Project Title	Disaster Resource Allocation using Greedy Algorithm
Student Name	Meet Dadhaniya & Param Dholakia
Enrolment Numbers	12202040501038 & 12202040501049

Problem Description

The problem is to allocate limited resources to multiple districts in a disaster scenario, where each district has a different risk score based on its population, land type, and urbanization. The goal is to allocate resources in a way that maximizes the overall risk reduction. The program takes input from the user for the number of districts, total available resources, and details for each district, and then allocates resources to the districts based on their risk scores.

Source Code

```
import java.util.Arrays;
import java.util.Scanner;
class RiskStatistics
{
   public static int calculatePopulationRisk(int population) {
      if (population < 10000) {
         return 1;
      }
      else if (population >= 10000 && population <= 50000) {
         return 2;
      }
      else if(population>50000 && population<=100000) {
         return 3;
```

```
    }

    else{
        return 4;
    }
}

public static int calculateLandTypeRisk(String landType) {
    switch (landType) {
        case "Forest":
            return 1;
        case "Coastal":
            return 2;
        case "Desert":
            return 3;
        case "Urban":
            return 4;
        default:
            return 0;
    }
}
public static int calculateUrbanizationRisk(String urbanization) {
    switch (urbanization) {
        case "Rural":
            return 1;
        case "Suburban":
            return 2;
```

```
            case "Urban":
                return 3;
            default:
                return 0;
        }
    }
    public static int calculateTotalRisk(int populationRisk, int landTypeRisk, int
urbanizationRisk)
    {
        int weightPopulation = 3;
        int weightLandType = 2;
        int weightUrbanization = 1;
        int totalRisk=(populationRisk * weightPopulation) + (landTypeRisk * weightLandType)
+ (urbanizationRisk * weightUrbanization);
            return totalRisk;
    }
}
class District
{
    String name;
    int population;
    String landType;
    String urbanization;
    int riskScore;
    int resourceDemand;

    public District(String name, int population, String landType, String urbanization, int
resourceDemand)
```

```
    {
        this.name = name;
        this.population = population;
        this.landType = landType;
        this.urbanization = urbanization;
        this.resourceDemand = resourceDemand;
    }

    public String getName()
    {   return name;    }
    public int getPopulation()
    {   return population;    }
    public String getLandType()
    {   return landType;    }
    public String getUrbanization()
    {   return urbanization;    }
    public int getRiskScore()
    {   return riskScore;    }
    public void setRiskScore(int riskScore)
    {   this.riskScore = riskScore;    }
    public int getResourceDemand()
    {
        return resourceDemand;
    }
}
class DisasterResponse
{
```

```
    public static void main(String[] args)

    {

        Scanner sc = new Scanner(System.in);

        System.out.println("\n\n*----------MINI PROJECT----------*\n\n");

        System.out.println("SUBJECT NAME : DESIGN AND ANALYSIS OF
ALGORITHMS");

        System.out.println("SUBJECT CODE : 202045601");

        System.out.println("MINI PROJECT TITLE : DISASTER RESOURCE
ALLOCATION USING GREEDY ALGORITHM");

        System.out.println("\nDEVLOPED BY : MEET DADHANIYA <12202040501038> &
PARAM DHOLAKIA <12202040501049>");

        System.out.print("\n\n\nEnter the number of districts: ");

        int numDistricts = sc.nextInt();

        System.out.print("Enter the total available resources: ");

        int totalResources = sc.nextInt();

        District[] districts = new District[numDistricts];

        System.out.println("\n\n\t\t---------------------:Enter Details for each district:----------------
-----\n");

        for (int i = 0; i < numDistricts; i++) {

            System.out.println("\nEnter details for District " + (i + 1));

            System.out.print("Name: ");

            String name = sc.next();

            System.out.print("Population: ");

            int population = sc.nextInt();

            System.out.print("Available Land Types: ");

            System.out.print("1) Forest\t");

            System.out.print("2) Coastal\t");

            System.out.print("3) Desert\t");
```

```
        System.out.println("4) Urban\t");
    System.out.print("Enter choice: ");
        int ch1=sc.nextInt();
        String landType;
        switch (ch1) {
            case 1:
                landType = "Forest";
                break;
            case 2:
                landType = "Coastal";
                break;
            case 3:
                landType = "Desert";
                break;
            case 4:
                landType = "Urban";
                break;
            default:
                landType = "";
                break;
        }

        System.out.print("Available Urbanization types: ");
        System.out.print("1) Rural\t");
        System.out.print("2) Suburban\t");
        System.out.println("3) Urban\t");
    System.out.print("Enter choice: ");
```

```
        int ch2=sc.nextInt();
        String urbanization ;
        switch (ch2) {
            case 1:
                urbanization = "Rural";
                break;
            case 2:
                urbanization = "Suburban";
                break;
            case 3:
                urbanization = "Urban";
                break;
            default:
                urbanization = "";
                break;
        }

        System.out.print("Enter Resource demand for "+name+" : ");
        int resourceDemand = sc.nextInt();

        districts[i] = new District(name, population, landType, urbanization,
resourceDemand);
        districts[i].setRiskScore(RiskStatistics.calculateTotalRisk(
                RiskStatistics.calculatePopulationRisk(population),
                RiskStatistics.calculateLandTypeRisk(landType),
                RiskStatistics.calculateUrbanizationRisk(urbanization)));
        System.out.print("--------------------------------\n");
```

```
        }


         Arrays.sort(districts, (d1, d2) -> {
            double ratio1 = (double) d1.getRiskScore() / d1.getResourceDemand();
            double ratio2 = (double) d2.getRiskScore() / d2.getResourceDemand();
            return Double.compare(ratio2, ratio1); // Descending order
        });
        System.out.println("\n\n\t\t-------------------------------------------------------------------------
-\n");


        for(int m=0;m<districts.length;m++)
        {
            System.out.println("Risk Score of "+ districts[m].getName() +" : "+
districts[m].getRiskScore());
            }


        System.out.println("\n\nResource Allocation...");
        int remainingResources = totalResources;


        for (District district : districts)
    {
            if (remainingResources >= district.getResourceDemand())
        {
                System.out.println("Allocating " + district.getResourceDemand() + " resources to "
+ district.getName());
                remainingResources -= district.getResourceDemand();
            }
        else
```

```
        {
            System.out.println("Allocating " + remainingResources + " resources to " +
district.getName() + " (partial allocation)");
            remainingResources = 0;
            break;
        }
        }

        if (remainingResources > 0)
        {
           System.out.println("\nUnused resources remaining: " + remainingResources);
        }
        else
        {
           System.out.println("\nAll resources have been allocated.");
        }
        sc.close();
    }
}
```

Output

```
*----------MINI PROJECT----------*


SUBJECT NAME : DESIGN AND ANALYSIS OF ALGORITHMS
SUBJECT CODE : 202045601
MINI PROJECT TITLE : DISASTER RESOURCE ALLOCATION USING GREEDY ALGORITHM

DEVLOPED BY : MEET DADHANIYA <122020405010038> & PARAM DHOLAKIA <122020405010049>



Enter the number of districts: 3
Enter the total available resources: 20000


                --------------------:Enter Details for each district:----------------------


Enter details for District 1
Name: Morbi
Population: 100000
Available Land Types: 1) Forest 2) Coastal      3) Desert       4) Urban
Enter choice: 4
Available Urbanization types: 1) Rural  2) Suburban     3) Urban
Enter choice: 2
Enter Resource demand for Morbi : 7000
--------------------------------

Enter details for District 2
Name: Bhavnagar
Population: 45000
Available Land Types: 1) Forest 2) Coastal      3) Desert       4) Urban
Enter choice: 4
Available Urbanization types: 1) Rural  2) Suburban     3) Urban
Enter choice: 1
Enter Resource demand for Bhavnagar : 5000
--------------------------------

Enter details for District 3
Name: Anand
Population: 125000
Available Land Types: 1) Forest 2) Coastal      3) Desert       4) Urban
Enter choice: 4
Available Urbanization types: 1) Rural  2) Suburban     3) Urban
Enter choice: 3
Enter Resource demand for Anand : 10000
--------------------------------


                --------------------------------------------------------------------------------

Risk Score of Bhavnagar : 15
Risk Score of Morbi : 19
Risk Score of Anand : 23


Resource Allocation...
Allocating 5000 resources to Bhavnagar
Allocating 7000 resources to Morbi
Allocating 8000 resources to Anand (partial allocation)

All resources have been allocated.
```