ASSIGNMENT-7

- Sk.sumiya
192372090

Java Foundations

Practices - Section 9:
Finding the Central Location

Overview

Have you ever wondered where the most centrally located point on campus is? How about a centrally located point between you and a number of your friends? In this Problem Set, you'll write a JavaFX program that answers these questions visually.

Tasks

Review CampusMap.mp4 from Section 9, Lesson 1, Slide 6.

Your goal is to create the Campus Map program that uses your map of campus, dorm names, dorm populations, and your group of friends. You're welcome to design your own campus map (this is your background graphic). You'll have to design your own campus map if your actual campus has fewer than 3 dorms, otherwise this Problem Set wouldn't be too interesting.

Campus

UNIVERSITY

BAY

DORMI

Dorm

Study

The Dorms

Choose a way to visually represent dorms. The name and population of a dorm must also be visible. The population and location of each dorm must somehow be adjustable while the program is running.

STUDENT

CENTER

The Center Points

DORM 2

Dorm 2

SCIENCE

LIBRARY

GYM

279

ENGINEERING

DORM3

MEDICINE

PARKING

STREET-

DORM 6

DORMS

DORMY

SOUTH

Your program must show two center points. The first point represents the central location of all students in all dorms. This is essentially a center of mass problem where dorms with a larger population are considered more "massive" and have a greater influence over the center point's location.

The second point represents the central location of your study group. Create a study group of at least 3 people, 1 of which must live in a different dorm.

Both center points must include a visual representation, a label, and display their location as numeric values. These points should automatically update as a dorm's location or population changes. You're welcome to leave these measurements as pixels or convert them into real-life units of distance.

However you choose to represent your dorms and points, remember to perform your distance calculations based on the geometric center of these visuals, and not the top-left corners.

Hints:

There are certain concepts we didn't cover in Section 9, like how to work with a Text node. But we did discuss how to consult the JavaFX API Documentation. Part of the challenge of this Problem Set is understanding how to consult resources. If you have ideas about a feature you'd like to implement or a technique you'd like to explore, don't be afraid to consult the JavaFX API Documentation. It has a lot of fun things to show you.

```
public class CampusMapApp {

   static String[] dormNames = {"Dorm 1", "Dorm 2", "Dorm 3"};

   static int[] dormPopulations = {200, 150, 250};

   static double[][] dormLocations = {{100, 100}, {300, 150}, {500, 200}};


   public static void main(String[] args) {

      double[] overallCenter = calculateCenter(dormLocations, dormPopulations);

      double[] studyGroupCenter = calculateCenter(new double[][]{{100, 100}, {300, 150}, {500, 200}}, new int[]{1, 2, 3});


      System.out.printf("Overall Center Location: (%.2f, %.2f)%n", overallCenter[0], overallCenter[1]);

      System.out.printf("Study Group Center Location: (%.2f, %.2f)%n", studyGroupCenter[0], studyGroupCenter[1]);

   }


   public static double[] calculateCenter(double[][] locations, int[] populations) {

      double totalWeight = 0;

      double weightedX = 0;
```

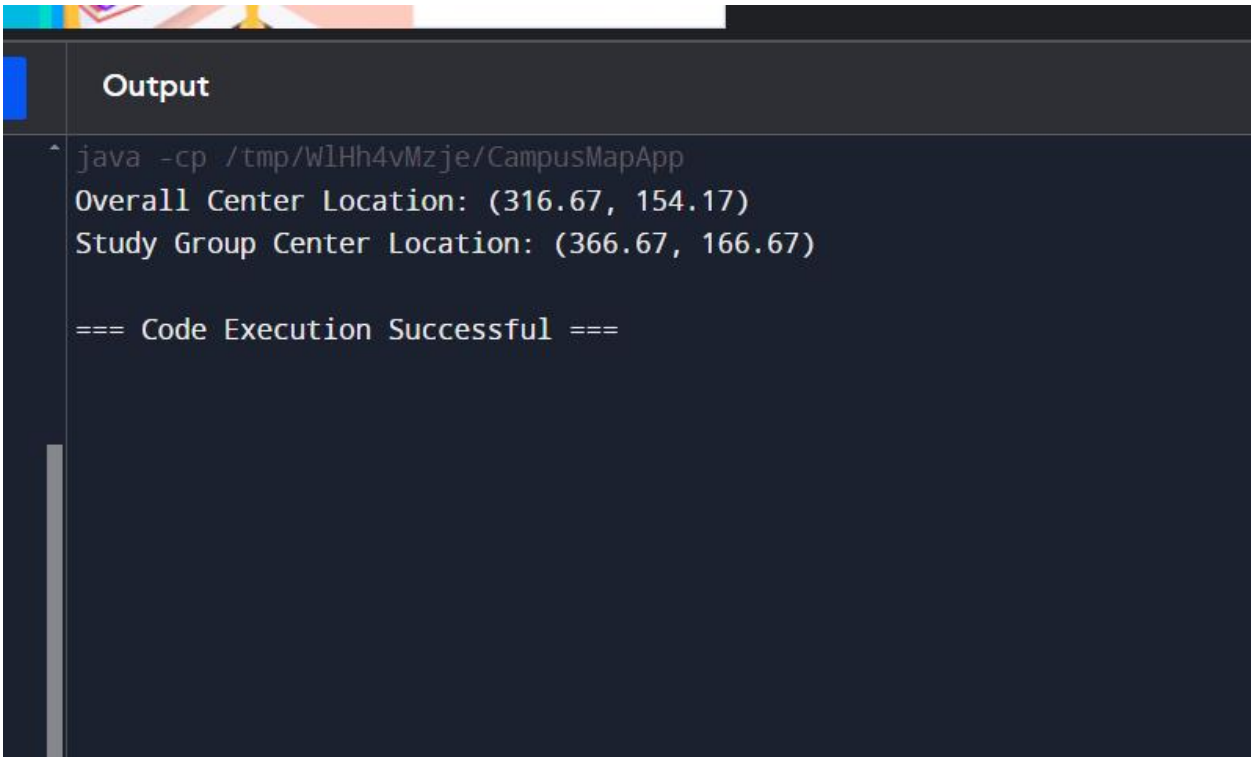
```
        double weightedY = 0;

        for (int i = 0; i < locations.length; i++) {
            double x = locations[i][0];
            double y = locations[i][1];
            int population = populations[i];

            weightedX += x * population;
            weightedY += y * population;
            totalWeight += population;
        }

        return new double[]{weightedX / totalWeight, weightedY / totalWeight};
    }
}
```

Output:


Output
java -cp /tmp/WlHh4vMzje/CampusMapApp
Overall Center Location: (316.67, 154.17)
Study Group Center Location: (366.67, 166.67)
=== Code Execution Successful ===