|  |  |
| --- | --- |
| **Name** | Shubhan Singh |
| **UID no.** | 2022300118 |
| **Experiment No.** | 4-B |

|  |  |
| --- | --- |
| **PROBLEM STATEMENT :** | Write a program to make a 2-D array of Temperatures in different cities, and find the minimum and maximum temperatures across all the cities in the given time frame |
| **THEORY:** | 2-D arrays in Java:  In Java, a 2D array is an array of arrays, where each element of the array is another array. It is also known as a matrix or a table. Here are some important things to keep in mind about 2D arrays in Java:   1. A 2D array is declared by specifying two sets of square brackets after the array type. For example, "int[][] myArray = new int[3][4];" creates a 2D array of integers with 3 rows and 4 columns. 2. The elements of a 2D array are accessed using two indices, one for the row and one for the column. For example, "myArray[0][1]" refers to the element in the first row and second column of the array. 3. The number of rows and columns in a 2D array can be different, but each row must have the same number of columns. 4. 2D arrays can be initialized with values when they are declared. For example, "int[][] myArray = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};" creates a 2D array with 3 rows and 3 columns and initializes the values of the array. 5. 2D arrays can be used to represent matrices, tables, grids, and other structures that have rows and columns. 6. 2D arrays can be used with loops to iterate through all the elements of the array. For example, a nested loop can be used to iterate through each row and column of the array.   2D arrays in Java provide a way to store and manipulate data in a tabular format. They are useful for representing matrices, tables, and other structures that have rows and columns. They can be initialized with values and accessed using two indices. They can also be used with loops to iterate through all the elements of the array.  Clearing input buffer left by nextInt or nextFloat before reading the string:  When using the Scanner class in Java to read input from the console, there may be cases where a newline character is left in the input buffer after reading a numeric value using the nextInt() or nextFloat() methods. This can cause issues when trying to read a String input afterwards, as the newline character will be read as the input and the String will not be read properly.  To clear the newline character from the input buffer before reading a String input, you can use the nextLine() method of the Scanner class. Here's how you can do it:   1. After reading the numeric input using nextInt() or nextFloat(), call the nextLine() method once to clear the newline character from the input buffer. 2. Then, you can read the String input using the nextLine() method as usual. |
| **PROGRAM:** | import java.util.Scanner;  public class Citytemp {  public static void main(String[] *args*) {  int i,j;  int days,cities,day;  float mintemp=1000,maxtemp=-100,tempsearch=-150;  int mincity=0,maxcity=0,minday=0,maxday=0;  String city;  Scanner sc=new Scanner(System.in); // Creating Scanner object to read input from user  System.out.println("Enter number of cities");// Prompting user to enter number of cities  cities=sc.nextInt();  // Prompting user to enter number of days to keep record for  System.out.println("enter number of days to keep record for");  days=sc.nextInt();  // Consuming newline character left behind by nextInt() method  sc.nextLine();  // Initializing arrays to hold city names and temperature data  String[] citynames =new String[cities];  float[][] temps=new float[days][cities];  // Looping through the cities and days to get temperature data for each city  for(i=0;i<cities;i++){  // Prompting user to enter the name of the city  System.out.println("Enter name of city");  citynames[i]=sc.nextLine();  // Prompting user to enter temperature data for each day in the city  System.out.println("Enter temperature in the city for the "+days+" days");  for(j=0;j<days;j++) {  temps[j][i] = sc.nextFloat();  }  // Consuming newline character left behind by nextFloat() method  sc.nextLine();  }  // Printing the table of temperatures across those days in the cities  System.out.println("The table of temperatures across those days in the cities are:");  for(i=0;i<cities;i++){  System.out.printf("| %15s |",citynames[i]);  }  System.out.printf("\n");  for(j=0;j<days;j++){  for(i=0;i<cities;i++){  System.out.printf("| %15.2f |",temps[j][i]);  if(temps[j][i]<mintemp){// Finding the minimum temperature, city and day  mintemp=temps[j][i];  mincity=i;  minday=j;  }  if(temps[j][i]>maxtemp){// Finding the maximum temperature, city and day  maxtemp=temps[j][i];  maxcity=i;  maxday=j;  }  }  System.out.printf("\n");  }  System.out.printf("\n");  // Printing the minimum and maximum temperature along with the city and day they occur  System.out.println("The minimum temperature "+temps[minday][mincity]+" occurs in "+citynames[mincity]+" at day "+(minday+1));  System.out.println("The maximum temperature "+temps[maxday][maxcity]+" occurs in "+citynames[maxcity]+" at day "+(maxday+1));  // Prompting user to enter day and city to find temperature  System.out.println("Enter day and city to find temperature:");  day=sc.nextInt();  sc.nextLine();  city=sc.nextLine();  // Looping through the cities to find the temperature for the given day and city  for(int x=0;x<cities;x++){  if(citynames[x].equals(city)){  tempsearch=temps[day][x];  }  }  // Printing the temperature for the given day and city if input is valid  if(tempsearch!=-150) {  System.out.printf("The temperature in %s at day %d was %f", city, day, tempsearch);  }  else {  System.out.println("Invalid input");  }  } } |
| **RESULT:** | |