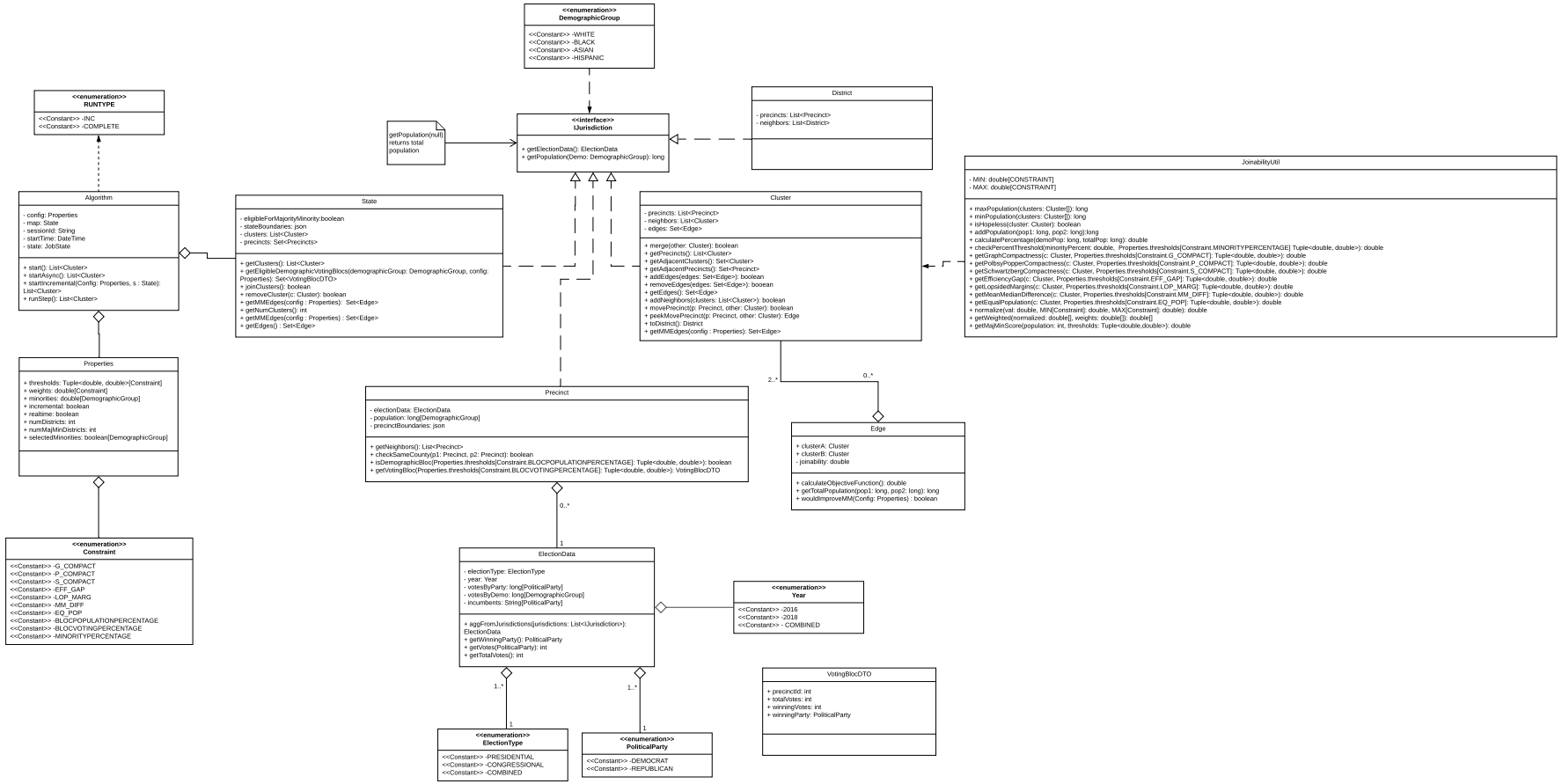
<<enumeration>>
DemographicGroup
<<Constant>> -WHITE
<<Constant>> -BLACK
<<Constant>> -ASIAN
<<Constant>> -HISPANIC
<<enumeration>>
RUNTYPE
<<Constant>> -INC
<<Constant>> -COMPLETE
getPopulation(null) returns total population
<<interface>>
IJurisdiction
+ getElectionData(): ElectionData
+ getPopulation(Demo: DemographicGroup): long
District
- precincts: List<Precinct>
- neighbors: List<District>
JoinabilityUtil
- MIN: double[CONSTRAINT]
- MAX: double[CONSTRAINT]
+ maxPopulation(clusters: Cluster[]): long
+ minPopulation(clusters: Cluster[]): long
+ isHopeless(cluster: Cluster): boolean
+ addPopulation(pop1: long, pop2: long):long
+ calculatePercentage(demoPop: long, totalPop: long): double
+ checkPercentThreshold(minorityPercent: double, Properties.thresholds[Constraint.MINORITYPERCENTAGE] Tuple<double, double>): double
+ getGraphCompactness(c: Cluster, Properties.thresholds[Constraint.G_COMPACT]: Tuple<double, double>): double
+ getPolbyPopperCompactness(c: Cluster, Properties.thresholds[Constraint.P_COMPACT]: Tuple<double, double>): double
+ getSchwartzbergCompactness(c: Cluster, Properties.thresholds[Constraint.S_COMPACT]: Tuple<double, double>): double
+ getEfficiencyGap(c: Cluster, Properties.thresholds[Constraint.EFF_GAP]: Tuple<double, double>): double
+ getLopsidedMargins(c: Cluster, Properties.thresholds[Constraint.LOP_MARG]: Tuple<double, double>): double
+ getMeanMedianDifference(c: Cluster, Properties.thresholds[Constraint.MM_DIFF]: Tuple<double, double>): double
+ getEqualPopulation(c: Cluster, Properties.thresholds[Constraint.EQ_POP]: Tuple<double, double>): double
+ normalize(val: double, MIN[Constraint]: double, MAX[Constraint]: double): double
+ getWeighted(normalized: double[], weights: double[]): double[]
+ getMajMinScore(population: int, thresholds: Tuple<double,double>): double
Algorithm
- config: Properties
- map: State
- sessionId: String
- startTime: DateTime
- state: JobState
+ start(): List<Cluster>
+ startAsync(): List<Cluster>
+ startIncremental(Config: Properties, s : State): List<Cluster>
+ runStep(): List<Cluster>
State
- eligibleForMajorityMinority:boolean
- stateBoundaries: json
- clusters: List<Cluster>
- precincts: Set<Precincts>
+ getClusters(): List<Cluster>
+ getEligibleDemographicVotingBlocs(demographicGroup: DemographicGroup, config: Properties): Set<VotingBlocDTO>
+ joinClusters(): boolean
+ removeCluster(c: Cluster): boolean
+ getMMEdges(config : Properties): Set<Edge>
+ getNumClusters(): int
+ getMMEdges(config : Properties) : Set<Edge>
+ getEdges() : Set<Edge>
Cluster
- precincts: List<Precinct>
- neighbors: List<Cluster>
- edges: Set<Edge>
+ merge(other: Cluster): boolean
+ getPrecincts(): List<Cluster>
+ getAdjacentClusters(): Set<Cluster>
+ getAdjacentPrecincts(): Set<Precinct>
+ addEdges(edges: Set<Edge>): boolean
+ removeEdges(edges: Set<Edge>): boolean
+ getEdges(): Set<Edge>
+ addNeighbors(clusters: List<Cluster>): boolean
+ movePrecinct(p: Precinct, other: Cluster): boolean
+ peekMovePrecinct(p: Precinct, other: Cluster): Edge
+ toDistrict(): District
+ getMMEdges(config : Properties): Set<Edge>
2..*
0..*
Edge
+ clusterA: Cluster
+ clusterB: Cluster
- joinability: double
+ calculateObjectiveFunction(): double
+ getTotalPopulation(pop1: long, pop2: long): long
+ wouldImproveMM(Config: Properties) : boolean
Properties
+ thresholds: Tuple<double, double>[Constraint]
+ weights: double[Constraint]
+ minorities: double[DemographicGroup]
+ incremental: boolean
+ realtime: boolean
+ numDistricts: int
+ numMajMinDistricts: int
+ selectedMinorities: boolean[DemographicGroup]
Precinct
- electionData: ElectionData
- population: long[DemographicGroup]
- precinctBoundaries: json
+ getNeighbors(): List<Precinct>
+ checkSameCounty(p1: Precinct, p2: Precinct): boolean
+ isDemographicBloc(Properties.thresholds[Constraint.BLOCPOPULATIONPERCENTAGE]: Tuple<double, double>): boolean
+ getVotingBloc(Properties.thresholds[Constraint.BLOCVOTINGPERCENTAGE]: Tuple<double, double>): VotingBlocDTO
0..*
1
<<enumeration>>
Constraint
<<Constant>> -G_COMPACT
<<Constant>> -P_COMPACT
<<Constant>> -S_COMPACT
<<Constant>> -EFF_GAP
<<Constant>> -LOP_MARG
<<Constant>> -MM_DIFF
<<Constant>> -EQ_POP
<<Constant>> -BLOCPOPULATIONPERCENTAGE
<<Constant>> -BLOCVOTINGPERCENTAGE
<<Constant>> -MINORITYPERCENTAGE
ElectionData
- electionType: ElectionType
- year: Year
- votesByParty: long[PoliticalParty]
- votesByDemo: long[DemographicGroup]
- incumbents: String[PoliticalParty]
+ aggFromJurisdictions(jurisdictions: List<IJurisdiction>): ElectionData
+ getWinningParty(): PolitiscalParty
+ getVotes(PoliticalParty): int
+ getTotalVotes(): int
<<enumeration>>
Year
<<Constant>> -2016
<<Constant>> -2018
<<Constant>> - COMBINED
VotingBlocDTO
+ precinctId: int
+ totalVotes: int
+ winningVotes: int
+ winningParty: PoliticalParty
1..*
1
1..*
1
<<enumeration>>
ElectionType
<<Constant>> -PRESIDENTIAL
<<Constant>> -CONGRESSIONAL
<<Constant>> -COMBINED
<<enumeration>>
PoliticalParty
<<Constant>> -DEMOCRAT
<<Constant>> -REPUBLICAN