Design Cricinfo

Let's design Cricinfo.

We'll cover the following

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

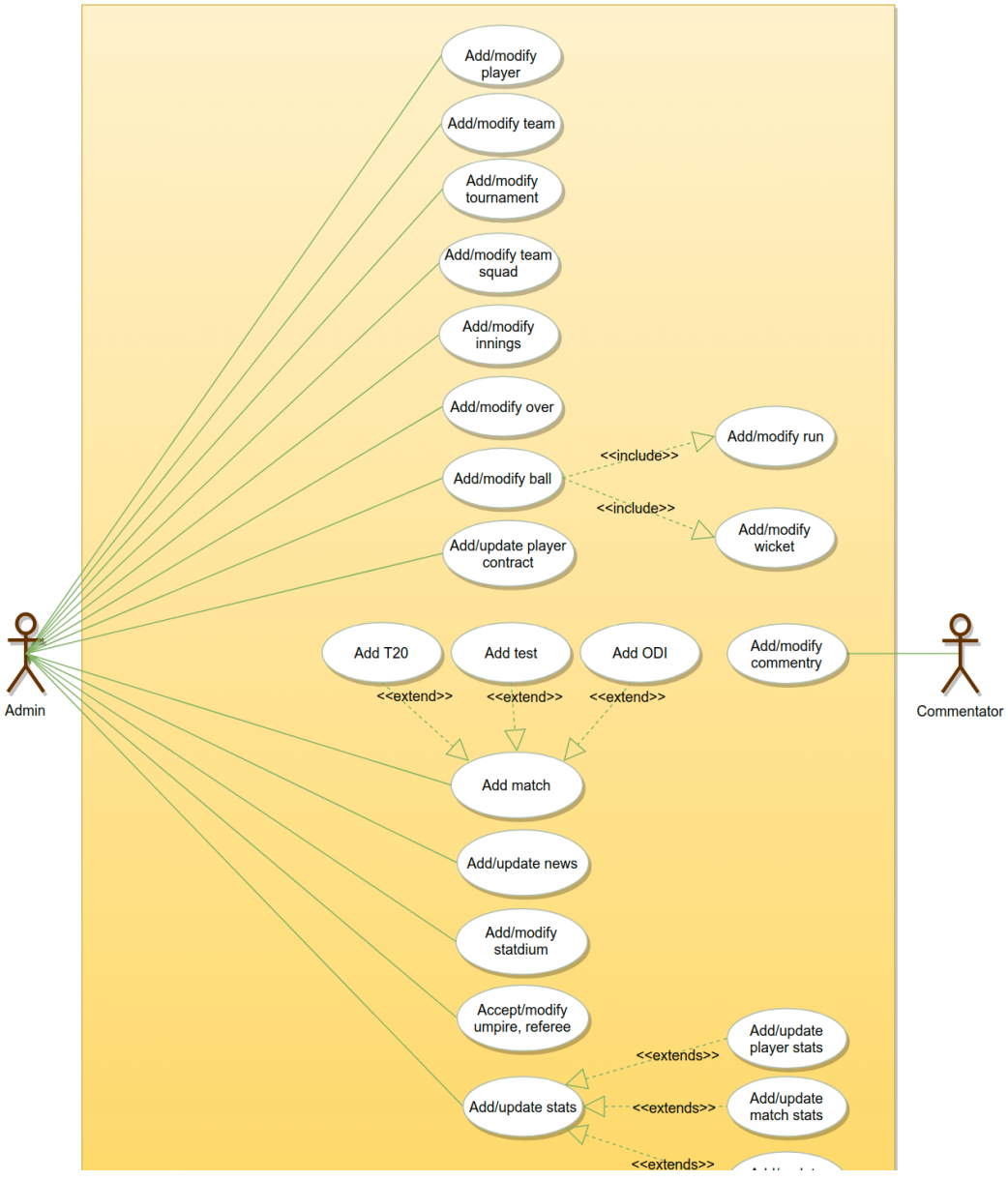
Use case diagram

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.



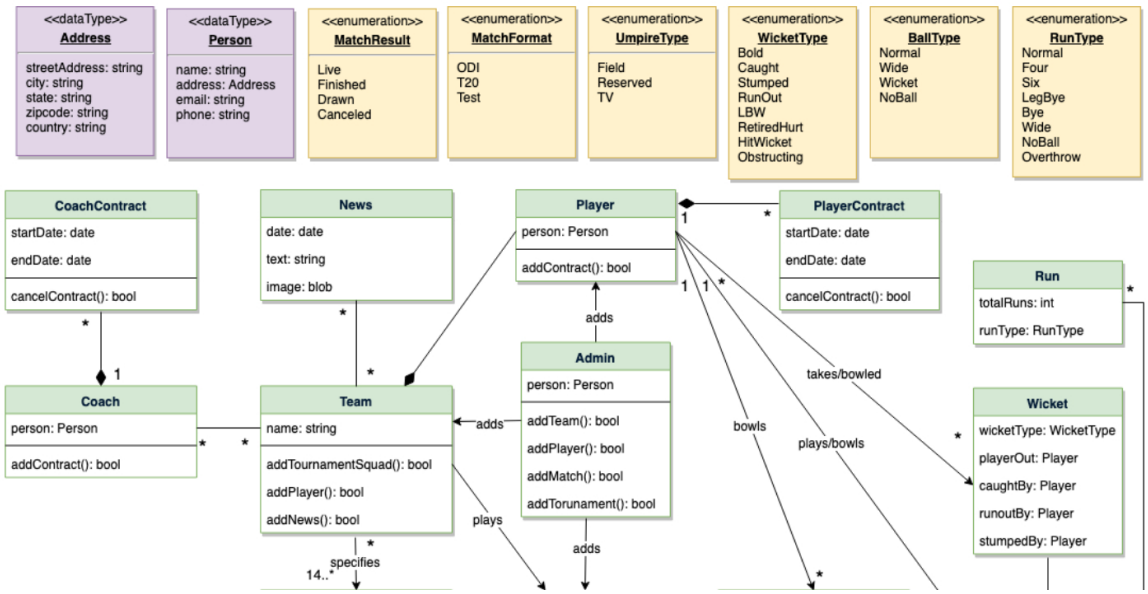
Use case diagram

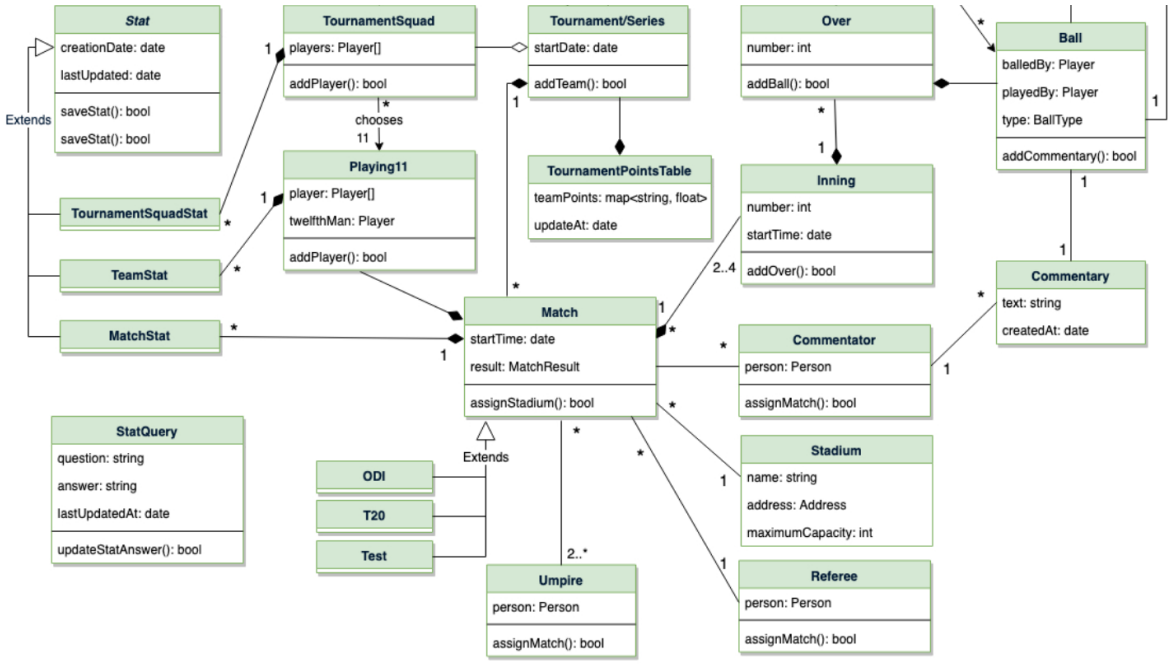
Class diagram

#

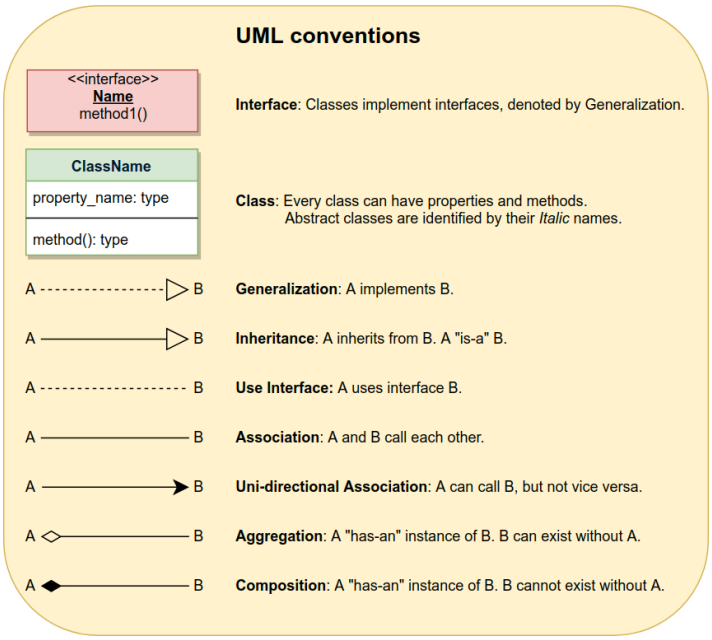
Here are the main classes of the Cricinfo system:

- **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.





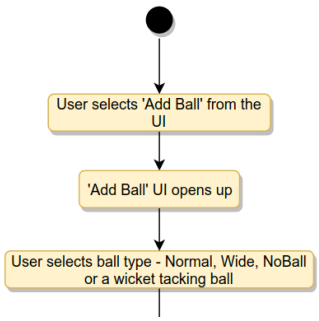
Class diagram

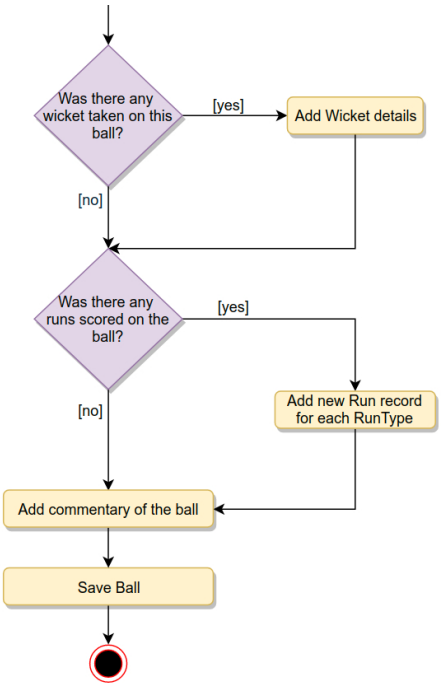


Activity diagrams

#

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

```python
class Address:
  def __init__(self, street, city, state, zip_code, country):
    self.__street_address = street
    self.__city = city
    self.__state = state
    self.__zip_code = zip_code
    self.__country = country


class Person():
  def __init__(self, name, address, email, phone):
    self.__name = name
    self.__address = address
    self.__email = email
    self.__phone = phone


class MatchFormat(Enum):
  ODI, T20, TEST = 1, 2, 3


class MatchResult(Enum):
  LIVE, FINISHED, DRAWN, CANCELLED = 1, 2, 3, 4


class UmpireType(Enum):
  FIELD, RESERVED, TV = 1, 2, 3


class WicketType(Enum):
  BOLD, CAUGHT, STUMPED, RUN OUT, LBW, RETIRED HURT, HIT WICKET, OBSTRUCTING = 1, 2, 3, 4, 5, 6, 7, 8
```

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

Java Python

```python
# 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 methods function.

class Player:
  def __init__(self, person):
    self.__person = person
    self.__contracts = []

  def add_contract(self, contract):
    None


class Admin:
  def __init__(self, person):
    self.__person = person

  def add_match(self, match):
    None

  def add_team(self, team):
    None

  def add_tournament(self, tournament):
    None


class Umpire:
  def __init__(self, person):
    self.__person = person
```

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

Java Python

```python
class Team:
  def __init__(self, name, coach):
    self.__name = name
    self.__players = []
    self.__news = []
    self.__coach = coach

  def add_tournament_squad(self, tournament_squad):
    None

  def add_player(self, player):
    None

  def add_news(self, news):
    None


class TournamentSquad:
  def __init__(self):
    self.__players = []
    self.__tournament_stats = []

  def add_player(self, player):
    None


class Playing11:
  def __init__(self):
    self.__players = []
    self.__twelfth_man = None
```

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

Java Python

```python
class Over:
  def __init__(self, number):
    self.__number = number
    self.__balls = []

  def add_ball(self, ball):
```

```
    None


class Ball:
  def __init__(self, balled_by, played_by, ball_type, wicket, runs, commentary):
    self.__balled_by = balled_by
    self.__played_by = played_by
    self.__type = ball_type

    self.__wicket = wicket
    self.__runs = runs
    self.__commentary = commentary


class Wicket:
  def __init__(self, wicket_type, player_out, caught_by, runout_by, stumped_by):
    self.__wicket_type = wicket_type
    self.__player_out = player_out
    self.__caught_by = caught_by
    self.__runout_by = runout_by
    self.__stumped_by = stumped_by


class Commentary:
  def __init__(self, text, commentator):
```


Back

Design LinkedIn

Next

Design Facebook - a social network

Mark as Completed