|  |
| --- |
| Puissance 4 et MCTS  Algorithme de recherche d’arbre Monte Carlo UCT en temps limité |
| V. DAMM – F. NOSARI - 14-déc.-16 |
| Table des matières  Aucune entrée de table des matières n'a été trouvée. |

# Programme

Le programme a été refait en java pour bénéficier d’un langage objet. Nous allons détailler les classes et les parties importantes de l’algorithme.

## Pseudo algorithme

**search\_MCTS**(s0, budget)

i0 = new Node(null,s0,null)  
 do

ie = threePolicy(i0)

rk = defaultPolicy(ie);

backup(ie,rk);

while(budget)

return bestChild(n0).action()

end

**threePolicy**(i0)

ie = i0

while(isNonTerminal(ie))

if(!isFullyExpanded(ie))

return expand(ie)

else

ie = besChild(ie)

endif

endwhile

return ie

end

**defaultPolicy**(i0)

ie = i0

while(isNonTerminal(ie))

ie = expand(ie)

visit(ie)

endwhile

return ie.state.reward

end

**backup**(i0, rk)

ie = i0

while(ie != null)

visit(ie)

ie.reward += rk

ie = ie.parent

endwhile

end

**expand**(i0)

a = untriedAction(i0)

i = new Node(i0, use(i0.state, a), a)

addChild(i0,i)

return I

end

**bestChild**(i0)

maxB = -inf

foreach(ie.children as ic)

b = B(ic)

if( b > maxB)

maxB = b

bestChild = ic

endif

endforeach

return bestChild

end

**B**(i0, c)

coeff = 1

if(! I9.state.computerTurn)

coeff = -1

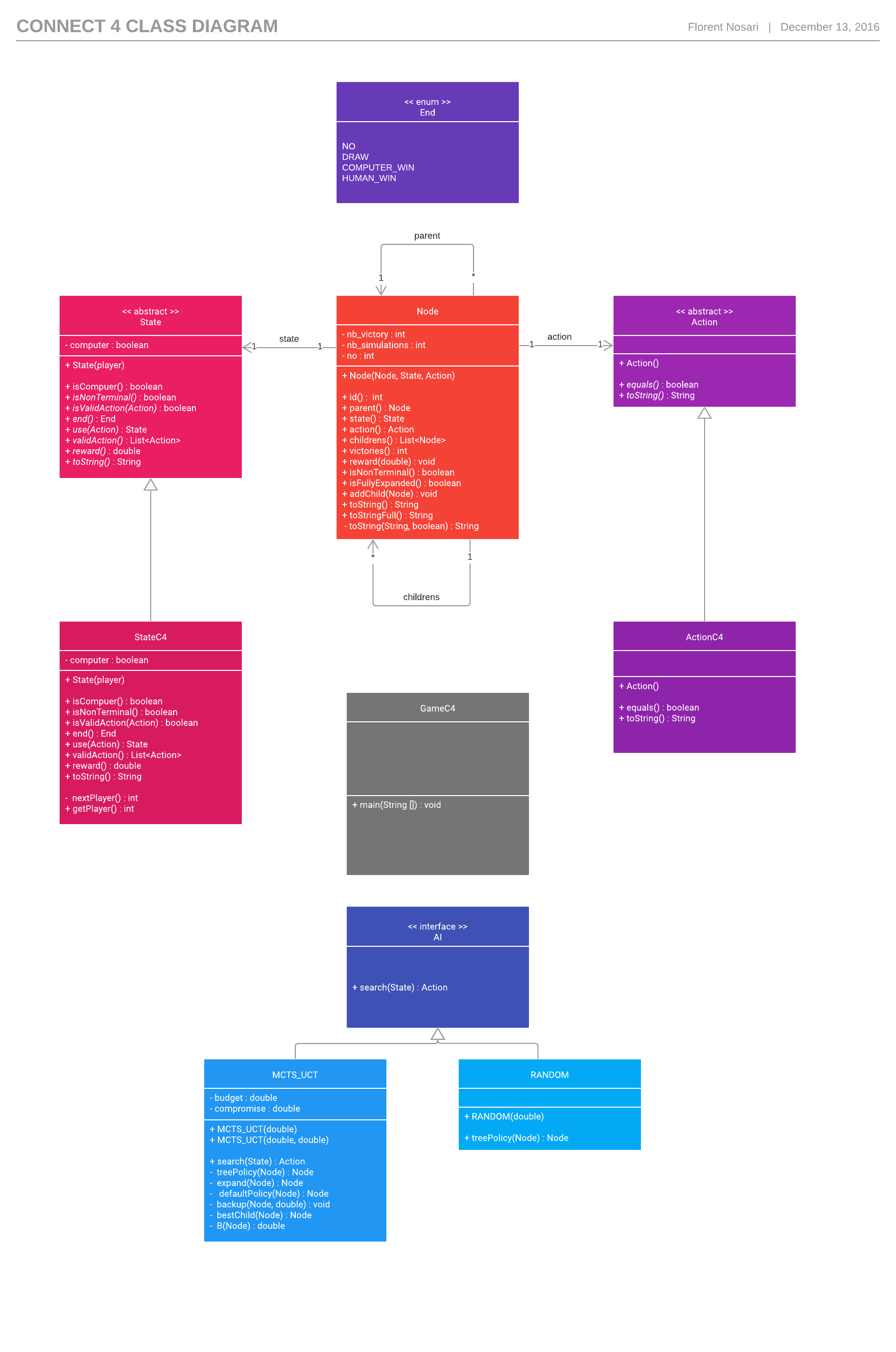
endif

return coeff \* (i0.victories / i.simulations)

+ c \* sqrt(log(i.parent.simulations) / i.simulations)

end

## Diagramme de classe du programme



# Questions

* Affichez à chaque coup de l’ordinateur, le nombre de simulations réalisées pour calculer ce coup et une estimation de la probabilité de victoire pour l’ordinateur.
* Testez différentes limites de temps pour l’ordinateur et comparez les résultats obtenus. A partir de quel temps de calcul l’ordinateur vous bat à tous les coups ?
* Implémentez l’amélioration des simulations consistant à toujours choisir un coup gagnant lorsque cela est possible. Comparez la qualité de jeu de cette nouvelle version avec la précédente et expliquez à quoi cela est dû.
* Si vous travaillez en C, quelle est l’utilité ici de compiler avec gcc -O3 plutôt qu’avec les options par défaut ? Donnez des exemples illustratifs.
* Comparez les critères "max" et "robuste" pour choisir le coup à jouer en fin d’algorithme.
* Conduisent-ils souvent à des coups différents ? Lequel paraît donner la meilleure performance ?
* Donnez une estimation du temps de calcul nécessaire pour jouer le premier coup avec l’algorithme Minimax (sans alpha-beta ni limitation de profondeur).