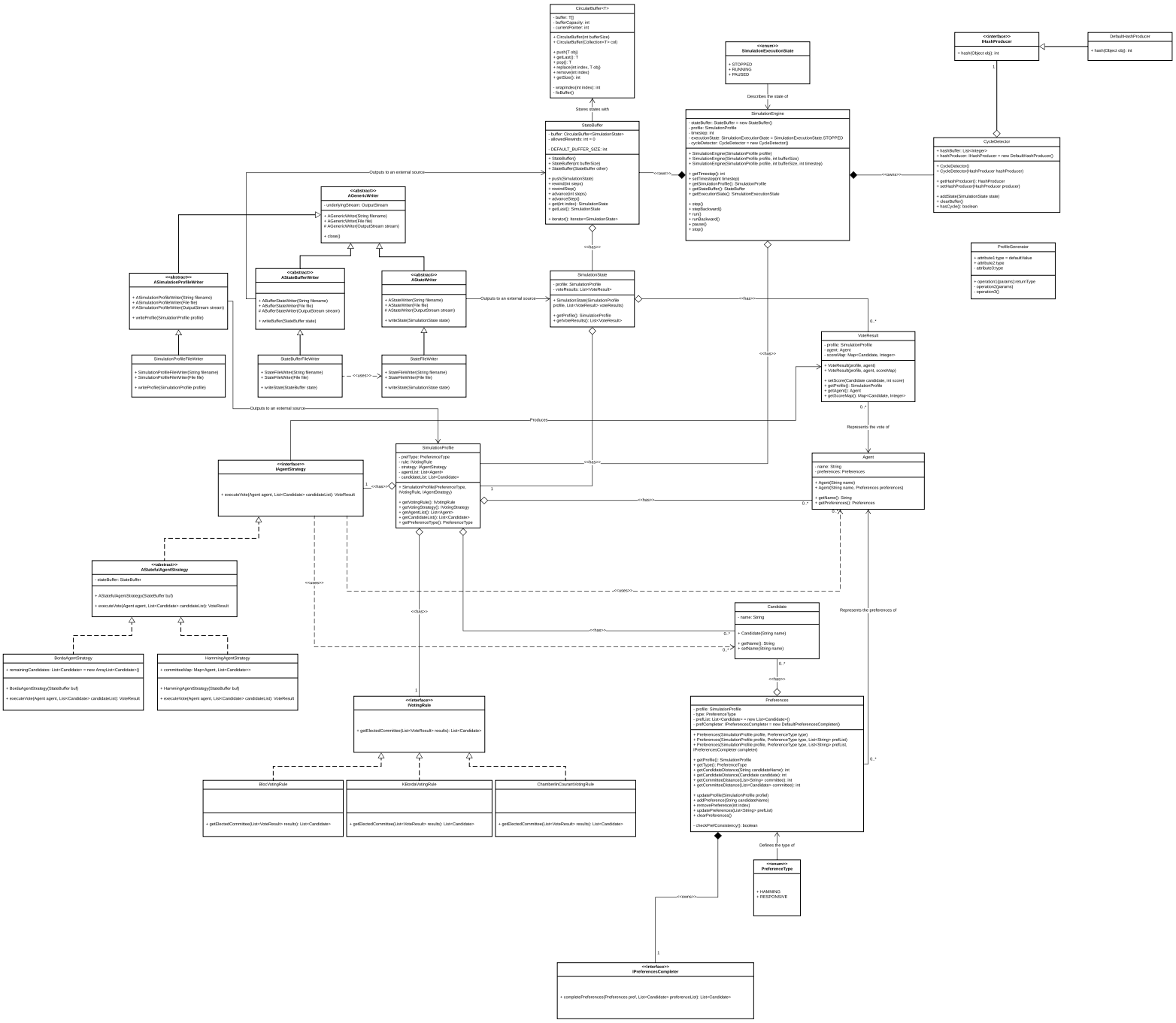
CircularBuffer<T>
- buffer: T[]
- bufferCapacity: int
- currentPointer: int
+ CircularBuffer(int bufferSize)
+ CircularBuffer(Collection<T> col)
+ push(T obj)
+ getLast(): T
+ pop(): T
+ replace(int index, T obj)
+ removeAt(int index)
+ getSize(): int
- wrapIndex(int index): int
- fixBuffer()
Stores states with
StateBuffer
- buffer: CircularBuffer<SimulationState>
- allowedRewinds: int = 0
- DEFAULT_BUFFER_SIZE: int
+ StateBuffer()
+ StateBuffer(int bufferSize)
+ StateBuffer(StateBuffer other)
+ push(SimulationState)
+ rewind(int steps)
+ rewindStep()
+ advance(int steps)
+ advanceStep()
+ get(int index): SimulationState
+ getLast(): SimulationState
+ iterator(): Iterator<SimulationState>
<<own>>
<<enum>>
SimulationExecutionState
+ STOPPED
+ RUNNING
+ PAUSED
Describes the state of
SimulationEngine
- stateBuffer: StateBuffer = new StateBuffer()
- profile: SimulationProfile
- timestep: int
- executionState: SimulationExecutionState = SimulationExecutionState.STOPPED
- cycleDetector: CycleDetector = new CycleDetector()
+ SimulationEngine(SimulationProfile profile)
+ SimulationEngine(SimulationProfile profile, int bufferSize)
+ SimulationEngine(SimulationProfile profile, int bufferSize, int timestep)
+ getTimestep(): int
+ setTimestep(int timestep)
+ getSimulationProfile(): SimulationProfile
+ getStateBuffer(): StateBuffer
+ getExecutionState(): SimulationExecutionState
+ step()
+ stepBackward()
+ run()
+ runBackward()
+ pause()
+ stop()
<<interface>>
IHashProducer
+ hash(Object obj): int
DefaultHashProducer
+ hash(Object obj): int
1
CycleDetector
+ hashBuffer: List<Integer>
+ hashProducer: IHashProducer = new DefaultHashProducer()
+ CycleDetector()
+ CycleDetector(IHashProducer hashProducer)
+ getHashProducer(): IHashProducer
+ setHashProducer(IHashProducer producer)
+ addState(SimulationState state)
+ clearBuffer()
+ hasCycle(): boolean
<<own>>
Outputs to an external source
<<abstract>>
AGenericWriter
- underlyingStream: OutputStream
+ AGenericWriter(String filename)
+ AGenericWriter(File file)
# AGenericWriter(OutputStream stream)
+ close()
<<abstract>>
ASimulationProfileWriter
+ ASimulationProfileWriter(String filename)
+ ASimulationProfileWriter(File file)
# ASimulationProfileWriter(OutputStream stream)
+ writeProfile(SimulationProfile profile)
<<abstract>>
AStateBufferWriter
+ AStateBufferWriter(String filename)
+ AStateBufferWriter(File file)
# AStateBufferWriter(OutputStream stream)
+ writeBuffer(StateBuffer state)
<<abstract>>
AStateWriter
+ AStateWriter(String filename)
+ AStateWriter(File file)
# AStateWriter(OutputStream stream)
+ writeState(SimulationState state)
Outputs to an external source
<<has>>
SimulationState
- profile: SimulationProfile
- voteResults: List<VoteResult>
+ SimulationState(SimulationProfile profile, List<VoteResult> voteResults)
+ getProfile(): SimulationProfile
+ getVoteResults(): List<VoteResult>
<<has>>
0..*
ProfileGenerator
+ attribute1:type = defaultValue
+ attribute2:type
attribute3:type
+ operation1()param():returnType
operation2()param()
operation3()
SimulationProfileFileWriter
+ SimulationProfileFileWriter(String filename)
+ SimulationProfileFileWriter(File file)
+ writeProfile(SimulationProfile profile)
StateBufferFileWriter
+ StateFileWriter(String filename)
+ StateFileWriter(File file)
+ writeState(StateBuffer state)
<<uses>>
StateFileWriter
+ StateFileWriter(String filename)
+ StateFileWriter(File file)
+ writeState(SimulationState state)
<<has>>
VoteResult
- profile: SimulationProfile
- agent: Agent
- scoreMap: Map<Candidate, Integer>
+ VoteResult(profile, agent)
+ VoteResult(profile, agent, scoreMap)
+ setScore(Candidate candidate, int score)
+ getProfile(): SimulationProfile
+ getAgent(): Agent
+ getScoreMap(): Map<Candidate, Integer>
0..*
Represents the vote of
Outputs to an external source
Produces
<<interface>>
IAgentStrategy
+ executeVote(Agent agent, List<Candidate> candidateList): VoteResult
1
<<has>>
SimulationProfile
- profileType: PreferenceType
- rule: IVotingRule
- strategy: IAgentStrategy
- agentList: List<Agent>
- candidateList: List<Candidate>
+ SimulationProfile(PreferenceType, IVotingRule, IAgentStrategy)
+ getVotingRule(): IVotingRule
+ getVotingStrategy(): IVotingStrategy
+ getAgentList(): List<Agent>
+ getCandidateList(): List<Candidate>
+ getPreferenceType(): PreferenceType
<<has>>
1
<<has>>
0..*
Agent
- name: String
- preferences: Preferences
+ Agent(String name)
+ Agent(String name, Preferences preferences)
+ getName(): String
+ getPreferences(): Preferences
0..*
<<abstract>>
AStateforAgentStrategy
- stateBuffer: StateBuffer
+ AStateforAgentStrategy(StateBuffer buf)
+ executeVote(Agent agent, List<Candidate> candidateList): VoteResult
<<uses>>
1
<<has>>
<<uses>>
Candidate
- name: String
+ Candidate(String name)
+ getName(): String
+ setName(String name)
<<has>>
0..*
0..*
Represents the preferences of
0..*
<<has>>
BordaAgentStrategy
+ remainingCandidates: List<Candidate> = new ArrayList<Candidate>()
+ BordaAgentStrategy(StateBuffer buf)
+ executeVote(Agent agent, List<Candidate> candidateList): VoteResult
HammingAgentStrategy
+ committeeMap: Map<Agent, List<Candidate>>
+ HammingAgentStrategy(StateBuffer buf)
+ executeVote(Agent agent, List<Candidate> candidateList): VoteResult
<<interface>>
IVotingRule
+ getElectedCommittee(List<VoteResult> results): List<Candidate>
1
Preferences
- profile: SimulationProfile
- type: PreferenceType
- preList: List<Candidate> = new List<Candidate>()
- prefCompleter: IPreferencesCompleter = new DefaultPreferencesCompleter()
+ Preferences(SimulationProfile profile, PreferenceType type)
+ Preferences(SimulationProfile profile, PreferenceType type, List<String> preList)
+ Preferences(SimulationProfile profile, PreferenceType type, List<String> preList, IPreferencesCompleter completer)
+ getProfile(): SimulationProfile
+ getType(): PreferenceType
+ getCandidateDistance(String candidateName): int
+ getCandidateDistance(Candidate candidate): int
+ getCommitteeDistance(List<String> committee): int
+ getCommitteeDistance(List<Candidate> committee): int
+ updateProfile(SimulationProfile profile)
+ addPreference(String candidateName)
+ removePreference(int index)
+ updatePreferences(List<String> preList)
+ clearPreferences()
- checkPrefConsistency(): boolean
Defines the type of
<<enum>>
PreferenceType
+ HAMMING
+ RESPONSIVE
BlocVotingRule
+ getElectedCommittee(List<VoteResult> results): List<Candidate>
KBordaVotingRule
+ getElectedCommittee(List<VoteResult> results): List<Candidate>
ChamberlinCourantVotingRule
+ getElectedCommittee(List<VoteResult> results): List<Candidate>
<<uses>>
1
<<interface>>
IPreferencesCompleter
+ completePreferences(Preferences pref, List<Candidate> preferenceList): List<Candidate>