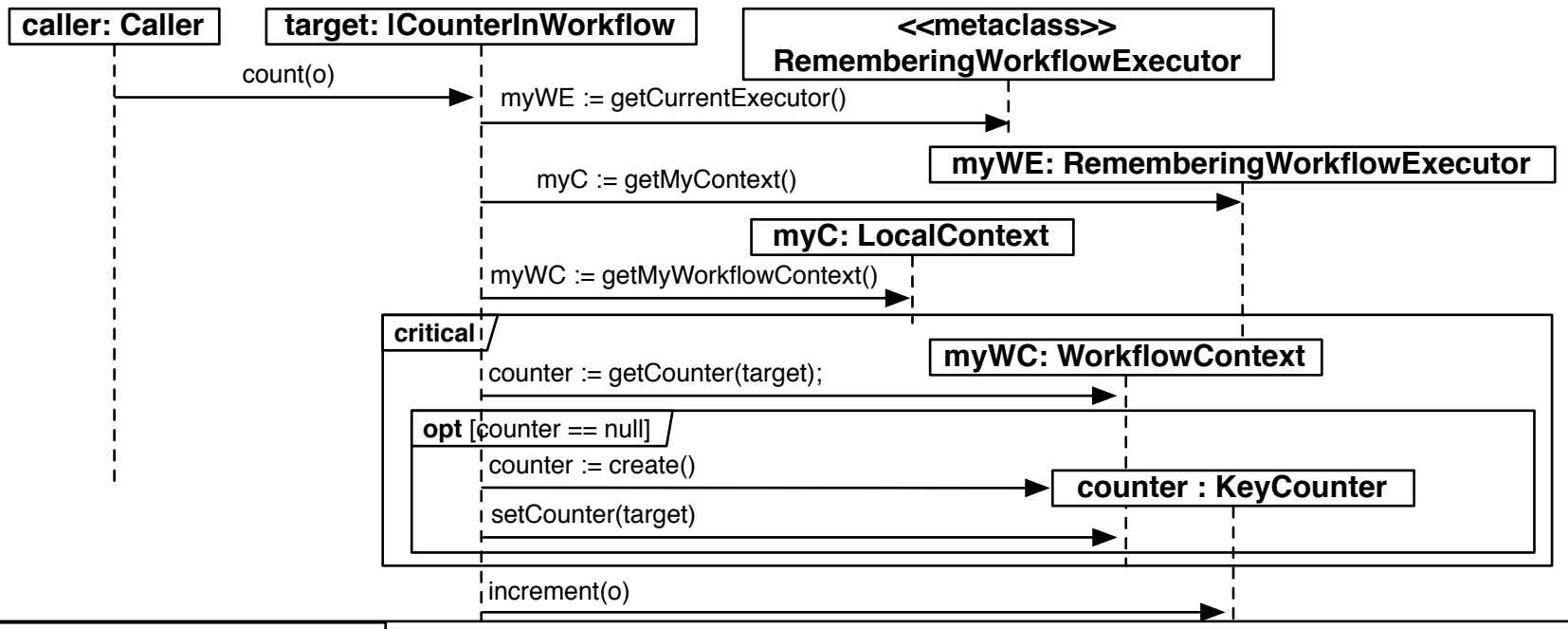
Instantiations:

Map: **IData → WorkflowContext; IKey → ICounterInWorkflow; IValue → KeyCounter; add → setCounter; getValue → getCounter**

KeyCounter: **IElement → IOccurrence;**

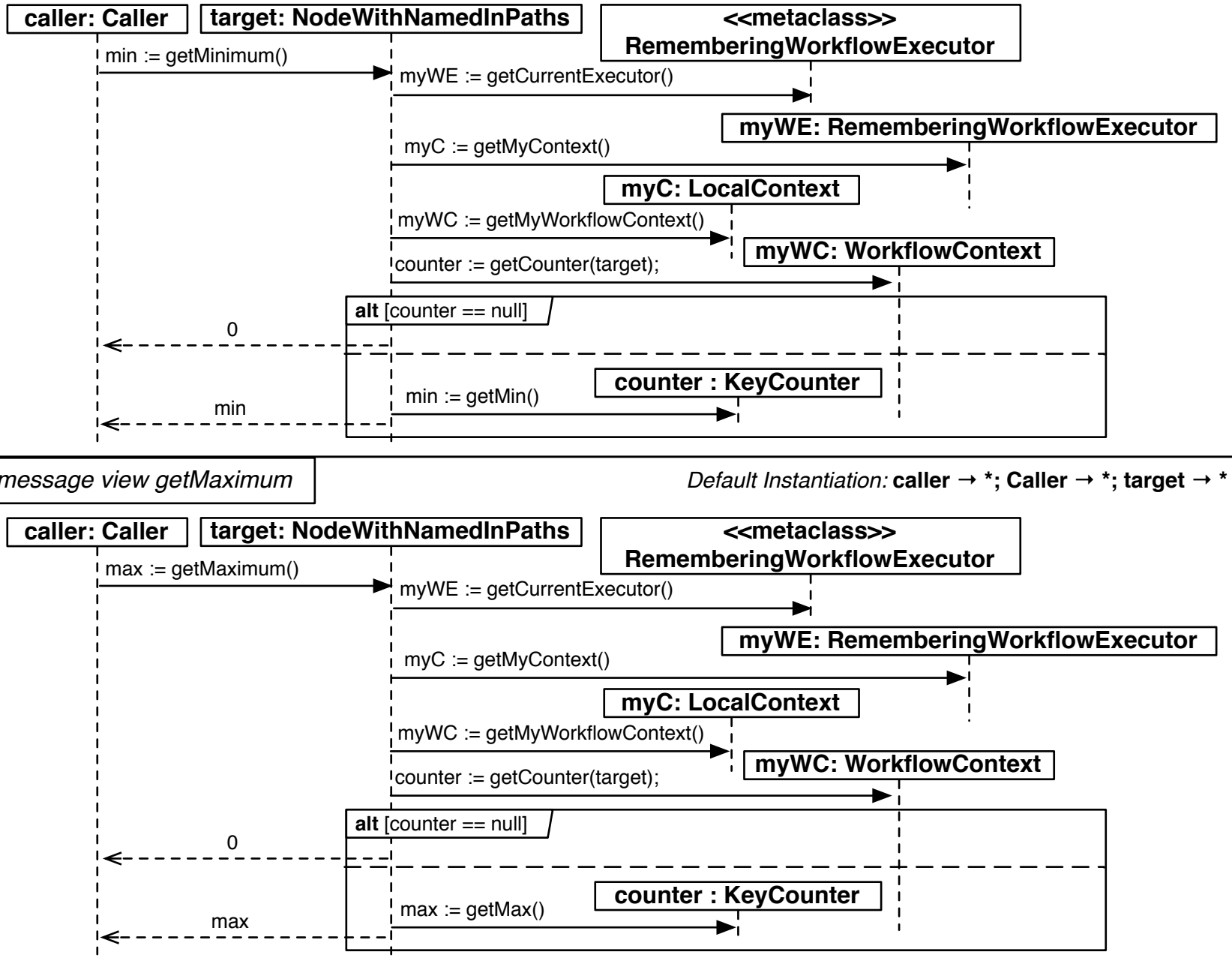
message view countIncoming

Default Instantiation: **caller → *; Caller → *; target → ***



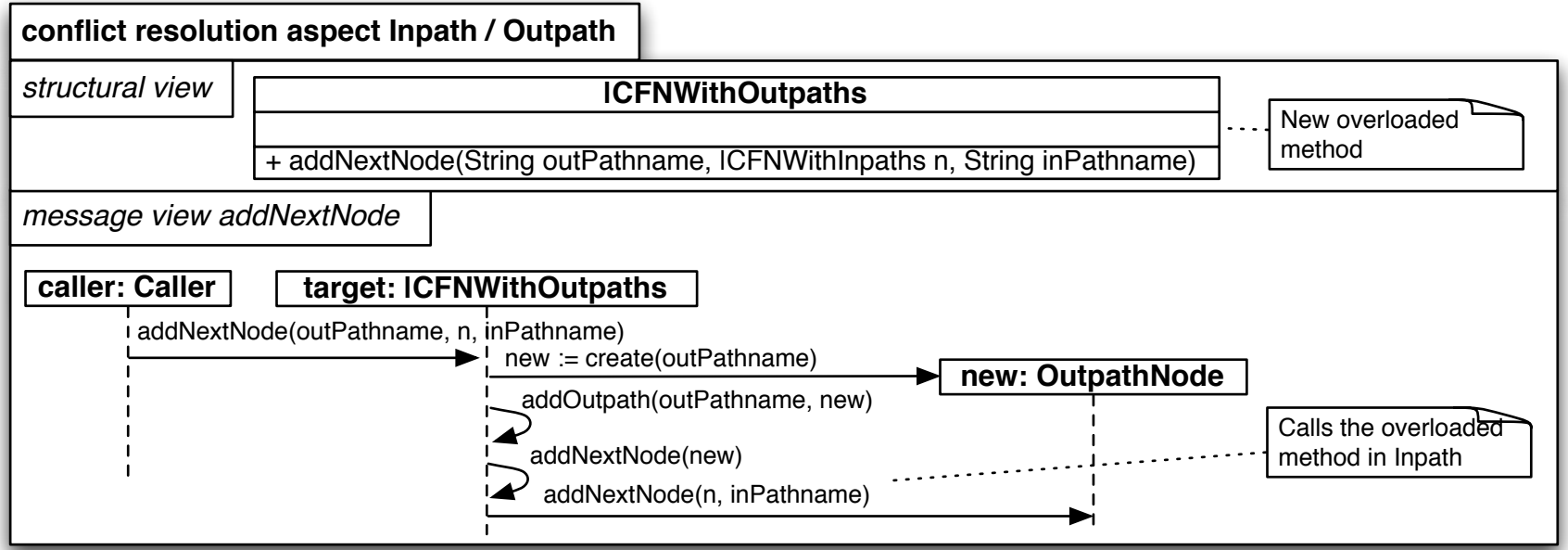
message view getMinimum

Default Instantiation: **caller → *; Caller → *; target → ***



message view getMaximum

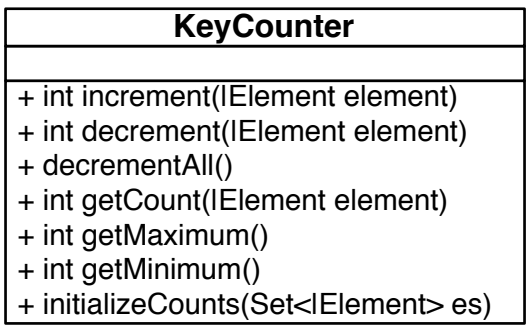
Default Instantiation: **caller → *; Caller → *; target → ***

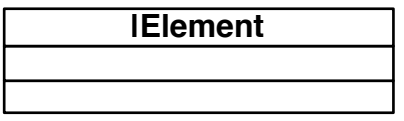

conflict resolution aspect Inpath / Outpath
structural view
ICFNWithOutpaths
+ addNextNode(String outPathname, ICFNWithInpaths n, String inPathname)
New overloaded method
message view addNextNode
caller: Caller
target: ICFNWithOutpaths
addNextNode(outPathname, n, inPathname)
new := create(outPathname)
new: OutpathNode
addOutpath(outPathname, new)
addNextNode(new)
addNextNode(n, inPathname)
Calls the overloaded method in Inpath

aspect KeyCounter depends on Map

IElement

structural view

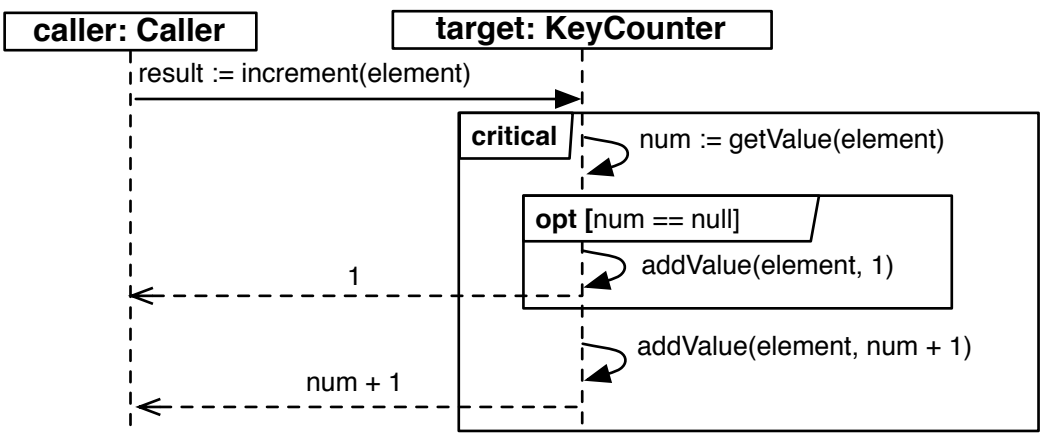




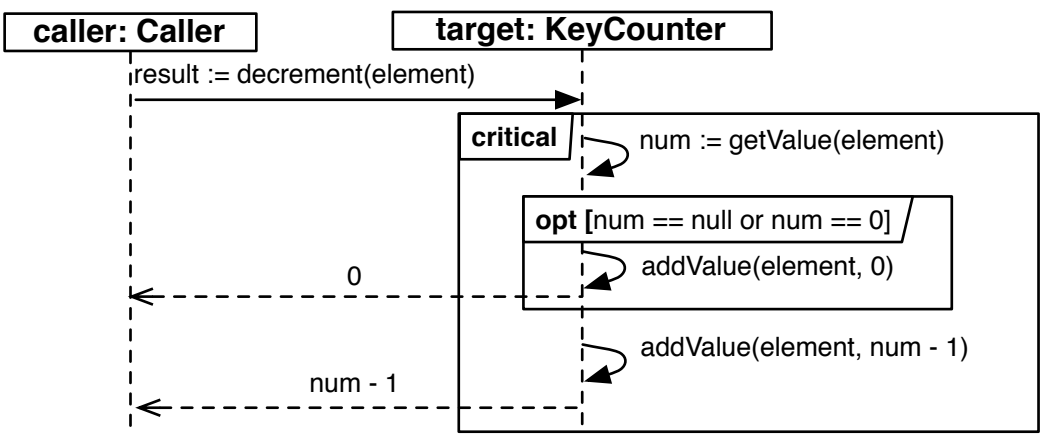
Implementation:
Math: java.lang.Math

Instantiations:
Map: **IData → KeyCounter; IKey → IElement; IValue → Integer; add → addValue**

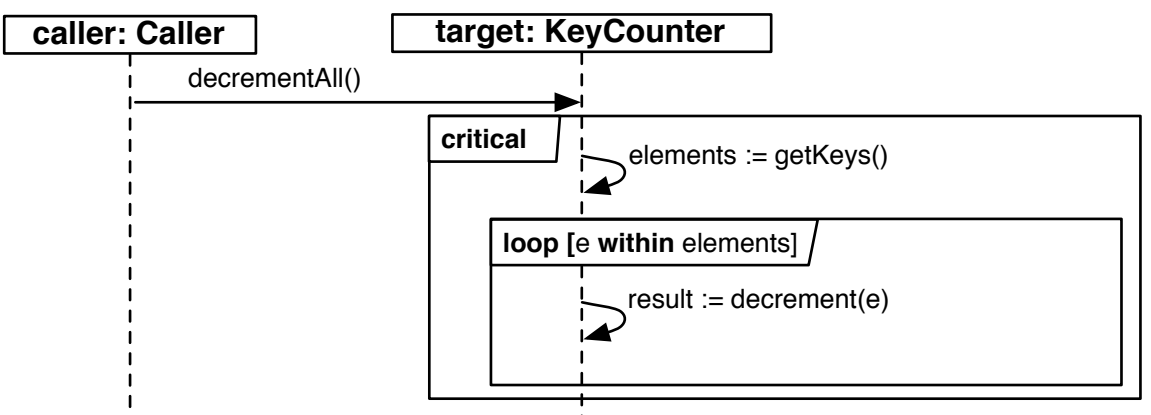
message view increment



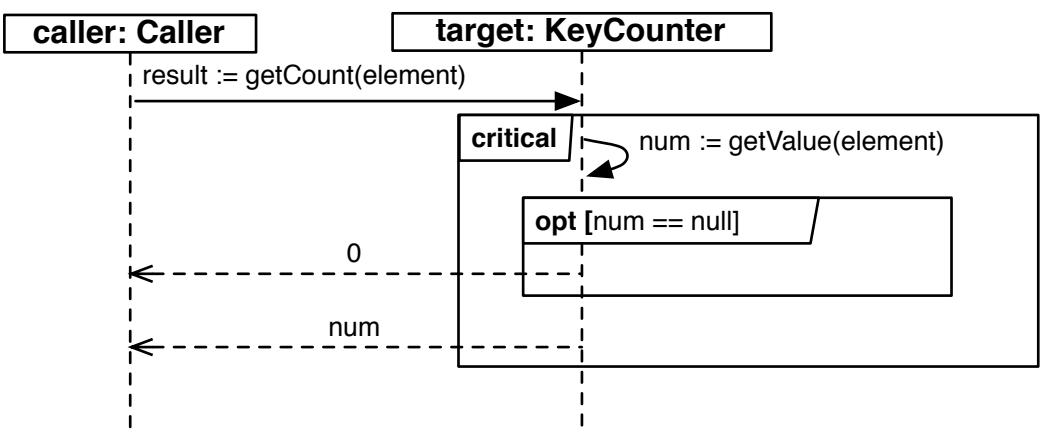
message view decrement



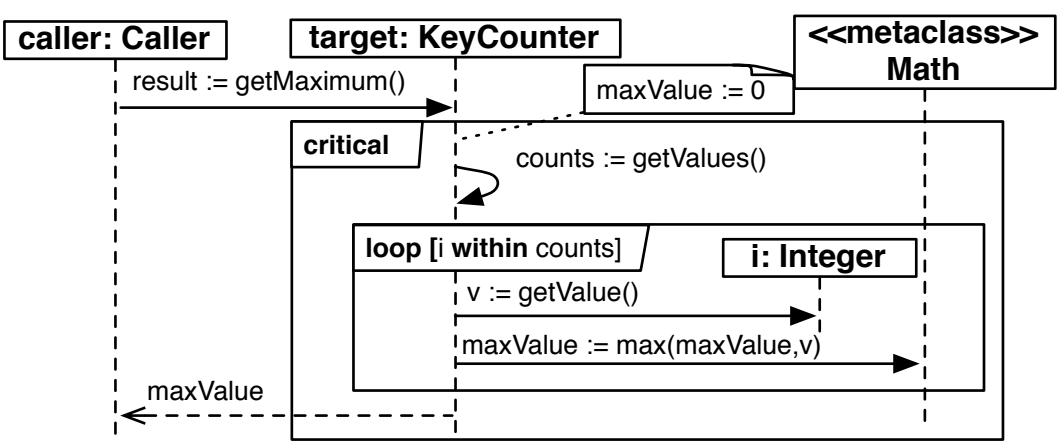
message view decrementAll



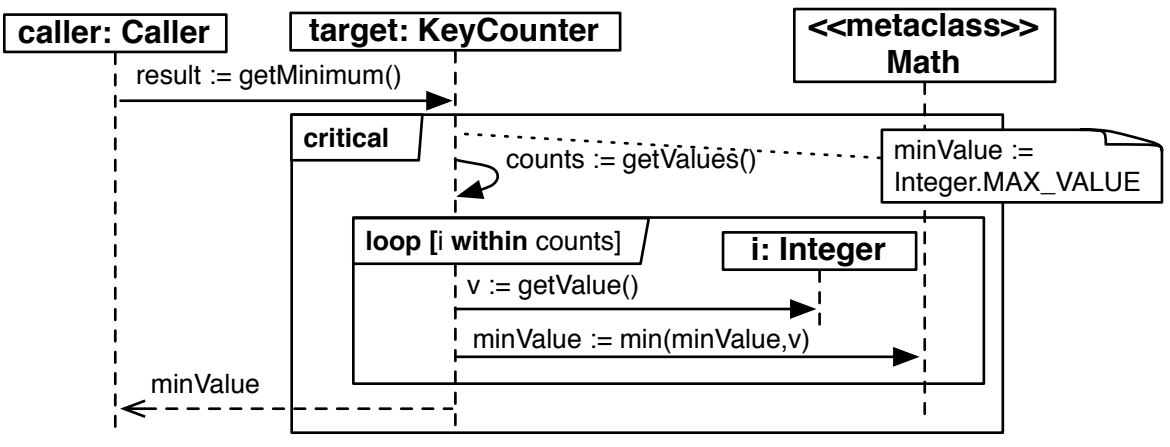
message view getCount



message view getMaximum



message view getMinimum



message view initializeCounts