TournamentInterfacePort

groups: Grouplist
timeM: TimeManager
teams: TeamList

nextDay()
getDate(): Date
getGroupInfo(group: string): string[]
getGroupsInfo(): string[]
getTeamInfo(team: string): string[]
getNextMatches(): string[]

TimeManager:

curDate: Date

elements: AffectedByTime[]
TimeManager(elements_: AffectedByTime)
nextDayAll()

Resolve returns a new
FilledPosition if:
*The Position is FinalPosition it returns self
*The Position is UnfilledPosition and its Rank
 is final it retund a new FilledPosition.
 This copies rank.rankInfo intopositon info
otherwise it returns NONE

«interface»
AffectedByTime

The number of positions
equals the number
of Ranks

TeamList

getTeamInfo(team: string)
isEligible(team: string, player: string): bool

«interface»
TeamTurnamentInterface

GroupList

getGroupInfo(group: string): string[]
getGroupsInfo(): string[]
getNextMatches(): string[]

groups

AbstractGroup

groupname: string {id}
curDate: date
groupZone: string
groupPhase: string
groupName: string

nextDay()
getGroupInfo(): string[]
getNextMatches(): string[]

positions

Position

reports: MatchReport[]

resolve():FinalPosition

ranks

Rank

isFinal:bool
info: RankInfo

position

rank

UnfilledPosition

FilledPosition

positionInfo: RankInfo

team

Team

name: string {id}

recordMatch(yellowCards: integer[], redCards: integer[])
isEligible(player: string): bool

teams

«interface»
TurnamentRankingInterfaceRanking

SetDrawPots

drawDate: Date
ranking: TournamentRankingInterface

^nextDay()

PlayGroup

rankingType: RankingStrategy

^nextDay()
^getGroupInfo(): string[]
^getNextMatches(): string[]

matches

Match

matchDate: Date
sides: Position[0..1]
report: MatchReport
generator:TournamentMatchInterface
FIFARanking: TournamentRankingInterface

play()

When matches are
played all Positions
are FilledPositions

PlayerDisciplinaryRecord

player: integer {id}
yellowCards: integer
redCards: integer

recordMatch(yellowCards: integer, redCards: integer)
isEligible(): bool

Draw

drawDate: Date

^nextDay()

The size of possitionsEquivalence
and rankEquivalence equal
the size of positions and ranks

«interface»
RankingStrategy

rank(positions: Position[]): map<string,RankInfo

«interface»
TurnamentMatchInterfaceRecord

«interface»
TurnamentRankingInterfacePosition

teamPosition(string): int

rules
1

DrawRules

positionsEquivalence[]
rankEquivelence[]

makeDraw(): int[0..1][]

The mulisets positionsEquio\valence
and rankEquivalence are equal.

«datatype»
MatchReport

teams: string[0..1][]
goals: int[0..1]
yellowCards: int[0..1][]
redCards: int[0..1][]
report: string[]

«datatype»
MatchRules

extraTimePossible: bool
extraTimeResult: int[0..1]

«datatype»
RankInfo

position: int
points: int
goalsFor: int
goalsAgainst: int
yellowCards: int
redCards: int
pointsSpecial: int
goalsForSpecial: int
goalsAgainstSpecial: int

FIFARankingStrategy

FIFARanking: TournamentRankingInterfaceRecord

RankAccrosGroupsStrategy

FIFARanking: TournamentRankingInterfaceRecord

Special points and score
is used in case when
the results against last
placed team is not counted
to determine the ranking
among different groups

EliminationRankingStrategy