```
import java.util.List;
import java.util.ArrayList;
import java.util.Date;

/** Representation for Major League Baseball
 */
public class MLB
{
    private League [] league;
    private List<Season> pastSeasons;
    private Season currentSeason;

    /** Constructor
     */
    public MLB()
    { league = new League[2];
      league[0] = new League(this, "American");
      league[1] = new League(this, "National");
      pastSeasons = new ArrayList<Season>();
    }

    /** Start a new season
     *  @param year the year for the season
     */
    public void startSeason(int year)
    { if (currentSeason != null)
        pastSeasons.add(currentSeason);
      currentSeason = new Season(year);
    }
}
```

```
/** Representation for one of the leagues of Major
 *  League Baseball
 */
class League
{
    private MLB mlb;
    private String name;
    private Division [] division;

    /** Constructor
     *  @param mlb the object representing Major League Baseball
     *  @param name the name of this league
     */
    League(MLB mlb, String name)
    { this.mlb = mlb;
      this.name = name;
      division = new Division[3];
      division[0] = new Division(this, "Eastern");
      division[1] = new Division(this, "Central");
      division[2] = new Division(this, "Western");
    }

    /** Accessor for name
     *  @return the name of this league
     */
    public String getName()
    { return name; }
}
```

```
/** Representation for one of the divisions of an MLB league
 */
class Division
{
    private League league;
    private String name;
    private List<Team> teams;

    /** Constructor
     *  @param league the league to which this division belongs
     *  @param name the name of this division
     */
    Division(League league, String name)
    { this.league = league;
      this.name = league.getName() + " " + name;
      teams = new ArrayList<Team>();
    }

    /** Accessor for name
     *  @return the name of this division
     */
    public String getName()
    { return name; }

    /** Add a team to this division
     *  @param team the team to add
     */
    public void addTeam(Team team)
    { teams.add(team); }

    /** Remove a team from this division
     *  @param team the team to remove
     */
    public void removeTeam(Team team)
    { teams.remove(team); }
}
```

```
/** Representation for a Major League Baseball team
 */
class Team
{
    private Division division;
    private String name;
    private List<Player> players;
    private List<Game> homeGames;
    private List<Game> awayGames;

    /** Constructor
     *  @param division the division to which this team belongs
     *  @param name the name of this team
     */
    Team(Division division, String name)
    { this.division = division;
      this.name = name;
      players = new ArrayList<Player>();
    }

    /** Accessor for name
     *  @return the name of this division
     */
    public String getName()
    { return name; }

    /** Move this team to a different division
     *  @param newDivision the division to move to
     */
    public void changeDivision(Division newDivision)
    { division.removeTeam(this);
      newDivision.addTeam(this);
      division = newDivision;
    }
```

```
    /** Add a player to this team
     *  @param player the player to add
     */
    public void addPlayer(Player player)
    { players.add(player); }

    /** Remove a player from this team
     *  @param player the player to remove
     */
    public void removePlayer(Player player)
    { players.remove(player); }

    /** Add a home game to the list of home games
     *  @param game the game to add
     */
    public void addHomeGame(Game game)
    { homeGames.add(game); }

    /** Add an away game to the list of away games
     *  @param game the game to add
     */
    public void addAwayGame(Game game)
    { awayGames.add(game); }
}
```

```
/** Representation for a Major League Baseball player
 */
class Player
{
    private Team team;
    private String name;

    /** Constructor
     *  @param team the (initial) team for the player -
     *   null if not yet assigned
     *  @param name the player's name
     */
    public Player(Team team, String name)
    { this.team = team;
      this.name = name;
    }

    /** Move this player to a different team
     *  @param newTeam the new Team.
     *   If null the player is not on any team
     */
    public void changeTeam(Team newTeam)
    { if (team != null)
        team.removePlayer(this);
      if (newTeam != null)
        newTeam.addPlayer(this);
      team = newTeam;
    }
}
```

```
/** Representation for a Major League Baseball Season
 */
class Season
{
    private int year;
    private List<Game> games;

    /** Constructor
     *  @param year the year for this season
     */
    public Season(int year)
    { this.year = year;
      games = new ArrayList<Game>();
    }

    /** Accessor for the year of this season
     *  @return the year
     */
    public int getYear()
    { return year; }

    /** Add a game to the list of games
     *  @param game the game to add
     */
    public void addGame(Game game)
    { games.add(game); }
}
```

```
/** Representation for a single game
 */
class Game
{
    private Season season;
    private Date date;
    private Team home;
    private Team visitor;
    private Umpire [] umpire;
    private int homeScore;
    private int visitorScore;

    /** Constructor
     *  @param season the season for this game
     *  @param date the date of this game
     *  @param home the home team for this game
     *  @param visitor the visitor team for this game
     */
    public Game(Season season,
                Date date,
                Team home,
                Team visitor)
    { this.season = season;
      this.date = date;
      this.home = home;
      this.visitor = visitor;
      home.addHomeGame(this);
      visitor.addAwayGame(this);
      // Scores are left at default initial value of 0
      // until the game is played
      // Umpires are recorded when umpires are assigned
    }
```

```
    /** Accessor for the year of this game
     *  @return the year
     */
    public int getYear()
    { return season.getYear(); }

    /** Accessor for the date of this game
     *  @return the date
     */
    public Date getDate()
    { return date; }

    /** Assign the umpires for this game
     *  @param plateUmp the home plate umpire
     *  @param firstBaseUmp the first base umpire
     *  @param secondBaseUmp the second base umpire
     *  @param thirdBaseUmp the third base umpire
     */
    public void assignUmpires(Umpire plateUmp,
                              Umpire firstBaseUmp,
                              Umpire secondBaseUmp,
                              Umpire thirdBaseUmp)
    { umpire = new Umpire[4];
      umpire[0] = plateUmp;
      umpire[1] = firstBaseUmp;
      umpire[2] = secondBaseUmp;
      umpire[3] = thirdBaseUmp;
      for (int i = 0; i < umpire.length; i ++)
        umpire[i].addGame(this);
    }
}
```

```
/** Representation for an umpire
 */
class Umpire
{
    private String name;
    private List<Game> gamesAssignedTo;

    /** Constructor
     *  @param name the umpire's name
     */
    public Umpire(String name)
    { this.name = name;
      this.gamesAssignedTo = new ArrayList<Game>();
    }

    /** Add a game to the list of games assigned to
     *  @param game the game
     */
    public void addGame(Game game)
    { gamesAssignedTo.add(game); }
}
```