**ASSIGNMENT-7**

* B. Rishitha

192324130

**Java Foundations**

*Practices - Section 8:*

*The Soccer League*

1. **Temperature Handling:**
   * Games are not played if the temperature is freezing (32°F or below).
   * The output mentions three consecutive "Too cold to play." messages, followed by "Season is over."
2. **Game and Season Statistics:**
   * Statistics for each game, including temperature and scores, are printed.
   * The hottest temperature and the average temperature are also calculated.

**Areas to Check in the Code:**

1. **Temperature Input and Freezing Check:**
   * Ensure that the program correctly identifies and handles freezing temperatures.
   * The season should end after three consecutive weeks of freezing temperatures.
2. **Game Results and Statistics:**
   * Ensure that the results of each game, including the temperature, are logged correctly.
   * Check that the statistics are cumulative and consistent with the description.

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 goalsScored;

private int goalsAllowed;

public Team(String name) {

this.name = name;

this.wins = 0;

this.losses = 0;

this.ties = 0;

this.goalsScored = 0;

this.goalsAllowed = 0;

}

public String getName() {

return name;

}

public void recordGame(int goalsFor, int goalsAgainst) {

goalsScored += goalsFor;

goalsAllowed += goalsAgainst;

if (goalsFor > goalsAgainst) {

wins++;

} else if (goalsFor < goalsAgainst) {

losses++;

} else {

ties++;

}

}

public void printStats() {

System.out.println("Team " + name);

System.out.println("Wins: " + wins + ", Losses: " + losses + ", Ties: " + ties);

System.out.println("Goals Scored: " + goalsScored + ", Goals Allowed: " + goalsAllowed);

}

}

class Game {

private static int gameCounter = 0;

private int gameId;

private Team homeTeam;

private Team awayTeam;

private int homeScore;

private int awayScore;

private int temperature;

public Game(Team homeTeam, Team awayTeam, int temperature) {

this.gameId = ++gameCounter;

this.homeTeam = homeTeam;

this.awayTeam = awayTeam;

this.temperature = temperature;

playGame();

}

private void playGame() {

Random rand = new Random();

int maxGoals = Math.max(1, temperature / 10); // Goals proportional to temperature

homeScore = rand.nextInt(maxGoals + 1);

awayScore = rand.nextInt(maxGoals + 1);

homeTeam.recordGame(homeScore, awayScore);

awayTeam.recordGame(awayScore, homeScore);

}

public int getTemperature() {

return temperature;

}

public void printGameResult() {

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 League {

private ArrayList<Team> teams;

private ArrayList<Game> games;

private int hottestTemperature;

private int totalTemperature;

private int temperatureReadings;

public League(String[] teamNames) {

teams = new ArrayList<>();

games = new ArrayList<>();

hottestTemperature = Integer.MIN\_VALUE;

totalTemperature = 0;

temperatureReadings = 0;

for (String name : teamNames) {

teams.add(new Team(name));

}

}

public ArrayList<Team> getTeams() {

return teams;

}

public void addGame(Game game) {

games.add(game);

int temp = game.getTemperature();

hottestTemperature = Math.max(hottestTemperature, temp);

totalTemperature += temp;

temperatureReadings++;

}

public void printSeasonResults() {

System.out.println("\*RESULTS\*");

for (Team team : teams) {

team.printStats();

}

for (Game game : games) {

game.printGameResult();

}

System.out.println("Hottest Temp: " + hottestTemperature);

System.out.println("Average Temp: " + (temperatureReadings == 0 ? 0 : (totalTemperature / temperatureReadings)));

}

}

public class Scheduler {

private League league;

private int freezingWeeks;

public Scheduler(String[] teamNames) {

league = new League(teamNames);

freezingWeeks = 0;

}

public void startSeason() {

Scanner scanner = new Scanner(System.in);

Random rand = new Random();

while (true) {

System.out.print("Enter temperature: ");

int temperature;

try {

temperature = Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter a valid temperature.");

continue;

}

if (temperature <= 32) {

freezingWeeks++;

System.out.println("Too cold to play.");

if (freezingWeeks >= 3) {

System.out.println("Season is over");

break;

}

} else {

freezingWeeks = 0;

playGames(temperature, rand);

}

}

league.printSeasonResults();

}

private void playGames(int temperature, Random rand) {

ArrayList<Team> teams = league.getTeams();

int numGames = 2; // 2 games every Tuesday

while (numGames > 0) {

int team1Index = rand.nextInt(teams.size());

int team2Index;

do {

team2Index = rand.nextInt(teams.size());

} while (team2Index == team1Index);

Team team1 = teams.get(team1Index);

Team team2 = teams.get(team2Index);

Game game = new Game(team1, team2, temperature);

league.addGame(game);

numGames--;

}

}

public static void main(String[] args) {

String[] teamNames = {"Team 1", "Team 2", "Team 3", "Team 4"};

Scheduler scheduler = new Scheduler(teamNames);

scheduler.startSeason();

}

}





