removeTokenExecutor

Actor

:GameController   :RemoveTokenAction   :CheckLegalMove   :Actor

Actor → :GameController : moveRemoveTokenExecutor (action, currentActor, nextActor)

loop [24]
- :GameController → :GameController : removeImage (nodeId, "removeToken")
- :GameController → :GameController : getChildList (nodeID)

:GameController → :RemoveTokenAction : execute (currentActor, nodeList)

:GameController → :RemoveTokenAction : getNodeId()
:RemoveTokenAction --> :GameController : int

:GameController → :GameController : getChildList (nodeID)

loop [childList.size() > 1]
- :GameController → :GameController : get(childList.size() -1)
- :GameController → :GameController : remove(node)

:GameController → :GameController : updateTokenCount(nextActor)

:GameController → :Actor : getNumberOfTokensOnBoard()
:Actor --> :GameController : int

alt [getNumberOfTokensOnBoard() > 3]
:GameController → :CheckLegalMove : calculateLegalMove(nextActor, nodeList)

:GameController → :CheckLegalMove : getCurrentActions()
:CheckLegalMove --> :GameController : Map<Integer, ArrayList<Action>>

alt [checkIfHaveLegalMove(returnAction, nextActor.getTokenColour())]

Loop [24]
- :GameController → :GameController : addChecker(i, returnAction, nextActor, currentActor)

[else]
- :GameController → :GameController : endGame(currentActor)

[else]

alt [getNumberOfTokensOnBoard() == 3]
:GameController → :CheckLegalMove : calculateLegalFly(nextActor, nodeList)

:GameController → :CheckLegalMove : getCurrentActions()
:CheckLegalMove --> :GameController : Map<Integer, ArrayList<Action>>

loop [24]
- :GameController → :GameController : addChecker(i, returnAction, nextActor, currentActor)

[else]
- :GameController → :GameController : endGame(currentActor)