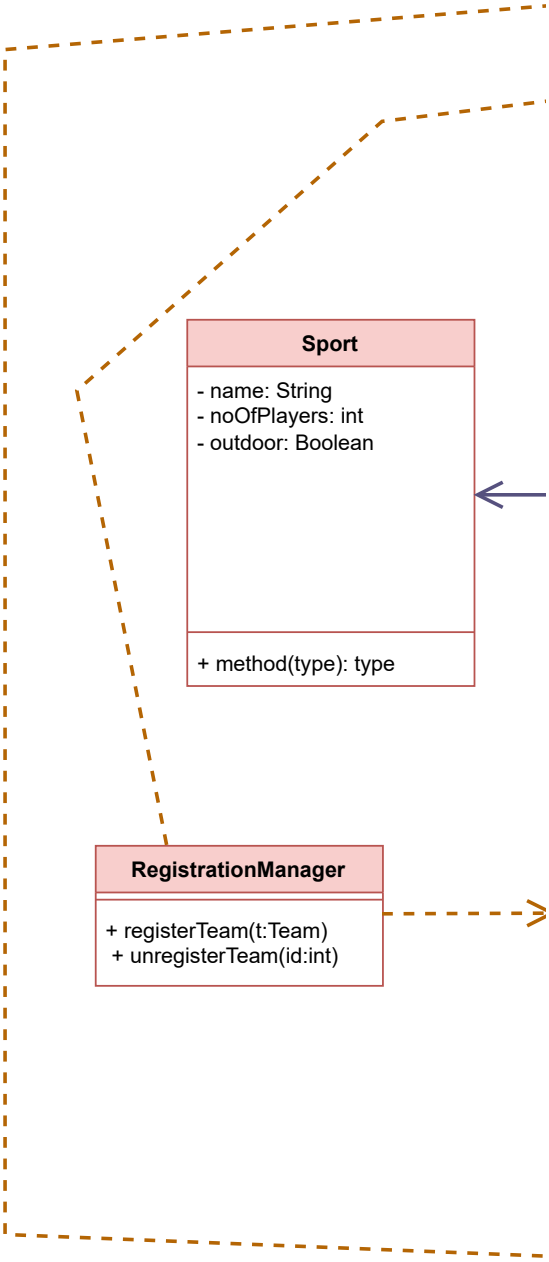
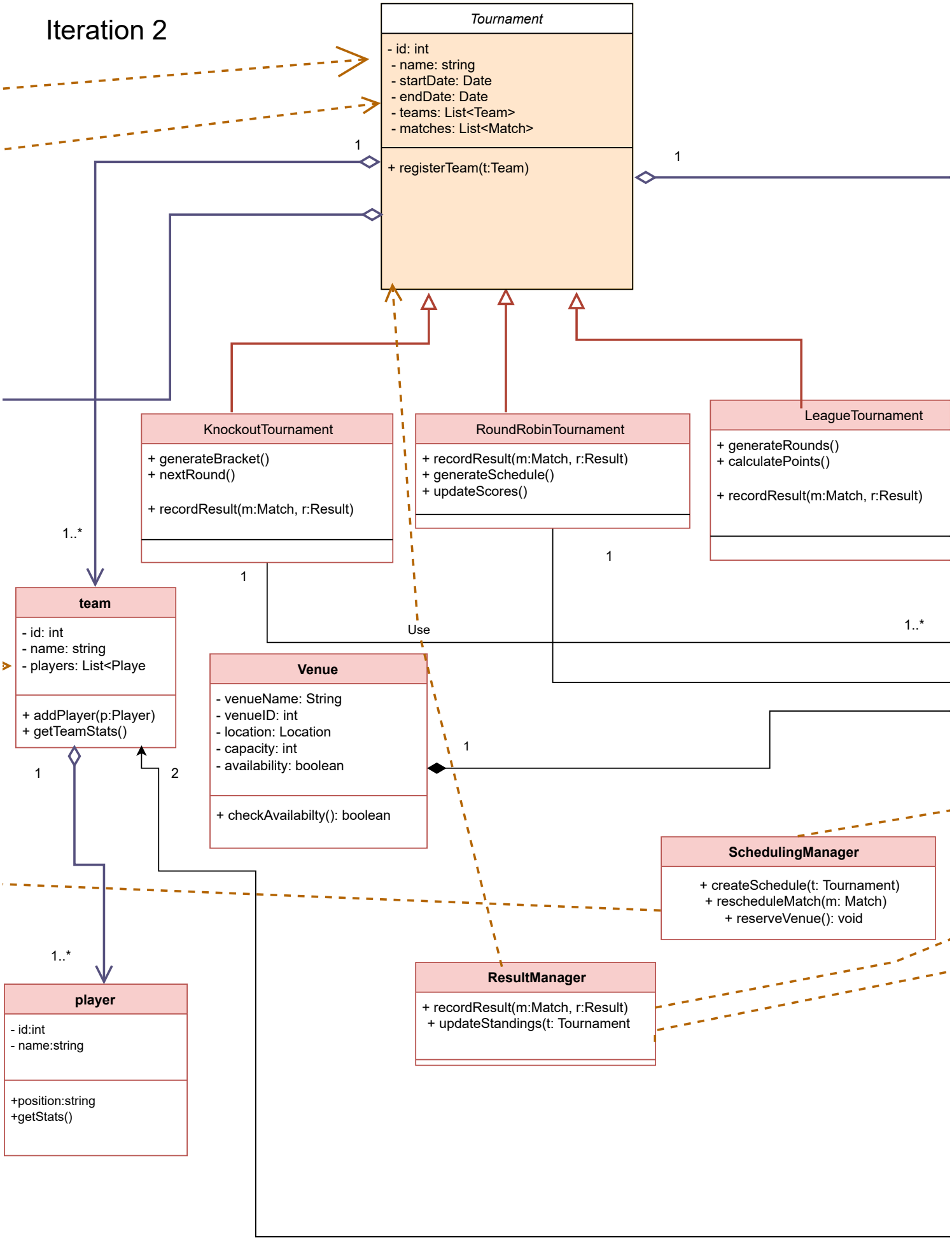
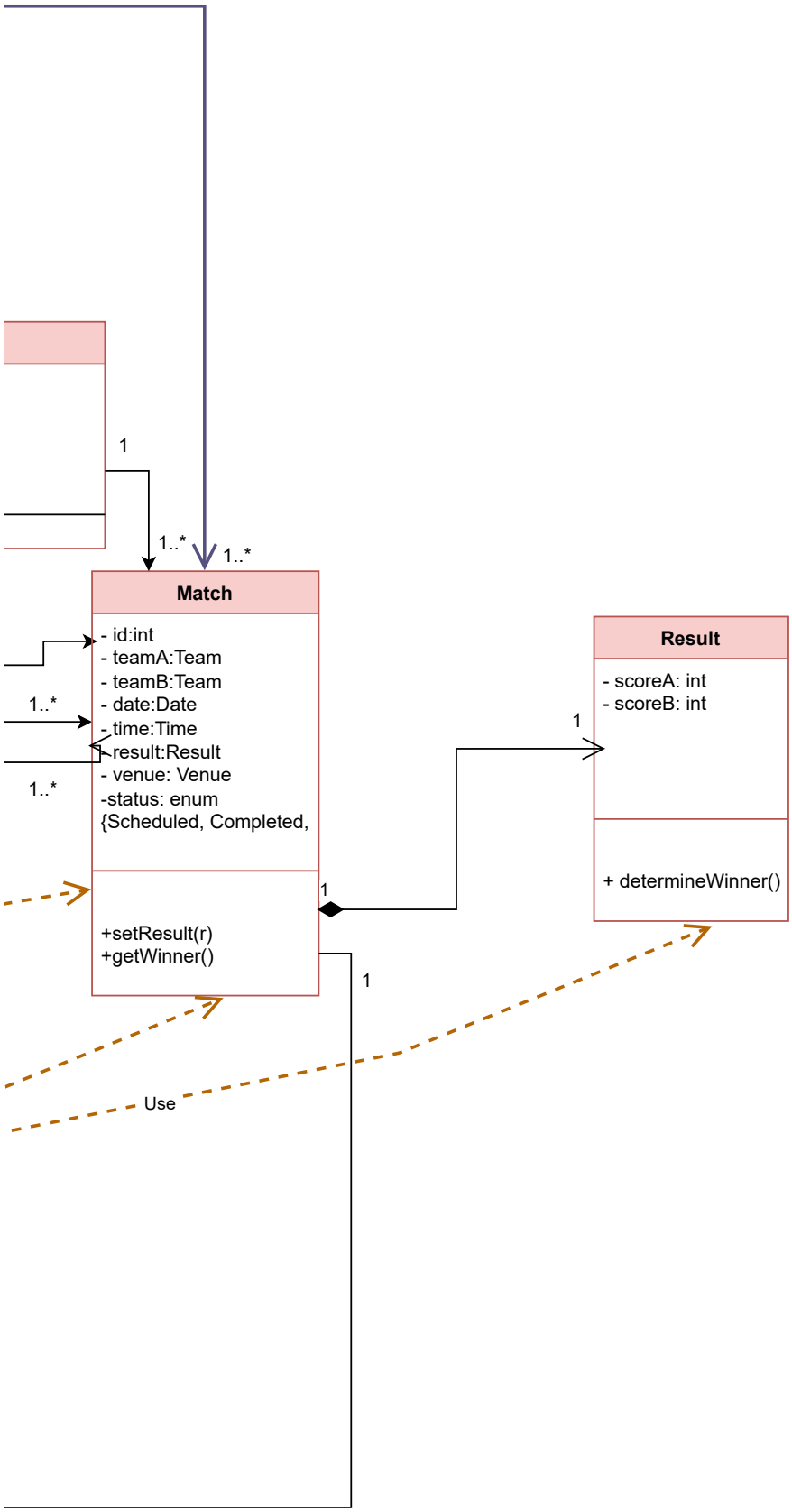
Use


Sport
- name: String
- noOfPlayers: int
- outdoor: Boolean
+ method(type): type
RegistrationManager
+ registerTeam(t:Team)
+ unregisterTeam(id:int)


Iteration 2
Tournament
- id: int
- name: string
- startDate: Date
- endDate: Date
- teams: List<Team>
- matches: List<Match>
+ registerTeam(t:Team)
1
1
KnockoutTournament
+ generateBracket()
+ nextRound()
+ recordResult(m:Match, r:Result)
RoundRobinTournament
+ recordResult(m:Match, r:Result)
+ generateSchedule()
+ updateScores()
LeagueTournament
+ generateRounds()
+ calculatePoints()
+ recordResult(m:Match, r:Result)
1..*
1
1
1..*
team
- id: int
- name: string
- players: List<Playe
+ addPlayer(p:Player)
+ getTeamStats()
Use
Venue
- venueName: String
- venueID: int
- location: Location
- capacity: int
- availability: boolean
+ checkAvailabilty(): boolean
1
1
2
SchedulingManager
+ createSchedule(t: Tournament)
+ rescheduleMatch(m: Match)
+ reserveVenue(): void
1..*
ResultManager
+ recordResult(m:Match, r:Result)
+ updateStandings(t: Tournament
player
- id:int
- name:string
+position:string
+getStats()


1
1..*
1..*
Match
- id:int
- teamA:Team
- teamB:Team
- date:Date
- time:Time
-result:Result
- venue: Venue
-status: enum
{Scheduled, Completed,
+setResult(r)
+getWinner()
1..*
1..*
1
1
Result
- scoreA: int
- scoreB: int
+ determineWinner()
1
Use