SECTION 8 PRACTICE

```
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;

class Team {
    private String name;
    private int wins;
    private int losses;
    private int ties;
    private int totalGoalsScored;
    private int totalGoalsAllowed;

    public Team(String name) {
        this.name = name;
        this.wins = 0;
        this.losses = 0;
        this.ties = 0;
        this.totalGoalsScored = 0;
        this.totalGoalsAllowed = 0;
    }
```

```
public String getName() {
   return name;
}

public int getWins() {
   return wins;
}

public int getLosses() {
   return losses;
}

public int getTies() {
   return ties;
}

public int getTotalGoalsScored() {
   return totalGoalsScored;
}

public int getTotalGoalsAllowed() {
   return totalGoalsAllowed;
}

public void addWin() {
```

```
        wins++;
    }

    public void addLoss() {
        losses++;
    }

    public void addTie() {
        ties++;
    }

    public void addGoalsScored(int goals) {
        totalGoalsScored += goals;
    }

    public void addGoalsAllowed(int goals) {
        totalGoalsAllowed += goals;
    }

    public void printStats() {
        System.out.println(name);
        System.out.println("Wins: " + wins + ", Losses: " + losses + ", Ties: " + ties);
        System.out.println("Points Scored: " + totalGoalsScored + ", Points Allowed: " +
totalGoalsAllowed);
    }
}
```

```
class Game {
    private static int gameCounter = 0;
    private int gameId;
    private Team awayTeam;
    private Team homeTeam;
    private int awayScore;
    private int homeScore;
    private int temperature;

    public Game(Team awayTeam, Team homeTeam, int temperature) {
        this.gameId = ++gameCounter;
        this.awayTeam = awayTeam;
        this.homeTeam = homeTeam;
        this.temperature = temperature;
        this.awayScore = generateScore(temperature);
        this.homeScore = generateScore(temperature);
        updateTeamStats();
    }

    private int generateScore(int temperature) {
        Random rand = new Random();
        return rand.nextInt(temperature / 10 + 1); // Score range increases with
temperature
    }
```

```
private void updateTeamStats() {
    awayTeam.addGoalsScored(awayScore);
    awayTeam.addGoalsAllowed(homeScore);
    homeTeam.addGoalsScored(homeScore);
    homeTeam.addGoalsAllowed(awayScore);

    if (awayScore > homeScore) {
        awayTeam.addWin();
        homeTeam.addLoss();
    } else if (awayScore < homeScore) {
        homeTeam.addWin();
        awayTeam.addLoss();
    } else {
        awayTeam.addTie();
        homeTeam.addTie();
    }
}

public int getTemperature() {
    return temperature;
}

public void printGameStats() {
    System.out.println("Game #" + gameId);
    System.out.println("Temperature: " + temperature);
    System.out.println("Away Team: " + awayTeam.getName() + ", " + awayScore);
```

```
        System.out.println("Home Team: " + homeTeam.getName() + ", " + homeScore);
    }
}

class Scheduler {
    private Team[] teams;
    private ArrayList<Game> games;
    private int consecutiveFreezingWeeks;
    private int hottestTemperature;
    private int totalTemperature;
    private int numberOfGames;

    public Scheduler(Team[] teams) {
        this.teams = teams;
        this.games = new ArrayList<>();
        this.consecutiveFreezingWeeks = 0;
        this.hottestTemperature = Integer.MIN_VALUE;
        this.totalTemperature = 0;
        this.numberOfGames = 0;
    }

    public void startSeason() {
        Scanner scanner = new Scanner(System.in);
        while (consecutiveFreezingWeeks < 3) {
            System.out.print("Enter this week's temperature: ");
            int temperature = getInputTemperature(scanner);
```

```
        if (temperature <= 32) {
            consecutiveFreezingWeeks++;
            System.out.println("Too cold to play.");
            if (consecutiveFreezingWeeks == 3) {
                System.out.println("Season is over due to 3 consecutive freezing weeks.");
                break;
            }
        } else {
            consecutiveFreezingWeeks = 0;
            scheduleGames(temperature);
        }
    }
    scanner.close();
    printSeasonSummary();
}

private int getInputTemperature(Scanner scanner) {
    while (true) {
        try {
            return Integer.parseInt(scanner.nextLine());
        } catch (NumberFormatException e) {
            System.out.print("Invalid input. Enter a valid temperature: ");
        }
    }
}
```

```
private void scheduleGames(int temperature) {
    numberOfGames += 2;
    totalTemperature += 2 * temperature;
    if (temperature > hottestTemperature) {
        hottestTemperature = temperature;
    }

    Random rand = new Random();
    ArrayList<Integer> teamIndexes = new ArrayList<>();
    for (int i = 0; i < teams.length; i++) {
        teamIndexes.add(i);
    }

    for (int i = 0; i < 2; i++) {
        int awayIndex = teamIndexes.remove(rand.nextInt(teamIndexes.size()));
        int homeIndex = teamIndexes.remove(rand.nextInt(teamIndexes.size()));
        games.add(new Game(teams[awayIndex], teams[homeIndex], temperature));
    }
}

private void printSeasonSummary() {
    System.out.println("*********RESULTS*********");
    for (Team team : teams) {
        team.printStats();
    }
    for (Game game : games) {
```

```
            game.printGameStats();
        }
        System.out.println("Hottest Temp: " + hottestTemperature);
        System.out.println("Average Temp: " + (totalTemperature / numberOfGames));
    }
}

public class SoccerLeagueSimulation {
    public static void main(String[] args) {
        Team[] teams = {
            new Team("Team 1"),
            new Team("Team 2"),
            new Team("Team 3"),
            new Team("Team 4")
        };
        Scheduler scheduler = new Scheduler(teams);
        scheduler.startSeason();
    }
}
```