# Getting Ready: The Cricinfo System

Understand the Cricinfo problem and learn the questions to simplify this problem further.

**We'll cover the following**

* [Problem definition](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Problem-definition)
* [Expectations from the interviewee](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Expectations-from-the-interviewee)
  + [Live matches](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Live-matches)
  + [Statistics](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Statistics)
  + [People](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#People)
  + [Tournament](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Tournament)
* [Design approach](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-approach)
* [Design pattern](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-pattern)

## Problem definition

**Cricinfo** also known as ESPNcricinfo, is one of the leading cricket-related websites in the world. The platform provides a live coverage of cricket matches, along with ball-by-ball commentary. This website has a database including historical matches from the 18th century till the present. This website provides cricket content globally and is followed by millions around the world. Users can find match updates, live cricket scores, match results on this website. This website also provides articles and news about cricket. Moreover, people can search for matches, teams, players, tournaments, etc. In short, this is a website for cricket lovers.

## Expectations from the interviewee

Cricinfo provides multiple functionalities to its users. It is essential to narrow down what components you will include in your Cricinfo design. The section below provides an overview of some of the main expectations that the interviewer will want to hear you discuss in more detail, during the interview.

### Live matches

Coverage of a live match is one of the most important features of a cricket website. The user must be able to see the live matches on the platform. To get a better understanding of live matches, you can ask interviewer questions like this:

* How is information of the match updated?
* What is the process for adding commentary?
* At what intervals is the commentary added?

### Statistics

It is crucial to store the statistics of the matches in the Cricinfo system. Therefore, you can ask the following questions:

* What kind of stats does the system record?
* Does the system record the statistics of players and teams?

### People

There are different people involved in cricket matches. Therefore, to get an idea of the type of people in the system, you can ask the following questions:

* What kind of people does our system need to store the information about, for example, players, coaches, etc?
* Will the umpire data be stored in the system?
* Who will input data into the Cricinfo system? What are the actors of the Cricinfo system?

### Tournament

A **tournament** is a series of matches played between a number of teams. Therefore, the interviewer would expect you to ask the following questions:

* How does the system keep track of different tournaments?
* How are the tournament points and rankings stored in the system?

## Design approach

We’ll design Cricinfo using the bottom-up approach. For this purpose, we will follow the steps below:

* Identify and design the smallest components first, like a ball and run.
* Use these small components to design bigger components, for example, an over, team, and an umpire.
* Repeat the steps above until we design the complete Cricinfo platform.

## Design pattern

It is always a good practice to discuss the design patterns that an airline management system falls under, during the interview. Stating the design patterns will give the interviewer a positive impression and shows that the interviewee is well-versed in the advanced concepts of object-oriented design. We can use the Factory design pattern to design Cricinfo.

Let’s explore the requirements for Cricinfo in the next lesson.

# Requirements for Cricinfo

Learn about all requirements of Cricinfo.

**We'll cover the following**

* [Requirements collection](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Requirements-collection)

In this lesson, we’ll list the requirements of Cricinfo. This is a very crucial step, since requirements define the scope of a problem. Therefore, getting them right from the interviewer and understanding them well will make the design of the rest of the system smooth and easy.

We’ll use the notational convention to identify each requirement with a unique label, "Rn" where "R" is short for Requirement and "n" is a natural number.

## Requirements collection

The requirements for Cricinfo are defined below:

**R1:** The system should be able to track the stats of all players, teams, and matches.

**R2:** The system should be able to track all scores or wickets that occurred for each ball. The system should also provide a live commentary for every ball.

**R3:** The system should be able to keep track of all matches—Test, T20, and ODI matches.

**R4:** The system should be able to keep track of ongoing and previous tournaments. The system should also be able to show a points table for all teams participating in a tournament.

**R5:** The system should be able to show the result of all previous televised matches.

**R6:** All teams should be able to select some players that will participate in the tournament.

**R7:** For every match, the teams should be able to select 11 players to play on the field, known as the **playing eleven**.

**R8:** The admin of the system should be able to add tournaments, matches, teams, players, and news to the system.

We've identified our requirements for the problem. In the next lesson, we will define different use cases of Cricinfo.

# Use Case Diagram for Cricinfo

Learn how to define use cases and create the corresponding use case diagram for the Cricinfo problem.

**We'll cover the following**

* [System](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#System)
* [Actors](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Actors)
  + [Primary actors](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Primary-actors)
  + [Secondary actors](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Secondary-actors)
* [Use cases](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Use-cases)
  + [Admin](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Admin)
  + [Commentator](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Commentator)
* [Relationships](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Relationships)
  + [Generalization](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Generalization)
  + [Associations](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Associations)
  + [Extend](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Extend)
* [Use case diagram](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Use-case-diagram)

Let's build the use case diagram of Cricinfo and understand the relationship between its different components.

First, we will define the different elements of our Cricinfo system, followed by the complete use case diagram of the system.

## System

Our system is "Cricinfo."

## Actors

Now, we’ll define the main actors of Cricinfo.

### Primary actors

* **Admin:** The admin is in charge of performing numerous operations—adding or modifying tournaments, innings, updating stats, etc.
* **Commentator:** This actor can add commentary to the match or modify it.

### Secondary actors

There is no secondary actor in the system.

## Use cases

In this section, we’ll define the use cases for Cricinfo. We have listed the use cases according to their respective interactions with a particular actor.

**Note:** You’ll see some use cases occurring multiple times because they are shared among different actors in the system.

### Admin

* **Add/modify team:**To add a team in the system or modify it
* **Add/modify player:**To add a player to the team or modify it
* **Add/modify tournament:**To add a tournament in the system or modify it
* **Add/modify team squad:**To add a team squad or modify it
* **Add/modify innings:** To add innings in the match or modify it
* **Add/modify over:** To add an over in the match or modify it
* **Add/modify ball:**To add a ball in the match or modify it
* **Add match:** To add a match in the system
* **Add/update news**: To add the match news in the system or update it
* **Add/modify stadium:**To add a stadium in the system or modify it
* **Add/modify umpire:** To add an umpire in the system or modify it
* **Add/update stats:** To add stats of a player, match, or tournament to the system

### Commentator

* **Add/modify commentary:** To add a commentary to the match or modify it

## Relationships

We describe the relationships between and among actors and their use cases in this section.

### Generalization

* The admin can add/update stats by adding/updating player, match, or tournament stats. This shows “Add/update stats” use case has a generalization relationship with the “Add/update player stats,” “Add/update match stats,” and “Add/update tournament stats” use cases.
* The admin can add the match type—T20, test, or ODI. Hence, the “Add match” use case has a generalization relationship with the “Add T20,” “Add test,” and “Add ODI” use cases.

### Associations

The table below shows the association relationship between actors and their use cases.

|  |  |
| --- | --- |
| **Admin** | **Commentator** |
| Add/modify team | Add/modify commentary |
| Add/modify player |  |
| Add/modify tournament |
| Add/modify team squad |
| Add/modify innings |
| Add/modify over |
| Add/modify ball |
| Add match |
| Add/update news |
| Add/modify stadium |
| Add/modify umpire |
| Add/update stats |

### Extend

* While adding/modifying the ball, either a run, wicket, or both are added/modified. Therefore, the “Add/modify ball” use case has an extend relationship with the “Add/modify run” and “Add/modify wicket” use cases.

## Use case diagram

Here’s the use case diagram of Cricinfo:

A diagram of a diagram

Description automatically generated

# Class Diagram for Cricinfo

Understand how to create a class diagram for Cricinfo using the bottom-up approach.

**We'll cover the following**

* [Components of Cricinfo](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Components-of-Cricinfo)
  + [Admin](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Admin)
  + [Run, ball, and wicket](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Run,-ball,-and-wicket)
  + [Over and innings](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Over-and-innings)
  + [Match](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Match)
  + [Stadium](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Stadium)
  + [Player, coach, and umpire](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Player,-coach,-and-umpire)
  + [Team, tournament squad, and playing eleven](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Team,-tournament-squad,-and-playing-eleven)
  + [Tournament and points table](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Tournament-and-points-table)
  + [Stats](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Stats)
  + [Commentator and commentary](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Commentator-and-commentary)
  + [News](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#News)
  + [Enumerations](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Enumerations)
  + [Custom data type](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Custom-data-type)
* [Relationship between classes](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Relationship-between-classes)
  + [Association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Association)
    - [One-way association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#One-way-association)
    - [Two-way association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Two-way-association)
  + [Aggregation](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Aggregation)
  + [Composition](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Composition)
  + [Inheritance](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Inheritance)
* [Class diagram of Cricinfo](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Class-diagram-of-Cricinfo)
* [Design pattern](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-pattern)

In this lesson, we’ll identify and design the classes, abstract classes, and interfaces based on the requirements we have previously gathered from the interviewer in our Cricinfo system.

## Components of Cricinfo

As mentioned earlier, we will design the Cricinfo system using a bottom-up approach.

### Admin

The Admin class is responsible for managing the system as well as adding and modifying updates. The representation of the class is shown below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a sports program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### Association

The class diagram has the following association relationships:

#### One-way association

* The Admin class has a one-way association with the Player, Team, Match, and Tournament classes.
* The Player class has a one-way association with the Run, Ball, Wicket, and Over classes.
* The Team class has a one-way association with the TournamentSquad and Tournament classes.
* The TournamentSquad class has a one-way association with the Playing11 class.

#### Two-way association

* The Ball class is associated with the Run, Wicket, and Commentary classes.
* The Team class is associated with the Coach and News classes.
* The Commentary class is associated with the Commentator class.
* The Match class is associated with the Umpire, Commentator, and Stadium classes.

A diagram of a tournament

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A diagram of a game

Description automatically generated

## Design pattern

In the Cricinfo system, we need to create different types of matches, tournaments, and squads at runtime. To do this, we can use the Factory design pattern. This pattern provides a way to create objects without specifying the exact class of object that will be created.

We have completed the class diagram of Cricinfo according to the requirements. Now, let’s design the sequence diagram of Cricinfo in the next lesson.

# Sequence Diagram for Cricinfo

Create a sequence diagram for adding a match to the Cricinfo system.

**We'll cover the following**

* [Add a match](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Add-a-match)

Sequence diagrams are a great way to understand the interactions between different entities and objects in the system. There can be different sequence diagrams that we can create for the Cricinfo system. In this lesson, we will create a sequence diagram for the following interaction:

* **Add a match:**The admin adds a new match to the system.

## Add a match

The sequence diagram for adding a match should have the following actors and objects that will interact with each other:

* **Actors:** Admin, Umpire, and Commentator
* **Objects:** Match and Stadium

The steps involved in adding a match are listed below:

1. The admin creates a new match of a specific match type.
2. The admin adds the playing teams for the match.
3. The admin assigns a stadium to the match.
4. The admin assigns an umpire to the match.
5. The admin assigns a commentator to the match.

Based on the order above, the sequence diagram for adding a match in the Cricinfo system is provided below:

A diagram of a graph

Description automatically generated

Next, let's look at the activity diagrams for Cricinfo to understand the control flow of the system.

# Activity Diagram for Cricinfo

Create some activity diagrams for the Cricinfo.

**We'll cover the following**

* [Make a record of a ball](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Make-a-record-of-a-ball)
  + [States](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#States)
  + [Actions](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Actions)

Activity diagrams are a great way to visualize the flow of messages from one activity to the other in the system. There can be different activity diagrams that we can create for Cricinfo. In this lesson, we will create an activity diagram for making a record of a ball.

## Make a record of a ball

The states and actions that will be involved in this activity diagram are provided below.

### States

**Initial state:**The system adds a ball to the over.

**Final state:** The ball record is saved.

### Actions

The system adds a ball to the over. The system chooses the ball type that was bowled. The system determines if the batter got out. If they did not, it adds the score that was made. Finally, the commentary is added to the ball, and the ball record is saved.

A diagram of a ball

Description automatically generated

# Code for Cricinfo

Write object-oriented code to implement the design of the Cricinfo problem.

**We'll cover the following**

* [Cricinfo classes](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Cricinfo-classes)
  + [Constants](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Constants)
  + [Admin, player, coach, and umpire](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Admin,-player,-coach,-and-umpire)
  + [Run, ball, wicket, over, and innings](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Run,-ball,-wicket,-over,-and-innings)
  + [Match](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Match)
  + [Team, tournament squad, and playing11](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Team,-tournament-squad,-and-playing11)
  + [Tournament, points table, and stadium](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Tournament,-points-table,-and-stadium)
  + [Commentator, commentary, and news](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Commentator,-commentary,-and-news)
  + [Statistics](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Statistics)
* [Wrapping up](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Wrapping-up)

We've gone over the different aspects of Cricinfo and observed the attributes attached to the problem using various UML diagrams. Let's now explore the more practical side of things, where we will work on implementing the Cricinfo problem using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in Cricinfo:

* Java
* C#
* Python
* C++
* JavaScript

## Cricinfo classes

In this section, we will provide the skeleton code of the classes designed in the class diagram lesson.

**Note:** For simplicity, we are not defining getter and setter functions. The reader can assume that all class attributes are private and accessed through their respective public getter methods and modified only through their public method functions.

### Constants

The following code defines the various enums and custom data types used in the Cricinfo design:

**Note:** JavaScript does not support enumerations. We’ll usethe Object.freeze() method as an alternative. It freezes an object and prevents further modifications.

public class Address {

private int zipCode;

private String streetAddress;

private String city;

private String state;

private String country;

}

enum MatchResult {

LIVE,

BAT\_FIRST\_WIN,

FIELD\_FIRST\_WIN,

DRAW,

CANCELED

}

enum UmpireType {

FIELD,

RESERVED,

THIRD\_UMPIRE

}

enum WicketType {

BOLD,

CAUGHT,

STUMPED,

RUN\_OUT,

LBW,

RETIRED\_HURT,

HIT\_WICKET,

OBSTRUCTING,

HANDLING

}

enum BallType {

NORMAL,

WIDE,

NO\_BALL,

WICKET

}

enum RunType {

NORMAL,

FOUR,

SIX,

LEG\_BYE,

BYE,

NO\_BALL,

OVERTHROW

}

enum PlayingPosition {

BATTING,

BOULING,

ALL\_ROUNDER

}

### Admin, player, coach, and umpire

The definitions of the Admin, Player, Coach and Umpire classes are as follows:

public class Admin {

public boolean addPlayer(Player player);

public boolean addTeam(Team team);

public boolean addMatch(Match match);

public boolean addTournament(Tournament tournament);

public boolean addStats(Stat stats);

public boolean addNews(News news);

}

public class Player {

private String name;

private int age;

private int country;

private PlayerPosition position;

private List<Team> teams;

private PlayerStat stat;

}

public class Coach {

private String name;

private int age;

private int country;

private List<Team> teams;

}

public class Umpire {

private String name;

private int age;

private int country;

public boolean assignMatch(Match match);

}

### Run, ball, wicket, over, and innings

In cricket, the mandatory concepts can be of five types: run, ball, wicket, over, and innings. To store information about these identities, we defined the Run, Ball, Wicket, Over, and Innings classes as shown below:

public class Run {

private int totalRuns;

private RunType type;

private Player scoredBy;

}

public class Ball {

private Player balledBy;

private Player playedBy;

private BallType type;

private List<Run> runs;

private Wicket wicket;

public boolean addCommentary(Commentary commentary);

}

public class Wicket {

private WicketType type;

private Player playerOut;

private Player balledBy;

private Player caughtBy;

private Player runoutBy;

private Player stumpedBy;

}

public class Over {

private int number;

private Player bowler;

private int totalScore;

private List<Ball> balls;

public boolean addBall(Ball ball);

}

public class Innings {

private Playing11 bowling;

private Playing11 batting;

private Date startTime;

private Date endTime;

private int totalScores;

private int totalWickets;

private List<Over> overs;

public boolean addOver(Over over);

}

### Match

The Match is an abstract class representing matches in Cricinfo. Matches can be of three types: T20, Test, and ODI. The implementation of these classes is given below:

public abstract class Match {

private Date startTime;

private MatchResult result;

private int totalOvers;

private List<Playing11> teams;

private List<Innings> innings;

private Playing11 tossWin;

private Map<Umpire, UmpireType> umpires;

private Stadium stadium;

private List<Commentator> commentators;

private List<MatchStat> stats;

public abstract boolean assignStadium(Stadium stadium);

public abstract boolean assignUmpire(Umpire umpire);

}

public class T20 extends Match {

public boolean assignStadium(Stadium stadium);

public boolean assignUmpire(Umpire umpire);

}

public class Test extends Match {

public boolean assignStadium(Stadium stadium);

public boolean assignUmpire(Umpire umpire);

}

public class ODI extends Match {

public boolean assignStadium(Stadium stadium);

public boolean assignUmpire(Umpire umpire);

}

### Team, tournament squad, and playing11

The definitions of the AdminTeam, TournamentSquad, and Playing11 classes are as follows:

public class Team {

private String name;

private List<Player> players;

private Coach coach;

private List<News> news;

private TeamStat stats;

public boolean addSquad(TournamentSquad squad);

public boolean addPlayer(Player player);

public boolean addNews(News news);

}

public class TournamentSquad {

private List<Player> players;

private Tournament tournament;

private List<TournamentStat> stats;

public boolean addPlayer(Player player);

}

public class Playing11 {

private List<Player> players;

public boolean addPlayer(Player player);

}

### Tournament, points table, and stadium

The definitions of the Tournament, PointsTable, and Stadium classes are as follows:

public class Tournament {

private Date startDate;

private List<TournamentSquad> teams;

private List<Match> matches;

private PointsTable points;

public boolean addTeam(TournamentSquad team);

public boolean addMatch(Match match);

}

public class PointsTable {

private HashMap<String, float> teamPoints;

private HashMap<Team, MatchResult> matchResults;

private Tournament tournament;

private Date lastUpdated;

}

public class Stadium {

private String name;

private Address location;

private int maxCapacity;

}

A screenshot of a computer

Description automatically generated

The Stat is an abstract class which represents the statistics in the Cricinfo. Statistics can be of three types: PlayerStat, MatchStat, and TeamStat. These classes are represented below:

public abstract class Stat {

public abstract boolean updateStats();

}

public class PlayerStat extends Stat {

private int ranking;

private int bestScore;

private int bestWicketCount;

private int totalMatchesPlayed;

private int total100s;

private int totalHattricks;

public boolean updateStats();

}

public class MatchStat extends Stat {

private double winPercentage;

private Player topBatsman;

private Player topBowler;

public boolean updateStats();

}

public class TeamStat extends Stat {

private int totalSixes;

private int totalFours;

private int totalReviews;

public boolean updateStats();

}

**Wrapping up**

We've explored the complete design of Cricinfo in this chapter. We've looked at how Cricinfo can be visualized using various UML diagrams and designed using object-oriented principles and design patterns.