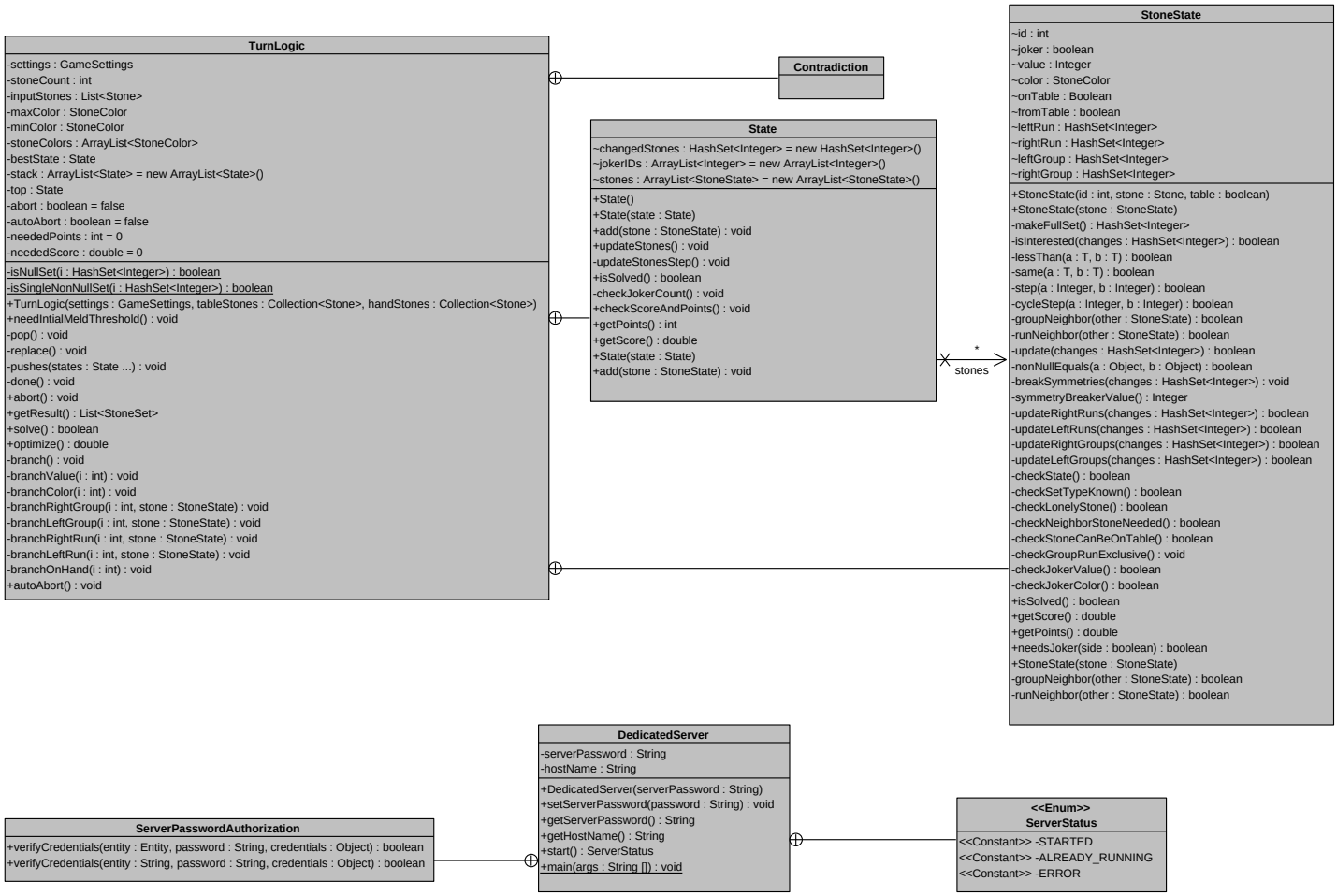
TurnLogic
-settings : GameSettings
-stoneCount : int
-inputStones : List<Stone>
-maxColor : StoneColor
-minColor : StoneColor
-stoneColors : ArrayList<StoneColor>
-bestState : State
-stack : ArrayList<State> = new ArrayList<State>()
-top : State
-abort : boolean = false
-autoAbort : boolean = false
-neededPoints : int = 0
-neededScore : double = 0
-isNullSet(i : HashSet<Integer>) : boolean
-isSingleNonNullSet(i : HashSet<Integer>) : boolean
+TurnLogic(settings : GameSettings, tableStones : Collection<Stone>, handStones : Collection<Stone>)
+needIntialMeldThreshold() : void
-pop() : void
-replace() : void
-pushes(states : State ...) : void
-done() : void
+abort() : void
+getResult() : List<StoneSet>
+solve() : boolean
+optimize() : double
-branch() : void
-branchValue(i : int) : void
-branchColor(i : int) : void
-branchRightGroup(i : int, stone : StoneState) : void
-branchLeftGroup(i : int, stone : StoneState) : void
-branchRightRun(i : int, stone : StoneState) : void
-branchLeftRun(i : int, stone : StoneState) : void
-branchOnHand(i : int) : void
+autoAbort() : void
Contradiction
State
~changedStones : HashSet<Integer> = new HashSet<Integer>()
~jokerIDs : ArrayList<Integer> = new ArrayList<Integer>()
~stones : ArrayList<StoneState> = new ArrayList<StoneState>()
+State()
+State(state : State)
+add(stone : StoneState) : void
+updateStones() : void
-updateStonesStep() : void
+isSolved() : boolean
-checkJokerCount() : void
+checkScoreAndPoints() : void
+getPoints() : int
+getScore() : double
+State(state : State)
+add(stone : StoneState) : void
*
stones
StoneState
~id : int
~joker : boolean
~value : Integer
~color : StoneColor
~onTable : Boolean
~fromTable : boolean
~leftRun : HashSet<Integer>
~rightRun : HashSet<Integer>
~leftGroup : HashSet<Integer>
~rightGroup : HashSet<Integer>
+StoneState(id : int, stone : Stone, table : boolean)
+StoneState(stone : StoneState)
-makeFullSet() : HashSet<Integer>
-isInterested(changes : HashSet<Integer>) : boolean
-lessThan(a : T, b : T) : boolean
-same(a : T, b : T) : boolean
-step(a : Integer, b : Integer) : boolean
-cycleStep(a : Integer, b : Integer) : boolean
-groupNeighbor(other : StoneState) : boolean
-runNeighbor(other : StoneState) : boolean
-update(changes : HashSet<Integer>) : boolean
-nonNullEquals(a : Object, b : Object) : boolean
-breakSymmetries(changes : HashSet<Integer>) : void
-symmetryBreakerValue() : Integer
-updateRightRuns(changes : HashSet<Integer>) : boolean
-updateLeftRuns(changes : HashSet<Integer>) : boolean
-updateRightGroups(changes : HashSet<Integer>) : boolean
-updateLeftGroups(changes : HashSet<Integer>) : boolean
-checkState() : boolean
-checkSetTypeKnown() : boolean
-checkLonelyStone() : boolean
-checkNeighborStoneNeeded() : boolean
-checkStoneCanBeOnTable() : boolean
-checkGroupRunExclusive() : void
-checkJokerValue() : boolean
-checkJokerColor() : boolean
+isSolved() : boolean
+getScore() : double
+getPoints() : double
+needsJoker(side : boolean) : boolean
+StoneState(stone : StoneState)
-groupNeighbor(other : StoneState) : boolean
-runNeighbor(other : StoneState) : boolean
DedicatedServer
-serverPassword : String
-hostName : String
+DedicatedServer(serverPassword : String)
+setServerPassword(password : String) : void
+getServerPassword() : String
+getHostName() : String
+start() : ServerStatus
+main(args : String []) : void
ServerPasswordAuthorization
+verifyCredentials(entity : Entity, password : String, credentials : Object) : boolean
+verifyCredentials(entity : String, password : String, credentials : Object) : boolean
<<Enum>>
ServerStatus
<<Constant>> -STARTED
<<Constant>> -ALREADY_RUNNING
<<Constant>> -ERROR