Back To Course Home

Grokking the Object Oriented Design Interview

0% completed

Search Course

Object-Oriented Design and UML

- Object-Oriented Basics
- OO Analysis and Design
- What is UML?
- Use Case Diagrams
- Class Diagram
- Sequence diagram
- Activity Diagrams

Object Oriented Design Case Studies

- Design a Library Management System
- Design a Parking Lot
- Design Amazon - Online Shopping System
- Design Stack Overflow
- Design a Movie Ticket Booking System
- Design an ATM
- Design an Airline Management System
- Design Blackjack and a Deck of Cards
- Design a Hotel Management System
- Design a Restaurant Management system
- Design Chess
- Design an Online Stock Brokerage System
- Design a Car Rental System
- Design LinkedIn
- Design Cricinfo
- Design Facebook - a social network

Appendix

- Contact us
- Other courses

Course Certificate

Mark Course as Completed

Design Cricinfo

Let's design Cricinfo.

We'll cover the following

- System Requirements
- Use case diagram
- Class diagram
- Activity diagrams
- Code

Cricinfo is a sports news website exclusively for the game of cricket. The site features live coverage of cricket matches containing ball-by-ball commentary and a database for all the historic matches. The site also provides news and articles about cricket.

System Requirements

We will focus on the following set of requirements while designing Cricinfo:

1. The system should keep track of all cricket-playing teams and their matches.
2. The system should show live ball-by-ball commentary of cricket matches.
3. All international cricket rules should be followed.
4. Any team playing a tournament will announce a squad (a set of players) for the tournament.
5. For each match, both teams will announce their playing-eleven from the tournament squad.
6. The system should be able to record stats about players, matches, and tournaments.
7. The system should be able to answer global stats queries like, "Who is the highest wicket taker of all time?", "Who has scored maximum numbers of 100s in test matches?", etc.
8. The system should keep track of all ODI, Test and T20 matches.

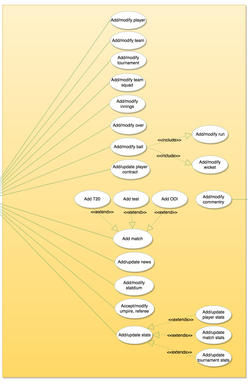
Use case diagram

We have two main Actors in our system:

- **Admin:** An Admin will be able to add/modify players, teams, tournaments, and matches, and will also record ball-by-ball details of each match.
- **Commentator:** Commentators will be responsible for adding ball-by-ball commentary for matches.

Here are the top use cases of our system:

- **Add/modify teams and players:** An Admin will add players to teams and keeps up-to-date information about them in the system.
- **Add tournaments and matches:** Admins will add tournaments and matches in the system.
- **Add ball:** Admins will record ball-by-ball details of a match.
- **Add stadium, umpire, and referee:** The system will keep track of stadiums as well as of the umpires and referees managing the matches.
- **Add/update stats:** Admins will add stats about matches and tournaments. The system will generate certain stats.
- **Add commentary:** Add ball-by-ball commentary of matches.

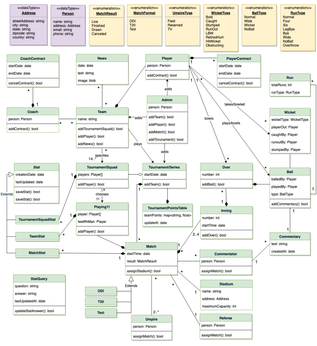


Use case diagram

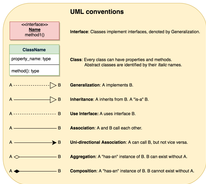
Class diagram

Here are the main classes of the Cricinfo system:

- **Player:** Keeps a record of a cricket player, their basic profile and contracts.
- **Team:** This class manages cricket teams.
- **Tournament:** Manages cricket tournaments and keeps track of the points table for all playing teams.
- **TournamentSquad:** Each team playing a tournament will announce a set of players who will be playing the tournament. TournamentSquad will encapsulate that.
- **Playing11:** Each team playing a match will select 11 players from their announced tournaments squad.
- **Match:** Encapsulates all information of a cricket match. Our system will support three match types: 1) ODI, 2) T20, and 3) Test
- **Innings:** Records all innings of a match.
- **Over:** Records details about an Over.
- **Ball:** Records every detail of a ball, such as the number of runs scored, if it was a wicket-taking ball, etc.
- **Run:** Records the number and type of runs scored on a ball. The different run types are: Wide, LegBy, Four, Six, etc.
- **Commentator and Commentary:** The commentator adds ball-by-ball commentary.
- **Umpire and Referee:** These classes will store details about umpires and referees, respectively.
- **Stat:** Our system will keep track of the stats for every player, match and tournament.
- **StatQuery:** This class will encapsulate general stat queries and their answers, like "Who has scored the maximum number of 100s in ODIs?" or, "Which bowler has taken the most wickets in test matches?", etc.

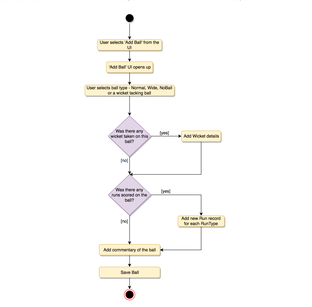


Class diagram



Activity diagrams

Record a Ball of an Over: Here are the steps to record a ball of an over in the system:



Code

Here is the high-level definition for the classes described above.

Enums, data types, and constants: Here are the required enums, data types, and constants:

Java | Python

```java
public class Address {
  private String streetAddress;
  private String city;
  private String state;
  private String zipCode;
  private String country;
}

public class Person {
  private String name;
  private Address address;
  private String email;
  private String phone;
}

public enum MatchFormat {
  ODI,
  T20,
  TEST
}

public enum MatchResult {
  LIVE,
  FINISHED,
  DRAWN,
  CANCELED
}

```

Admin, Player, Umpire, Referee, and Commentator: These classes represent the different people that interact with our system:

Java | Python

```java
// 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 method and modified only through their public setter method.

public class Player {
  private Person person;
  private ArrayList<PlayerContract> contracts;

  public boolean addContract();
}

public class Admin {
  private Person person;

  public boolean addMatch(Match match);

  public boolean addTeam(Team team);

  public boolean addTournament(Tournament tournament);
}

public class Umpire {
  private Person person;

  public boolean assignMatch(Match match);
}

public class Referee {
```

Team, TournamentSquad, and Playing11: Team will announce a squad for a tournament, out of which, the playing 11 will be chosen:

Java | Python

```java
public class Team {
  private String name;
  private List<Player> players;
  private List<News> news;
  private Coach coach;

  public boolean addTournamentSquad(TournamentSquad tournamentSquad);
  public boolean addPlayer(Player player);
  public boolean addNews(News news);
}

public class TournamentSquad {
  private List<Player> players;
  private List<TournamentStat> tournamentStats;

  public boolean addPlayer(Player player);
}

public class Playing11 {
  private List<Player> players;
  private Player twelfthMan;

  public boolean addPlayer(Player player);
}

```

Over, Ball, Wicket, Commentary, Inning, and Match: Match will be an abstract class, extended by ODI, Test, and T20:

Java | Python

```java
public class Over {
  private int number;
  private List<Ball> balls;

  public boolean addBall(Ball ball);
}

public class Ball {
  private Player balledBy;
  private Player playedBy;
  private BallType type;

  private Wicket wicket;
  private List<Run> runs;
  private Commentary commentary;

}

public class Wicket {
  private WicketType wicketType;
  private Player playerOut;
  private Player caughtBy;
  private Player runoutBy;
  private Player stumpedBy;
}

public class Commentary {
  private String text;
```