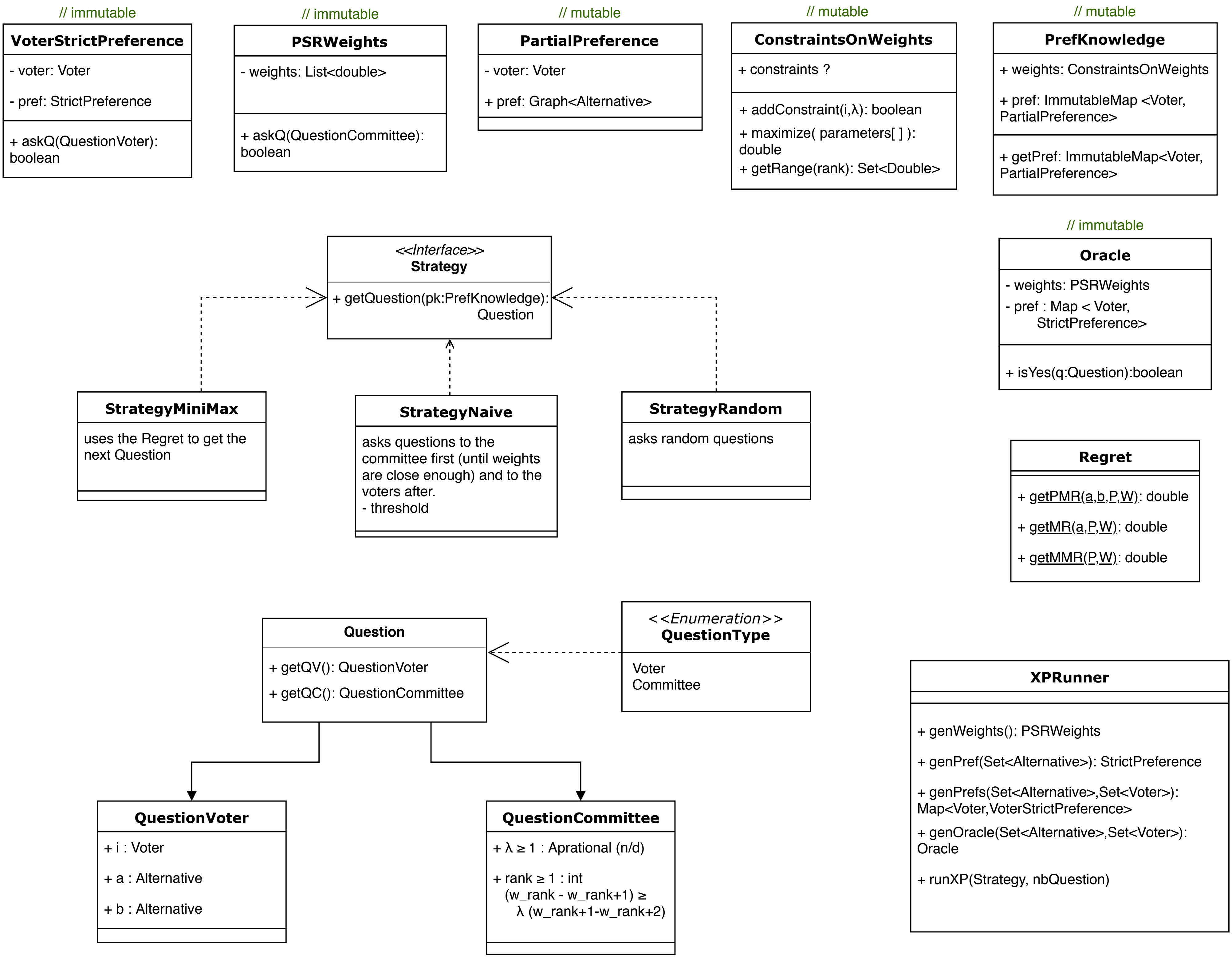

// immutable

VoterStrictPreference

- voter: Voter
- pref: StrictPreference

+ askQ(QuestionVoter): boolean

// immutable

PSRWeights

- weights: List<double>

+ askQ(QuestionCommittee): boolean

// mutable

PartialPreference

- voter: Voter

+ pref: Graph<Alternative>

// mutable

ConstraintsOnWeights

+ constraints ?

+ addConstraint(i,λ): boolean

+ maximize(parameters[ ]): double

+ getRange(rank): Set<Double>

// mutable

PrefKnowledge

+ weights: ConstraintsOnWeights

+ pref: ImmutableMap <Voter, PartialPreference>

+ getPref: ImmutableMap<Voter, PartialPreference>

<<Interface>>
Strategy

+ getQuestion(pk:PrefKnowledge): Question

StrategyMiniMax

uses the Regret to get the next Question

StrategyNaive

asks questions to the committee first (until weights are close enough) and to the voters after.
- threshold

StrategyRandom

asks random questions

// immutable

Oracle

- weights: PSRWeights
- pref : Map < Voter, StrictPreference>

+ isYes(q:Question):boolean

Regret

+ getPMR(a,b,P,W): double

+ getMR(a,P,W): double

+ getMMR(P,W): double

Question

+ getQV(): QuestionVoter

+ getQC(): QuestionCommittee

<<Enumeration>>
QuestionType

Voter
Committee

QuestionVoter

+ i : Voter

+ a : Alternative

+ b : Alternative

QuestionCommittee

+ λ ≥ 1 : Aprational (n/d)

+ rank ≥ 1 : int
(w_rank - w_rank+1) ≥
λ (w_rank+1-w_rank+2)

XPRunner

+ genWeights(): PSRWeights

+ genPref(Set<Alternative>): StrictPreference

+ genPrefs(Set<Alternative>,Set<Voter>): Map<Voter,VoterStrictPreference>

+ genOracle(Set<Alternative>,Set<Voter>): Oracle

+ runXP(Strategy, nbQuestion)