ASSIGNMENT-7

K.SRAVANTHI

192372127

SECTION_8 PRACTICE

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

public class SoccerLeague {

    // Team class
    public static 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() {
        this.wins++;
    }

    public void addLoss() {
        this.losses++;
    }

    public void addTie() {
        this.ties++;
    }

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

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

// Game class
```

```
public static class Game {
    private static int gameCount = 0;
    private int gameId;
    private Team awayTeam;
    private Team homeTeam;
    private int awayTeamScore;
    private int homeTeamScore;
    private int temperature;

    public Game(Team awayTeam, Team homeTeam, int temperature) {
        this.gameId = ++gameCount;
        this.awayTeam = awayTeam;
        this.homeTeam = homeTeam;
        this.temperature = temperature;
        this.awayTeamScore = new Random().nextInt(Math.max(1, temperature / 10));
        this.homeTeamScore = new Random().nextInt(Math.max(1, temperature / 10));

        awayTeam.addGoalsScored(awayTeamScore);
        awayTeam.addGoalsAllowed(homeTeamScore);
        homeTeam.addGoalsScored(homeTeamScore);
        homeTeam.addGoalsAllowed(awayTeamScore);

        if (awayTeamScore > homeTeamScore) {
            awayTeam.addWin();
            homeTeam.addLoss();
        } else if (homeTeamScore > awayTeamScore) {
            homeTeam.addWin();
```

```
            awayTeam.addLoss();
        } else {
            awayTeam.addTie();
            homeTeam.addTie();
        }
    }

    public void printGameResult() {
        System.out.println("Game #" + gameId);
        System.out.println("Temperature: " + temperature);
        System.out.println("Away Team: " + awayTeam.getName() + ", " + awayTeamScore);
        System.out.println("Home Team: " + homeTeam.getName() + ", " + homeTeamScore);
    }
}

// Scheduler class
public static class Scheduler {
    private Team[] teams;
    private ArrayList<Game> games;
    private ArrayList<Integer> temperatures;
    private int freezingWeeks;

    public Scheduler(Team[] teams) {
        this.teams = teams;
        this.games = new ArrayList<>();
        this.temperatures = new ArrayList<>();
        this.freezingWeeks = 0;
```

```
}

public void startSeason() {
    Scanner scanner = new Scanner(System.in);
    Random random = new Random();
    while (true) {
        System.out.print("Enter this week's temperature: ");
        int temperature = 0;
        try {
            temperature = Integer.parseInt(scanner.nextLine());
        } catch (NumberFormatException e) {
            System.out.println("Invalid input. Please enter a valid temperature.");
            continue;
        }

        temperatures.add(temperature);

        if (temperature <= 32) {
            freezingWeeks++;
            System.out.println("Too cold to play.");
            if (freezingWeeks >= 3) {
                System.out.println("Season is over");
                printSeasonResults();
                break;
            }
            continue;
        } else {
```

```
            freezingWeeks = 0;
        }

        ArrayList<Team> teamsList = new ArrayList<>();
        for (Team team : teams) {
            teamsList.add(team);
        }

        for (int i = 0; i < 2; i++) {
            Team team1 = teamsList.remove(random.nextInt(teamsList.size()));
            Team team2 = teamsList.remove(random.nextInt(teamsList.size()));
            Game game = new Game(team1, team2, temperature);
            games.add(game);
        }
      }
      scanner.close();
    }

    public void printSeasonResults() {
      System.out.println("*********RESULTS*********");
      for (Team team : teams) {
        System.out.println(team.getName());
        System.out.println("Wins: " + team.getWins() + ", Losses: " + team.getLosses() + ",
Ties: " + team.getTies());
        System.out.println("Goals Scored: " + team.getTotalGoalsScored() + ", Goals Allowed:
" + team.getTotalGoalsAllowed());
      }
```

```
            for (Game game : games) {
                game.printGameResult();
            }

            int hottestTemp = temperatures.stream().mapToInt(v -> v).max().orElse(0);
            double averageTemp = temperatures.stream().mapToInt(v -> v).average().orElse(0.0);
            System.out.println("Hottest Temp: " + hottestTemp);
            System.out.println("Average Temp: " + averageTemp);
        }
    }

    // Main method
    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();
    }
}
```

OUTPUT:

```
java -cp /tmp/ZyDPtY86Oy/SoccerLeague
Enter this week's temperature: 45
Enter this week's temperature: -10
Too cold to play.
Enter this week's temperature: 2
Too cold to play.
Enter this week's temperature: 1
Too cold to play.
Season is over
*********RESULTS*********
Team 1
Wins: 0, Losses: 1, Ties: 0
Goals Scored: 1, Goals Allowed: 3
Team 2
Wins: 1, Losses: 0, Ties: 0
Goals Scored: 1, Goals Allowed: 0
Team 3
Wins: 0, Losses: 1, Ties: 0
Goals Scored: 0, Goals Allowed: 1
Team 4
Wins: 1, Losses: 0, Ties: 0
Goals Scored: 3, Goals Allowed: 1
Game #1
Temperature: 45
Away Team: Team 2, 1
Home Team: Team 3, 0
Game #2
Temperature: 45
Away Team: Team 4, 3
Home Team: Team 1, 1
Hottest Temp: 45
Average Temp: 9.5

=== Code Execution Successful ===
```

SECTION-7

import java.util.Random;

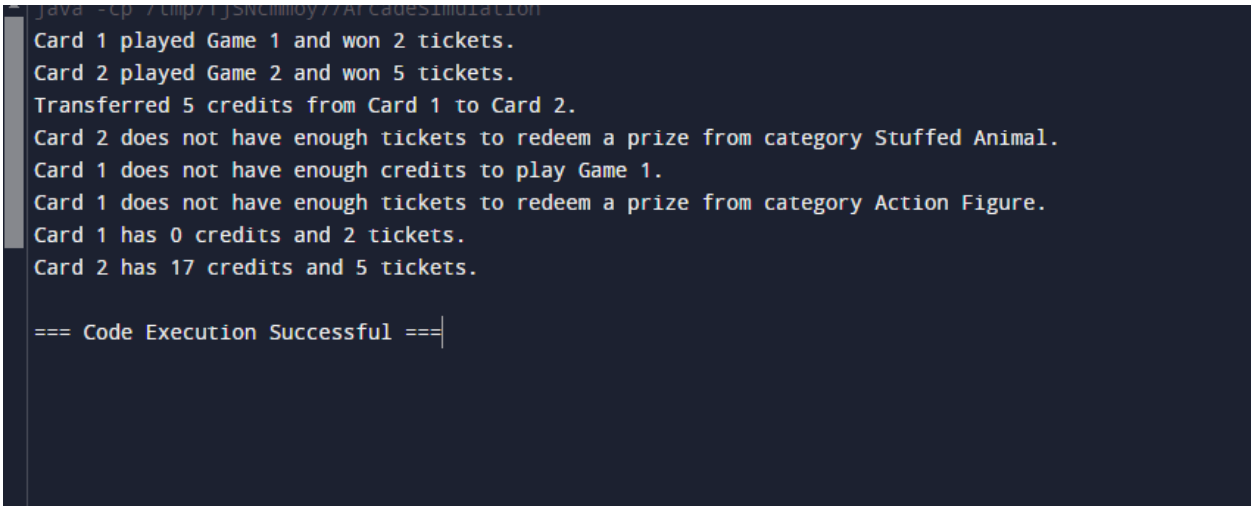
```
class ArcadeCard {
    private int cardNumber;
    private int creditBalance;
    private int ticketBalance;

    public ArcadeCard(int cardNumber) {
        this.cardNumber = cardNumber;
        this.creditBalance = 0;
        this.ticketBalance = 0;
    }

    public int getCardNumber() {
        return cardNumber;
    }

    public int getCreditBalance() {
        return creditBalance;
    }

    public int getTicketBalance() {
        return ticketBalance;
    }

    public void addCredits(int credits) {
        creditBalance += credits;
    }
```

```
    public void subtractCredits(int credits) {
        if (creditBalance >= credits) {
            creditBalance -= credits;
        } else {
            System.out.println("Insufficient credits.");
        }
    }

    public void addTickets(int tickets) {
        ticketBalance += tickets;
    }

    public void subtractTickets(int tickets) {
        if (ticketBalance >= tickets) {
            ticketBalance -= tickets;
        } else {
            System.out.println("Insufficient tickets.");
        }
    }
}

// Game class
class Game {
    private String name;
    private int creditsRequired;
    private int ticketBalance;
```

```
    public Game(String name, int creditsRequired) {
        this.name = name;
        this.creditsRequired = creditsRequired;
        this.ticketBalance = 0;
    }

    public String getName() {
        return name;
    }

    public int getCreditsRequired() {
        return creditsRequired;
    }

    public int getTicketBalance() {
        return ticketBalance;
    }

    public void play(ArcadeCard card) {
        if (card.getCreditBalance() >= creditsRequired) {
            card.subtractCredits(creditsRequired);
            Random random = new Random();
            int ticketsWon = random.nextInt(10);
            card.addTickets(ticketsWon);
            ticketBalance += ticketsWon;
            System.out.println("Card " + card.getCardNumber() + " played " + name + " and won " +
ticketsWon + " tickets.");
```

```
    } else {
        System.out.println("Card " + card.getCardNumber() + " does not have enough credits to play " + name + ".");
    }
  }
}
```

OUTPUT:



SECTION-9

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

class Dorm {
  private String name;
  private int population;
  private double x, y;

  public Dorm(String name, double x, double y, int population) {
```

```
    this.name = name;
    this.population = population;
    this.x = x;
    this.y = y;
}

public double getX() {
    return x;
}

public double getY() {
    return y;
}

public int getPopulation() {
    return population;
}

public void setPopulation(int population) {
    this.population = population;
}

public void setLocation(double x, double y) {
    this.x = x;
    this.y = y;
}
```

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

class Student {
    private Dorm dorm;

    public Student(Dorm dorm) {
        this.dorm = dorm;
    }

    public double getX() {
        return dorm.getX();
    }

    public double getY() {
        return dorm.getY();
    }
}

public class CampusMap {
    private static ArrayList<Dorm> dorms = new ArrayList<>();
    private static ArrayList<Student> studyGroup = new ArrayList<>();

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
```

```
// Adding dorms
dorms.add(new Dorm("Dorm A", 100, 200, 100));
dorms.add(new Dorm("Dorm B", 500, 300, 150));
dorms.add(new Dorm("Dorm C", 300, 500, 200));

// Adding students to the study group
studyGroup.add(new Student(dorms.get(0)));
studyGroup.add(new Student(dorms.get(1)));
studyGroup.add(new Student(dorms.get(2)));

while (true) {
   System.out.println("Current Dorm Populations:");
   for (Dorm dorm : dorms) {
      System.out.println(dorm.getName() + ": " + dorm.getPopulation());
   }

   System.out.println("Enter dorm name to update population (or 'exit' to finish):");
   String dormName = scanner.nextLine();
   if (dormName.equals("exit")) break;

   System.out.println("Enter new population:");
   int newPopulation = Integer.parseInt(scanner.nextLine());

   for (Dorm dorm : dorms) {
      if (dorm.getName().equals(dormName)) {
         dorm.setPopulation(newPopulation);
```

```
            }
        }

        updateCenters();
    }

    scanner.close();
}

private static void updateCenters() {
    double allX = 0, allY = 0, totalPopulation = 0;
    for (Dorm dorm : dorms) {
        allX += dorm.getX() * dorm.getPopulation();
        allY += dorm.getY() * dorm.getPopulation();
        totalPopulation += dorm.getPopulation();
    }
    double centerX = allX / totalPopulation;
    double centerY = allY / totalPopulation;

    System.out.println(String.format("Center of All Students: (%.2f, %.2f)", centerX,
centerY));

    // Update the study group center
    double studyX = 0, studyY = 0;
    for (Student student : studyGroup) {
        studyX += student.getX();
        studyY += student.getY();
```

```
    }
    double studyCenterX = studyX / studyGroup.size();
    double studyCenterY = studyY / studyGroup.size();

    System.out.println(String.format("Center of Study Group: (%.2f, %.2f)", studyCenterX,
studyCenterY));
  }
}
```

OUTPUT: