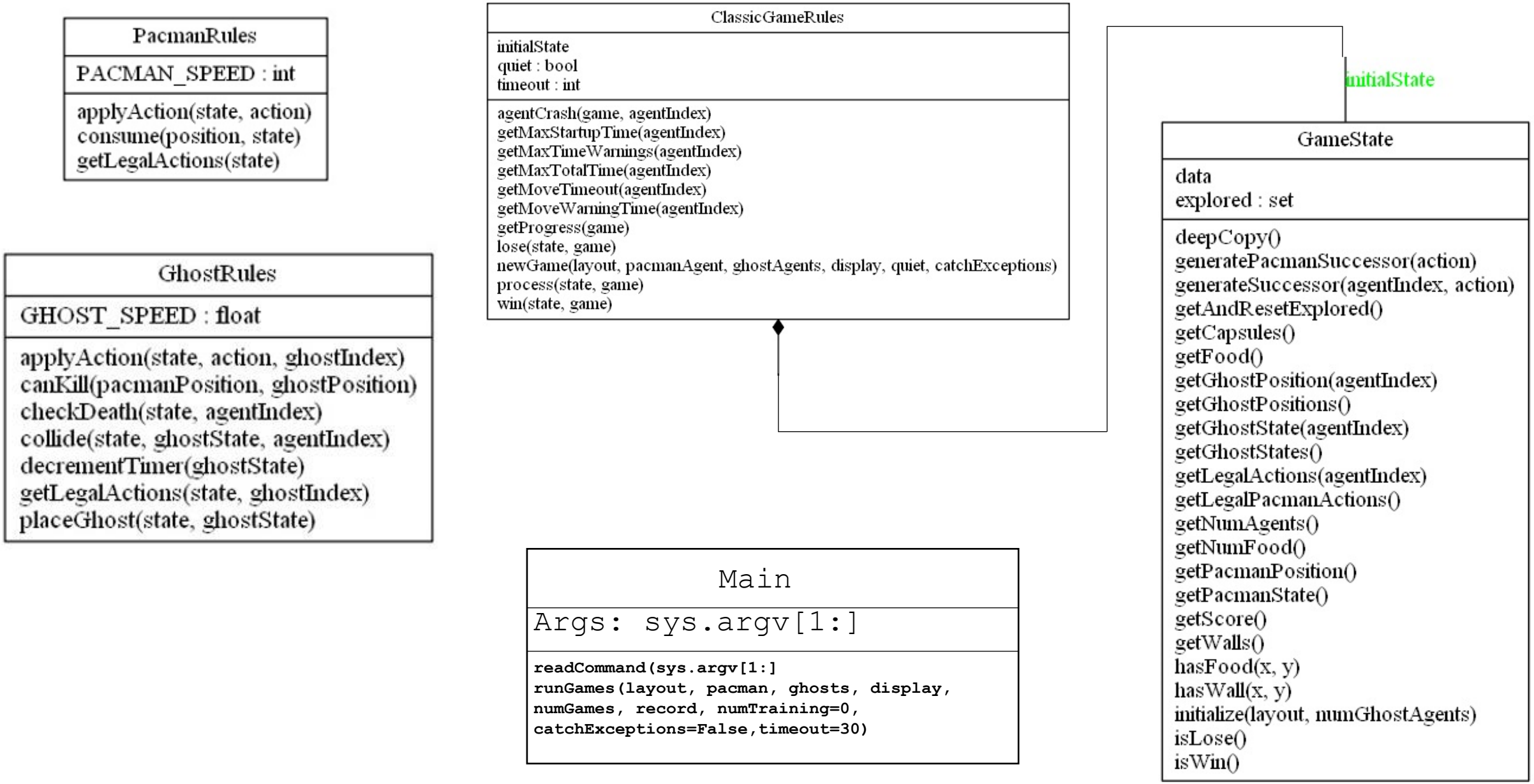
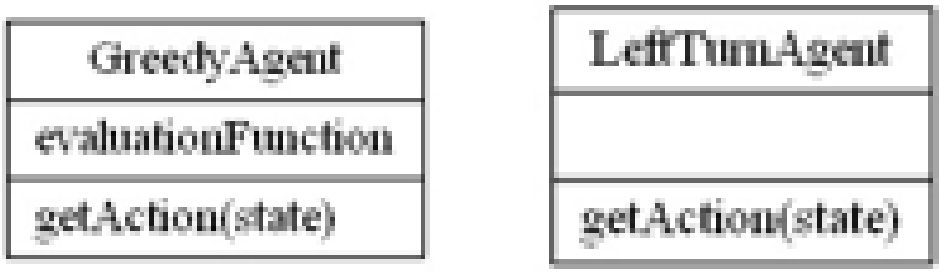
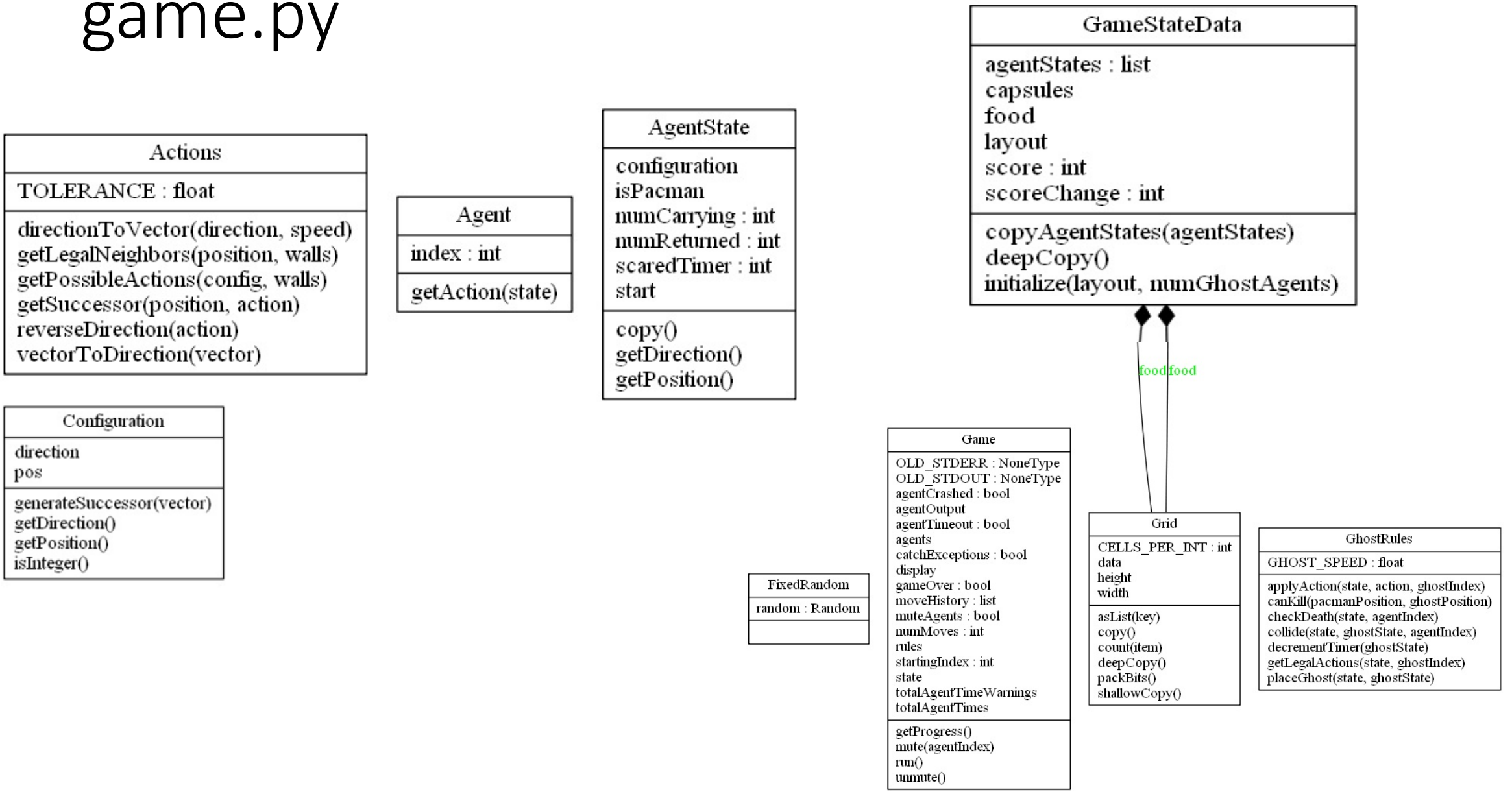
pacman.py


PacmanRules
PACMAN_SPEED : int
applyAction(state, action)
consume(position, state)
getLegalActions(state)
ClassicGameRules
initialState
quiet : bool
timeout : int
agentCrash(game, agentIndex)
getMaxStartupTime(agentIndex)
getMaxTimeWarnings(agentIndex)
getMaxTotalTime(agentIndex)
getMoveTimeout(agentIndex)
getMoveWarningTime(agentIndex)
getProgress(game)
lose(state, game)
newGame(layout, pacmanAgent, ghostAgents, display, quiet, catchExceptions)
process(state, game)
win(state, game)
initialState
GameState
data
explored : set
deepCopy()
generatePacmanSuccessor(action)
generateSuccessor(agentIndex, action)
getAndResetExplored()
getCapsules()
getFood()
getGhostPosition(agentIndex)
getGhostPositions()
getGhostState(agentIndex)
getGhostStates()
getLegalActions(agentIndex)
getLegalPacmanActions()
getNumAgents()
getNumFood()
getPacmanPosition()
getPacmanState()
getScore()
getWalls()
hasFood(x, y)
hasWall(x, y)
initialize(layout, numGhostAgents)
isLose()
isWin()
GhostRules
GHOST_SPEED : float
applyAction(state, action, ghostIndex)
canKill(pacmanPosition, ghostPosition)
checkDeath(state, agentIndex)
collide(state, ghostState, agentIndex)
decrementTimer(ghostState)
getLegalActions(state, ghostIndex)
placeGhost(state, ghostState)
Main
Args: sys.argv[1:]
readCommand(sys.argv[1:]
runGames(layout, pacman, ghosts, display,
numGames, record, numTraining=0,
catchExceptions=False,timeout=30)

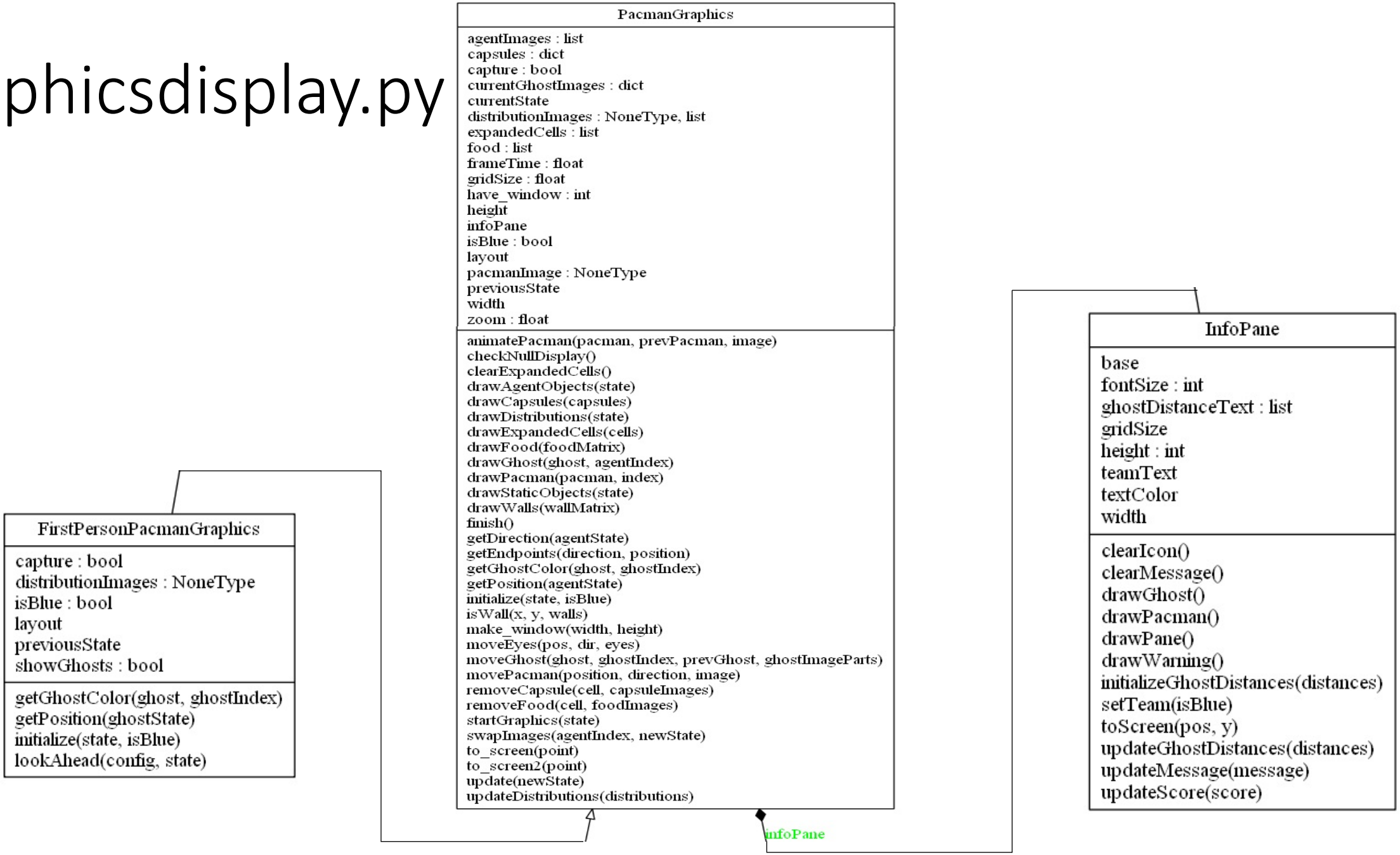
pacmanAgents.py



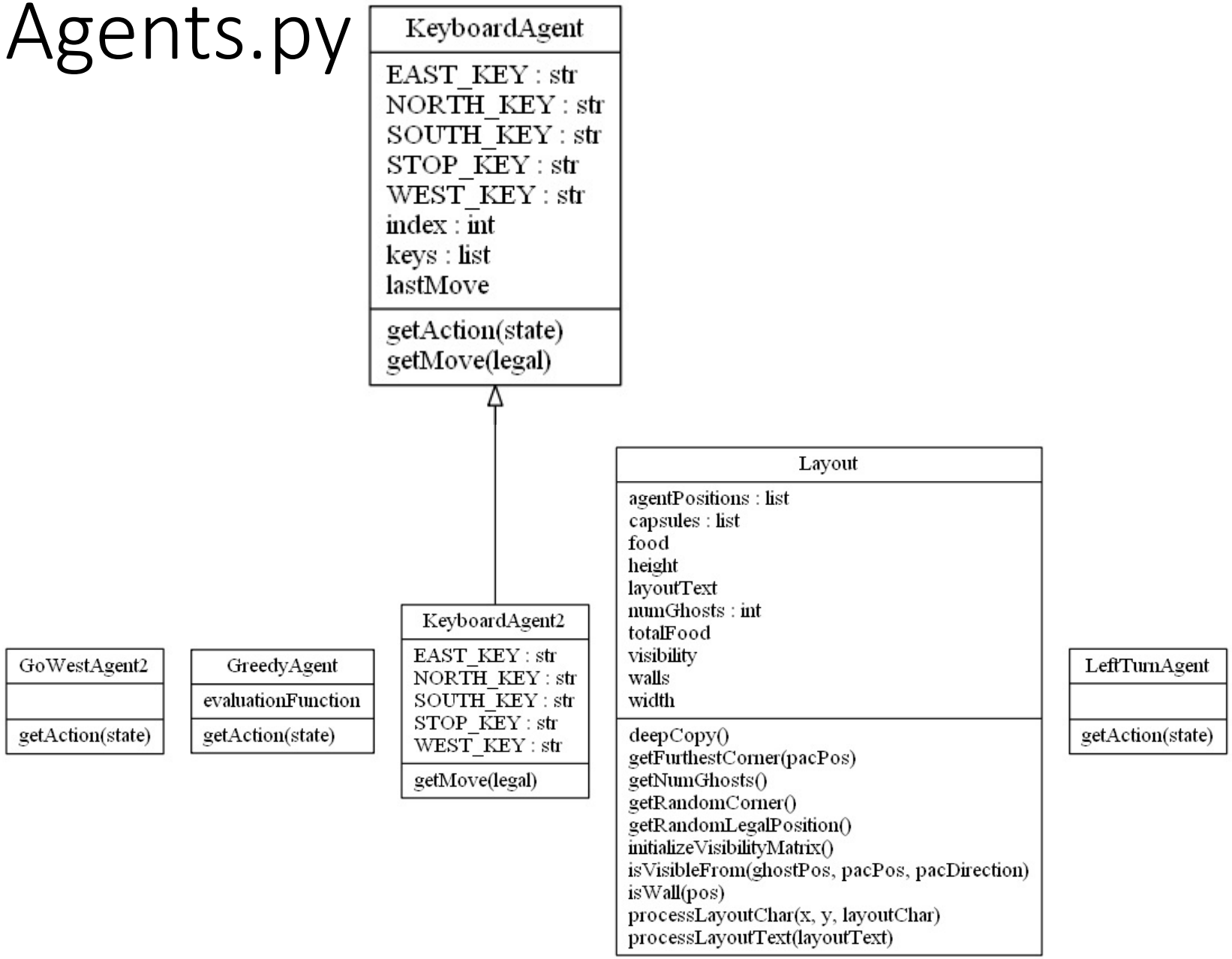
game.py


Actions
TOLERANCE : float
directionToVector(direction, speed)
getLegalNeighbors(position, walls)
getPossibleActions(config, walls)
getSuccessor(position, action)
reverseDirection(action)
vectorToDirection(vector)
Agent
index : int
getAction(state)
AgentState
configuration
isPacman
numCarrying : int
numReturned : int
scaredTimer : int
start
copy()
getDirection()
getPosition()
GameStateData
agentStates : list
capsules
food
layout
score : int
scoreChange : int
copyAgentStates(agentStates)
deepCopy()
initialize(layout, numGhostAgents)
food
food
Configuration
direction
pos
generateSuccessor(vector)
getDirection()
getPosition()
isInteger()
FixedRandom
random : Random
Game
OLD_STDERR : NoneType
OLD_STDOUT : NoneType
agentCrashed : bool
agentOutput
agentTimeout : bool
agents
catchExceptions : bool
display
gameOver : bool
moveHistory : list
muteAgents : bool
numMoves : int
rules
startingIndex : int
state
totalAgentTimeWarnings
totalAgentTimes
getProgress()
mute(agentIndex)
run()
unmute()
Grid
CELLS_PER_INT : int
data
height
width
asList(key)
copy()
count(item)
deepCopy()
packBits()
shallowCopy()
GhostRules
GHOST_SPEED : float
applyAction(state, action, ghostIndex)
canKill(pacmanPosition, ghostPosition)
checkDeath(state, agentIndex)
collide(state, ghostState, agentIndex)
decrementTimer(ghostState)
getLegalActions(state, ghostIndex)
placeGhost(state, ghostState)

graphicsdisplay.py



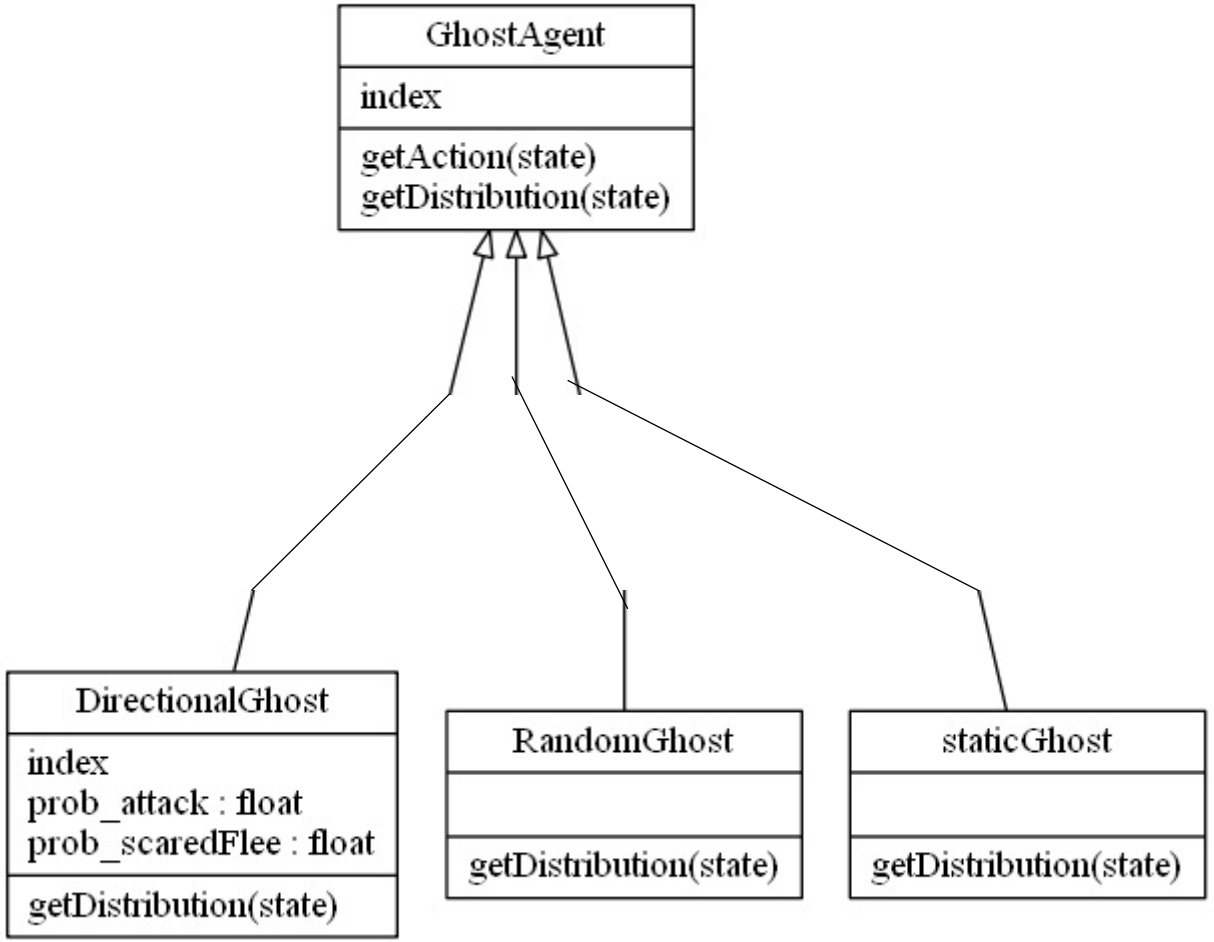
keyboardAgents.py



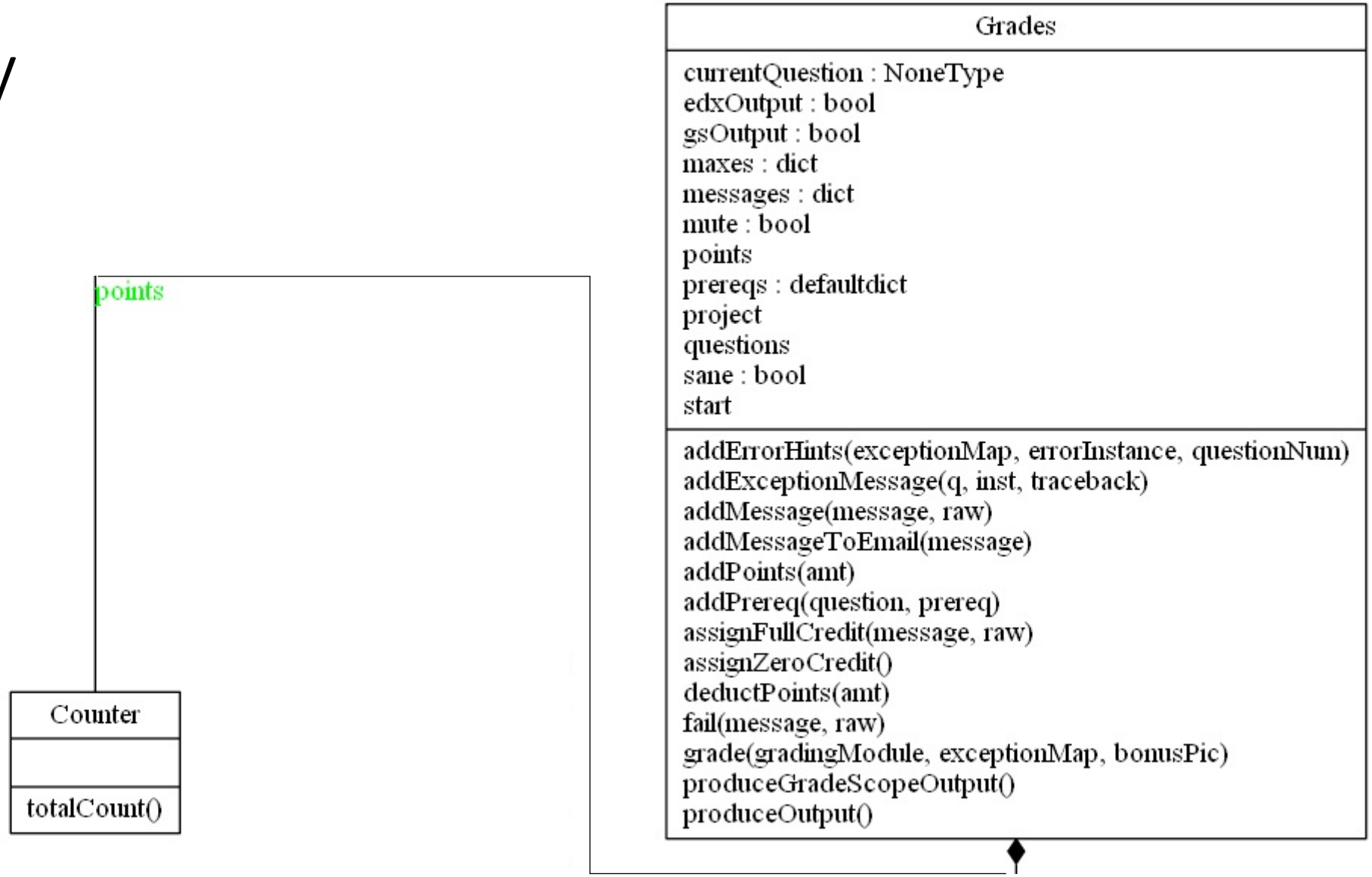
layout.py


Layout
agentPositions : list
capsules : list
food
height
layoutText
numGhosts : int
totalFood
visibility
walls
width
deepCopy()
getFurthestCorner(pacPos)
getNumGhosts()
getRandomCorner()
getRandomLegalPosition()
initializeVisibilityMatrix()
isVisibleFrom(ghostPos, pacPos, pacDirection)
isWall(pos)
processLayoutChar(x, y, layoutChar)
processLayoutText(layoutText)

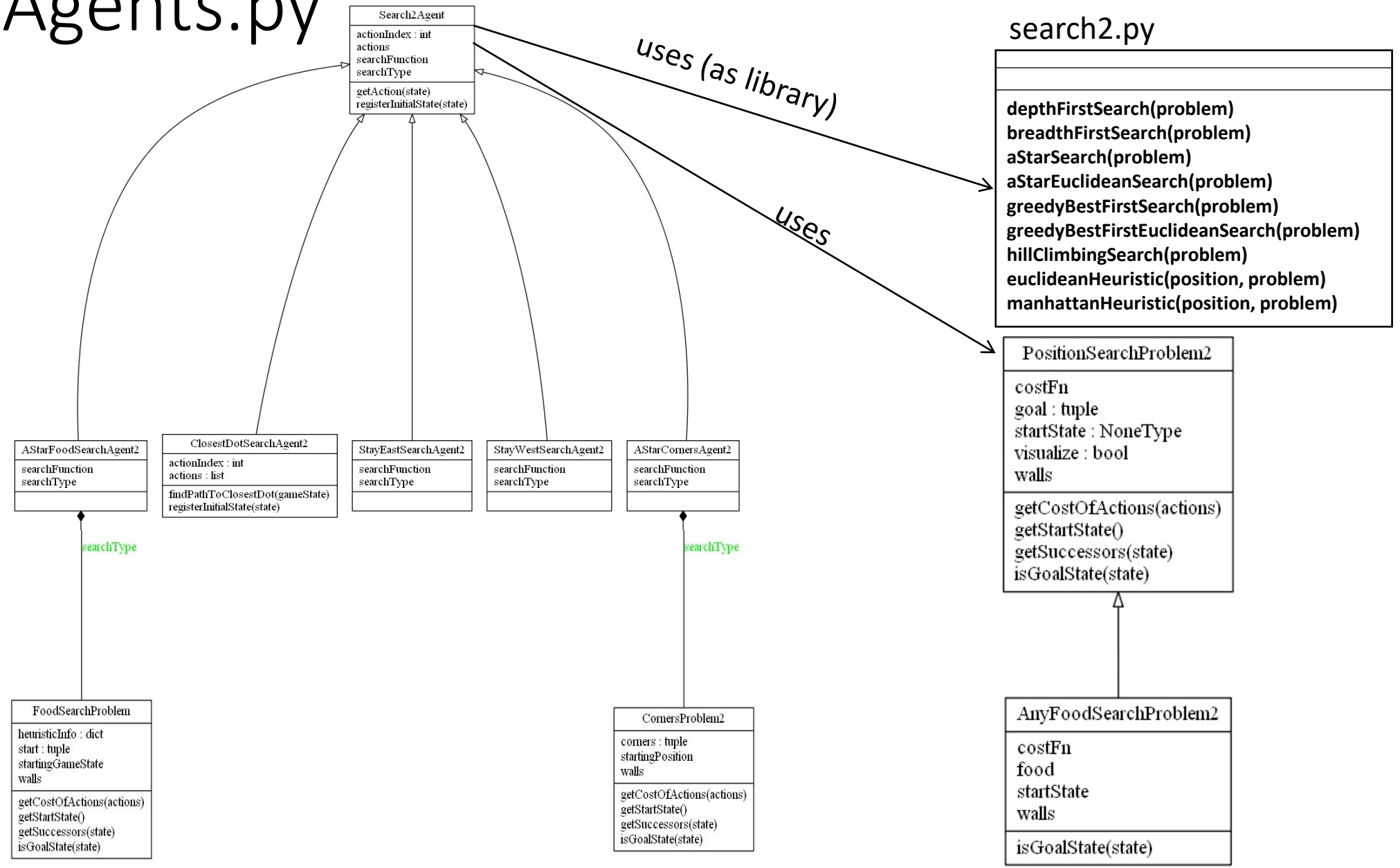
ghostAgents.py



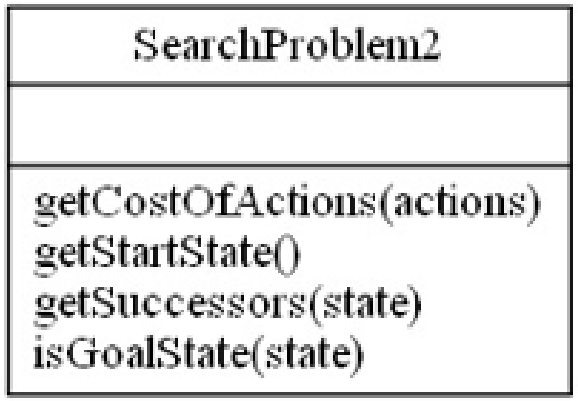
grading.py


points
Counter
totalCount()
Grades
currentQuestion : NoneType
edxOutput : bool
gsOutput : bool
maxes : dict
messages : dict
mute : bool
points
prereqs : defaultdict
project
questions
sane : bool
start
addErrorHints(exceptionMap, errorInstance, questionNum)
addExceptionMessage(q, inst, traceback)
addMessage(message, raw)
addMessageToEmail(message)
addPoints(amt)
addPrereq(question, prereq)
assignFullCredit(message, raw)
assignZeroCredit()
deductPoints(amt)
fail(message, raw)
grade(gradingModule, exceptionMap, bonusPic)
produceGradeScopeOutput()
produceOutput()

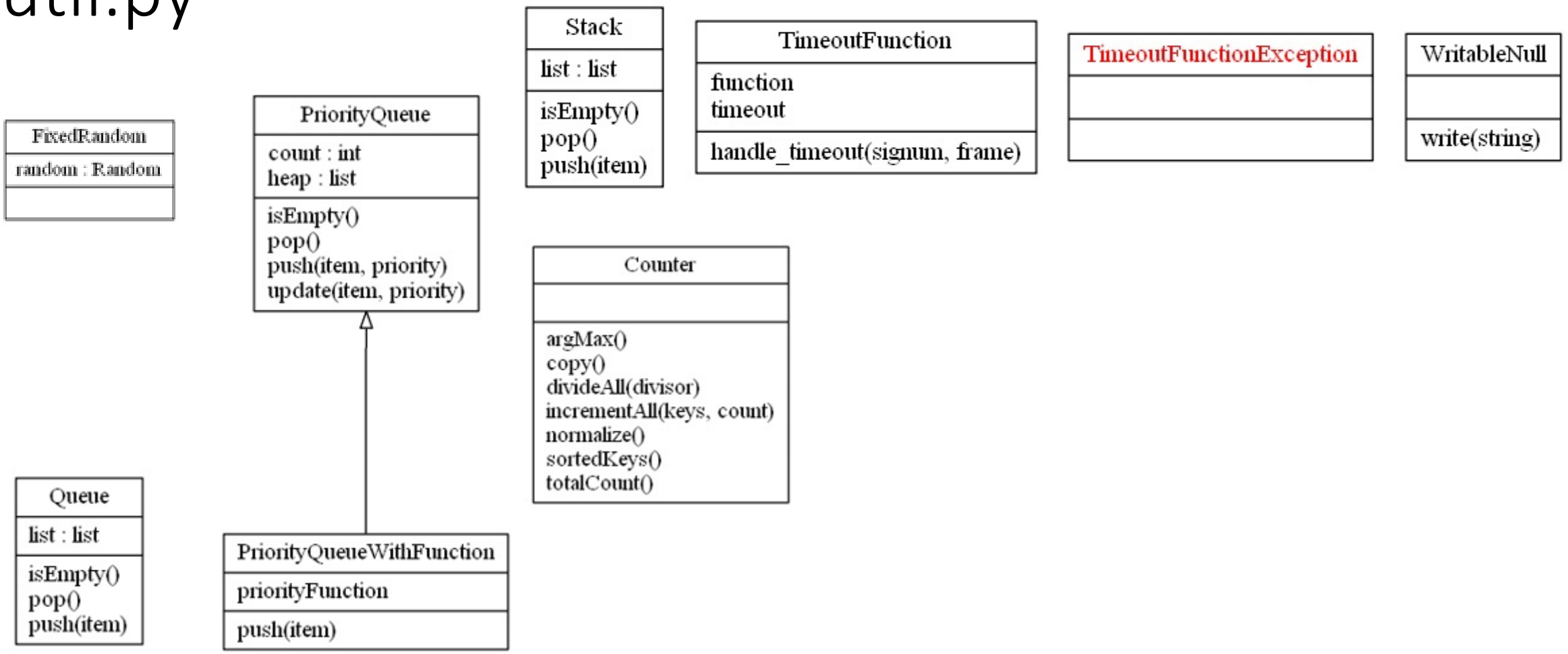
search2Agents.py



search2.py


SearchProblem2
getCostOfActions(actions)
getStartState()
getSuccessors(state)
isGoalState(state)

util.py


FixedRandom
random : Random
PriorityQueue
count : int
heap : list
isEmpty()
pop()
push(item, priority)
update(item, priority)
Stack
list : list
isEmpty()
pop()
push(item)
TimeoutFunction
function
timeout
handle_timeout(signum, frame)
TimeoutFunctionException
WritableNull
write(string)
Counter
argMax()
copy()
divideAll(divisor)
incrementAll(keys, count)
normalize()
sortedKeys()
totalCount()
Queue
list : list
isEmpty()
pop()
push(item)
PriorityQueueWithFunction
priorityFunction
push(item)