***COMPUTER SCIENCE PROJECT***

***NAME: SHAURRYA BAHETI***

***CLASS: XII***

***SECTION: A***

***ROLL NUMBER: 3239***

***SESSION: 2021-2022***

***SCHOOL NAME: M. C. KEJRIWAL VIDYAPEETH***

***Acknowledgements***

I would like to take this opportunity to thank our dear principal sir, Mr. Neelkanth Gupta, for giving me a chance to be in this school with my stream as Science with Computer.

I would like to thank our Computer Teacher Mr. Spondon Ganguli sir, to select such interesting programs for the project that required so much logic and understanding of the subject to complete.

I would like to thank my friends and my family for being as helpful as they could so that I could complete my project on time.

***Introduction***

*OOP:* Object Oriented Programming (OOP) Language or Technique is a type of programming language or technique, in which objects are used to pass information between classes and decrease the programs latency, unlike Procedural Oriented Programming (POP) Language or technique which is completely procedure based.

*JAVA:* Java is a programming language that uses both POP and OOP concepts and technique. It is a case sensitive programming language that is used to make applications that can run on any platform and any computer, and thanks to its web plug-in, you can even run those apps in your browser.



***System Specifications***

*Hardware specifications:*

* 128MB RAM or above
* 400MHz processor or above
* 500MB Free space for storage.

*Software specifications:*

* Windows 7 or later with a 64bit architecture

***PROGRAM 1***

*Question)*

Avoltri Travels were taking a group of 200 tourists to Balaji Temple. A class clTravel is designed, some of whose functions/methods are shown below –

Class name : clTravel

Data members/ instance variables:

int arAge[200] – an array to store the age of 200 tourists.

int arFreqDist[5] – an array to store the number of people in various age groups :-

cell 0 – number of people up to 20 years

cell 1 – number of people between 21 to 40 years

cell 2 – number of people between 41 to 60 years

cell 3 – number of people between 61 to 80 years

cell 4 – number of people above 80 years

Member functions/methods:

clTravel(): Constructor

void fnReadAge(): To input age of 200 tourists in arAge[]

void fnFrequency( ): To fill arFreqDist[ ] from arAge[ ]

void fnShowFreq( ): To show the frequency of age group in a table format.

Specify the class clTravel giving details of the mentioned functions. You need to write the main() method and create an object of the class and call the functions accordingly.

Algorithm)

* Start
* Take Input
* Run a loop over the array and increment the number at the respective position of the frequency distribution
* Display the distribution in tabular form
* End

Code)

import java.util.\*;

class clTravel

{

// Declaring class variables

int arAge[]/\* To store the ages of the tourists \*/, arFreqDist[]/\* To store the age distribution frequency table \*/;

// Default constructor to initialise class variables to default values

clTravel()

{

arAge=new int[200];

arFreqDist=new int[5];

for (int i = 0;i < 5;i++)

{

arFreqDist[i] = 0; // Initialising each element to 0

}

}

// Function to take input of the ages of all the tourists

void fnReadAge()

{

Scanner sc=new Scanner(System.in);

int i = 0/\* LCV \*/, x/\* Temporary variable \*/;

while(i < arAge.length)

{

System.out.print("Enter the age of tourist number " + (i+1) + ": ");

x = sc.nextInt();

// Validating input

if(x <= 0)

{

System.out.println("Age should be greater than zero");

}

else

{

arAge[i] = x; // Putting the validated input in the array

i++;

}

}

}

// Function to fill the distribution array

void fnFrequency()

{

for(int i = 0; i < arAge.length; i++)

{

// Incrementing the respective age group's slot by 1 each time there is a age inside that age group

if(arAge[i] <= 20)

arFreqDist[0]++;

else if(arAge[i] <= 40)

arFreqDist[1]++;

else if(arAge[i] <= 60)

arFreqDist[2]++;

else if(arAge[i] <= 80)

arFreqDist[3]++;

else

arFreqDist[4]++;

}

}

// Function to print the results in tabular form

void fnShowFreq()

{

System.out.println("Age groups:\t1-20\t21-40\t41-60\t61-80\t80<");

System.out.println("Frequency :\t"+arFreqDist[0]+"\t"+arFreqDist[1]+"\t"+arFreqDist[2]+"\t"+arFreqDist[3]+"\t"+arFreqDist[4]);

}

// Main method to create object and call the required functions

public static void main(String[] args)

{

clTravel clt=new clTravel();// Creating object

// Calling functions accordingly

clt.fnReadAge();

clt.fnFrequency();

clt.fnShowFreq();

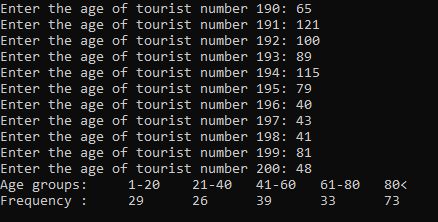
}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **DataType/ReturnType** | **Variable/Method** | **Description** |
| **1** | int[] | arAge | An array to store the age of 200 tourists. |
| **2** | int[] | arFreqDist | An array to store the number of people in  various age groups |
| **3** | - | clTravel() | Default constructor |
| **4** | void | fnReadAge() | To input age of 200 tourists in arAge[] |
| **5** | void | fnFrequency() | To fill arFreqDist[ ] from arAge[ ] |
| **6** | void | fnShowFreq() | To show the frequency of age group in a table format |
| **7** | void | main(String[]) | To create an object of the class and call the functions accordingly |

Output)

THE UPPER PART OF THE OUTPUT WAS CUT TO MAKE IT SHORT.

***Program 2***

Question)

Write a program in Java that will accept a string from the user that must be comprises of N sentences, where N will be entered by the user and must be greater than 1. Every sentence in the string must be ended with either ‘?’ or ‘.’ or ‘!’. The string should be a continuous string entered by the user. If the input string does not match with the conditions given above, user should be asked to re-enter the string again. Now ask the user to choose a sentence from the entered string and print that sentence on the screen in Sentence case. Now print a histogram on the frequency of each alphabet present in that sentence with the help of ‘#’ or ‘\*’ only. The histogram should be a column chart on the above-mentioned data.

Algorithm)

* Start
* Take required inputs
* Store the sentences in an array where each element is a sentence.
* According to user choice get the sentence to work on
* Make an array containing the frequency of each alphabet in the sentence
* Make a 2-dimensional array with one side being 26 and the other side being the frequency of the most appearing alphabet
* Fill that array originally with “ “.
* Now for each alphabet’s frequency fill that column up to the frequency of the alphabet with “\*” in place of “ “.
* Print the 2-dimensional array
* End

Code)

import java.util.\*;

class Alphagram

{

// Declaring class variables

String s\_arr[]; // To store the sentences in array

int n /\* To store number of sentences user will enter \*/, choice/\* To store number of sentence user wants a graph for \*/;

// Default constructor to initialise class variables to default values

Alphagram()

{

n = 0;

}

// Function to take input of an integer with explicit error handling recursively

int get\_int(String s)

{

Scanner sc=new Scanner(System.in);

System.out.print(s);

try

{

int x = sc.nextInt();

return x;

}

catch(Exception e) // Catching error when user enters something other than integer

{

System.out.println("\nPlease enter an integer only");

get\_int(s);

}

return 0;

}

// Function to take necessary inputs and checking them as well

void input()

{

while(true)

{

n = get\_int("Enter the number of sentences for the string : ");

if(n > 1) // Checking if the number complies with the rules

break; // Breaking when true

System.out.println("number of sentences should be at least 2");

}

s\_arr = new String[n]; // Initialise the array

while(true)

{

Scanner sc=new Scanner(System.in);

// Taking input of the sentences

System.out.println("Enter a string having " + n + " sentences ended with either ‘?’ or ‘.’ or ‘!’");

String s = sc.nextLine();

s = s.trim();

int l = s.length();

// Filling the array with the sentences

String ns="";

int a = 0;int e = 0;

for(int i = 0; i < l; i++)

{

char ch = s.charAt(i);

if(ch=='?'||ch=='.'||ch=='!')

{

try

{

s\_arr[a] = ns + ch;

a++;

}

catch(ArrayIndexOutOfBoundsException ex) // catching error when user enters more sentences

{

System.out.println("Please only enter as many sentences as you specified");

e = 1;

break;

}

ns = "";

}

else if(a<n && i==l-1) // Checking if user entered less sentences as required

{

System.out.println("Please enter as many sentences as you specified");

e = 1;

break;

}

else

{

if(i == l-1) // Checking if the string does not follow the rules

{

System.out.println("Please make sure your sentences are properly punctuated");

e = 1;

break;

}

else // Else appending sentence

ns = ns + ch;

}

}

if(s\_arr[n-1]==null) // Final check if the user entered less sentences

System.out.println("Please enter as many sentences as you specified");

else if(e==0)

{

break; // finally breaking out of the infinite loop

}

}

while(true)

{

// Taking input of the choice of sentence from user

choice = get\_int("Enter the sentence number of your choice : ");

if(choice >= 1 && choice <= n)

break; // breaking from infinite loop if the input is valid

System.out.println("The choice is not within the limits of number of sentences");

}

}

// Function to get the frequency of every alphabet in the sentence of user's choice

int[] AlphaFrequency(String s)

{

int freq[]=new int[26]; // initialising the array for frequencies

for(int i = 0;i < 26;i++)

freq[i] = 0; // initialising every element to 0

s = s.toUpperCase(); // Converting to uppercase to make things easy

int l = s.length();

for(int i = 0; i < l; i++)

{

int p = (int)(s.charAt(i)) - 65;

if(p >= 0 && p < 26)

freq[p]++; // increasing the frequency of the alphabet found by 1

}

return freq; // returning the array containing the frequencies

}

// Function to create a 2-dimensional array to print

char[][] ArrToPrint(int max, int arr[])

{

char ch[][] = new char[max+1][26]; // The array to be printed is initialised

for(int i = 0; i < max; i++)

for(int j = 0; j < 26; j++)

ch[i][j] = ' '; // Initialising each element of the array to a space

char c = 'A';

for(int i = 0; i < 26; i++)

{

ch[max][i] =(char)(c + i); // Initialising the last row of the array to each alphabet respectively

}

// Filling the array with \* to indicate 1 instance of the respective alphabet

for(int i = 0; i < 26; i++)

{

int a = max - 1;

while(arr[i] > 0)

{

ch[a][i] = '\*';

arr[i]--;

a--;

}

}

return ch; // Returning the array to be printed

}

// Function to print the graph

void display()

{

input(); // Calling input

String s = s\_arr[choice - 1]; // Getting the sentence of choice

int array[] = AlphaFrequency(s); // Getting the frequency of each alphabet in the sentence of choice

// Getting the maximum frequency shown by any alphabet

int max = 0;

for(int i = 0; i < array.length; i++)

if(array[i] > max)

max = array[i];

char arr[][] = ArrToPrint(max, array); // Getting the array to be printed

// Printing the things required

System.out.println("The sentence of your choice is : " + s);

for(int i = 0; i < max+1; i++)

{

for(int j = 0; j < 26; j++)

{

System.out.print(arr[i][j] + " ");

}

System.out.println();

}

}

// Main method to create object and call required functions

public static void main(String[] args)

{

Alphagram alpg=new Alphagram();

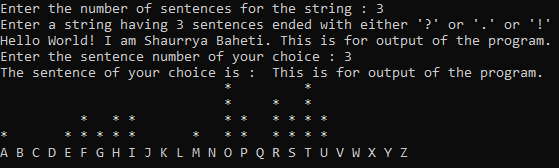
alpg.display();

}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| **1** | String[] | s\_arr | To store the sentences in array |
| **2** | int | n | To store number of sentences user will enter |
| **3** | int | choice | To store number of sentence user wants a graph for |
| **4** | - | Alphagram() | Default Constructor |
| **5** | int | get\_int(String) | to take input of an integer with explicit error handling recursively |
| **6** | void | input() | to take necessary inputs and checking them as well |
| **7** | int[] | AlphaFrequency(String) | to get the frequency of every alphabet in the sentence of user's choice |
| **8** | char[][] | ArrToPrint(int, int[]) | to create a 2-dimensional array to print |
| **9** | void | display() | to print the graph |
| **10** | void | main(String[]) | create object and call required functions |

Output)

Program 3

Question)

Input a sentence from the user and change the vowels to next alphabet and the consonant to previous alphabet of that word in that string. Display both the input and changed string. Class description is given below:-

Class name : WordChange

Data members/instance variable

* String s: input string
* String r: resultant string

Member functions:-

* WordChange(String): parameterized constructor to initialize the data members accordingly
* String change(String): return the changed word as instructed in the question
* void extract(void): extract each word from the string s and pass to the change() and form the new string.
* void display(): display both the strings

Specify the class WordChange giving the details of constructor and all the functions mentioned in the question. You need to write the main() method and create an object of the class and call the functions accordingly.

Algorithm)

* Start
* Take required inputs
* Make an array of extracted words and send them one by one to change(String)
* Int change(String), loop over the extracted word
* Increment each vowel character by 1 and decrement each consonant character by 1 and add the formed character to a new string
* Return the changed string
* Form the new string with the changed words
* Display the changed string
* End

Code)

import java.util.\*;

class WordChange

{

// Declaring class variables

String s/\* To store the string entered by user \*/, r/\* To store the changed string \*/;

// Parameterized constructor to initialise the class variables

WordChange(String str)

{

s = str;

r = "";

}

// Function to change the passed word

String change(String word)

{

int l = word.length(); // Length of the word passed

String chWord = ""; // To store the changed word

for(int i = 0; i < l; i++)

{

char ch = word.charAt(i); // getting the character at position i in the word

char cH = Character.toUpperCase(ch); // Getting the uppercase version as well to make the work easy

if(cH >= 'A' && cH <= 'Z') // Checking if the character is an alphabet or not

{

if(cH=='A'||cH=='E'||cH=='I'||cH=='O'||cH=='U') // Checking if the character is a vowel or not

{

chWord = chWord + (char)(ch+1); // Changing the vowel with the next alphabet and adding it to chWord

}

else

{

chWord = chWord + (char)(ch-1); // Changing the consonant with the previous alphabet and adding it to chWord

}

}

else

chWord = chWord + ch; // Adding the non-alphabet character as it is

}

return chWord; // Returning the changed word

}

// Function to extract each word and pass it to the change function then add the changed word to new string

void extract()

{

String arrS[] = s.split(" "); // Getting the array of words

int l = arrS.length; // Length of the array

for(int i = 0; i < l; i++)

{

r = r + change(arrS[i]) + " "; // Adding the changed words to new string

}

r = r.trim(); // Trimming extra spaces

}

// Function to display the original string and changed string

void display()

{

extract();

System.out.println("Entered string : " + s);

System.out.println("Changed String : " + r);

}

// Main method to create object and call required functions

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

// Taking input of the string to change

System.out.print("Enter a string : ");

String str = sc.nextLine();

WordChange wch=new WordChange(str);

wch.display();

}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | String | s | input string |
| 2 | String | r | resultant string |
| 3 | - | WordChange(String) | parameterized constructor to initialize the data members accordingly |
| 4 | String | change(String) | return the changed word as instructed in the question |
| 5 | void | extract() | extract each word from the string s and pass to the change() and form the new string. |
| 6 | void | display() | display both the strings |
| 7 | void | main(String[]) | create an object of the class and call the functions accordingly |

Output)

Program 4

Question)

A tourist company plans to organize tour to visit N major cities of India. WAP to accept minimum name of 4 cities and the starting location from the user. Now print the possible combinations to travel all those cities so that each city is to be routed only once.

Algorithm)

* Start
* Take required inputs
* Call the recursive function and give the parameters an empty array and the array with the cities without the selected city
* Base case of the recursive function prints the route number, selected city, the first array which initially was empty, and the second array if it is not empty, if the second array’s size is less than or equal to 1.
* In the recursive case there is loop that loops over the cities array, takes a city and joins the left and right side of the array from the taken city, adds the taken city to the empty array and then calls itself again, with the parameters being the empty array + the taken city, and the left array + the right array.
* End

Code)

import java.util.\*;

class Routes

{

// Declaring class variables

int n/\* To store the number of cities \*/, rn/\* To store the route number \*/;

String cities[]/\* To store the cities \*/, start/\* To store the starting city \*/;

// Default Constructor to initialize the class variables to default values;

Routes()

{

n = 0;

rn = 1;

start = "";

}

// Function to validate ang get the number of cities

void get\_number(String s)

{

Scanner sc=new Scanner(System.in);

System.out.print(s);

try

{

n = sc.nextInt();

if(n < 4) // Validating input

{

System.out.println("At least 4 cities are required");

get\_number(s); // Calling the function again when user entered an invalid input

}

}

catch(Exception e) // Handling error when user enters something other than an integer

{

System.out.println("\nPlease enter an integer only");

get\_number(s); // Calling the function again when user entered an invalid input

}

cities = new String[n]; // Initializing the cities array

}

// Function to validate and get the names of cities from the user

void get\_cities(String str)

{

// Taking input

Scanner sc=new Scanner(System.in);

System.out.println(str);

String s = sc.nextLine();

s = s.trim(); // Trimming extra spaces

int l = s.length();

String ns=""; // temporary variable

int a = 0, e = 0;

for(int i = 0; i < l; i++)

{

char ch = s.charAt(i);

if(ch==',' || ch==';')

{

try

{

cities[a] = ns.trim(); // Placing the name of the city into the cities

if(ch==';')

break; // breaking from loop when semi colon detected

a++;

}

catch(ArrayIndexOutOfBoundsException ex) // catching error when user enters more cities

{

System.out.println("Please only enter as many cities as you specified");

e = 1;

break;

}

ns = "";

}

else if(a<n && i==l-1) // Checking if the user entered less number of cities

{

System.out.println("Please enter as many cities as you specified");

e = 1;

break;

}

else

{

if(i == l-1) // Checking if the user ended the list with a semi colon or not

{

System.out.println("Please make sure you placed a semi-colon(;) at the end");

e = 1;

break;

}

else

ns = ns + ch;

}

}

if(cities[n-1]==null && e==0) // Final checking if the user entered less number of cities

{

System.out.println("Please enter as many cities as you specified");

get\_cities(str);

}

else if(e!=0) // if there was any error

get\_cities(str); // calling the function again

else

{

// Checking if the user entered the same city twice

For:

for(int i = 0; i < n; i++)

{

String city = cities[i];

for(int j = i + 1; j < n; j++)

if(city==cities[j])

{

System.out.println("You entered one city two times");

get\_cities(str); // calling function when invalid input found

break For;

}

}

return;

}

}

// Function to get the name of the starting city

void get\_starting\_city()

{

// Taking input

Scanner sc=new Scanner(System.in);

System.out.print("Enter the city you want to start with : ");

start = sc.nextLine().trim(); // trimming extra spaces

//Checking if the name of the city exists in the list entered

boolean found = false; // setting found as false

int i = 0; // LCV

while(i < n)

{

if(found==true)

break; // breaking if already found

if(cities[i].equals(start))

found=true; // setting found as true if found

i++;

}

if(found)

{

// if found, removing the city from the array to make things easy

String newarr[]=new String[n-1]; // initialising a temporary array with size 1 less than the original array

int a = 0;

for(int j = 0; j < n; j++)

{

if(j==i-1)

continue; // continuing if the city is the starting city

else

{

newarr[a] = cities[j];

a++;

}

}

cities=new String[n-1]; // re-initialising the original array with size 1 less than before

for(int j = 0; j < n-1; j++)

cities[j] = newarr[j]; // re-filling the array using the temporary array we just filled

return;

}

else

{

System.out.println("The city was not found in the list");

get\_starting\_city(); // If the city was not found re-running the function

}

}

// Displaying the possible routes

void display(String s1[], String s2[])

{

if (s2.length <= 1)

{

int l = s1.length + s2.length;

int l1 = s1.length;

System.out.print("Route " + rn + ": " + start + " => "); // Printing the route generated

for(int i = 0; i < l; i++)

{

// The if-ese block is for printing the route generated

if(i < l1)

{

if(i<l-1)

System.out.print(s1[i] + " => ");

else

System.out.println(s1[i]);

}

else

{

if(i == l-1)

System.out.println(s2[i-l1]);

else

System.out.print(s2[i-l1] + " => ");

}

}

rn++; // incrementing the route number

}

else

{

// Loop to generate routes

for (int i = 0; i < s2.length; i++)

{

String x[]=new String[1]; // Temporary array to store the i th element of s2 array

x[0] = s2[i]; // initialising the only term in the array to the i th element in the s2 array

String y[]=new String[i]; // Temporary array to store all the elements before the i th element in the s2 array

for(int j = 0; j < i; j++)

y[j] = s2[j]; // initialising the elements of the array

String z[]=new String[s2.length - (i+1)]; // Temporary array to store all elements in s2 array after the i th element

for(int j = 0; j < z.length; j++)

z[j] = s2[i+j+1]; // initialising the elements of the array

String p[]=new String[s1.length + x.length]; // Temporary array to store the elements of s1 and x arrays respectively

for(int j = 0; j < p.length; j++) // Filling the array p

if(j < s1.length)

p[j] = s1[j];

else

p[j] = x[j-s1.length];

String q[]=new String[y.length + z.length]; // Temporary array to store the elements of y and z arrays respectively

for(int j = 0; j < q.length; j++) // Filling the array q

if(j < y.length)

q[j] = y[j];

else

q[j] = z[j-y.length];

display(p, q); // Calling the function with p and q as the parameters

}

}

}

// Function that calls the display function because display function is recursive

void displayRoutes()

{

System.out.println("Possible routes are :-");

String x[]=new String[0]; // initialising a null array

display(x, cities); // calling the display function with null array and cities array respectively

}

// Main method to create object and call the required functions

public static void main(String[] args)

{

Routes rts=new Routes();

rts.get\_number("Enter the number of cities you want to travel to : ");

rts.get\_cities("Enter the names of the cities SEPERATED BY COMMAS(,) and END THE LIST BY A SEMICOLON(;)");

rts.get\_starting\_city();

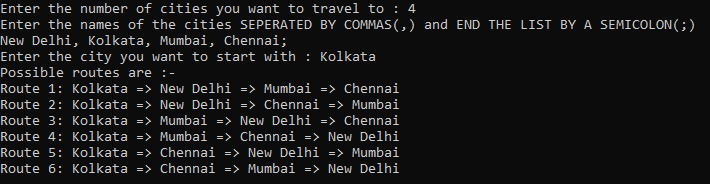
rts.displayRoutes();

}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | int | n | To store the number of cities |
| 2 | int | rn | To store the route number |
| 3 | String[] | cities | To store the cities |
| 4 | String | start | To store the starting city |
| 5 | - | Routes() | Default Constructor |
| 6 | void | get\_number(String) | to validate and get the number of cities |
| 7 | void | get\_cities(String) | to validate and get the names of cities from the user |
| 8 | void | get\_starting\_city() | to get the name of the starting city |
| 9 | void | display(String[], String[]) | to display the possible routes |
| 10 | void | displayRoutes() | To call the display function because display function is recursive |
| 11 | void | main() | to create object and call the required functions |

Output)

Program 5

Question)

Anagram of a word is all the possible combination of alphabets present in that particular word. Write a program in Java to accept one single word of any length from the user and print the anagrams of that word. The program should check that the word should consist of alphabets and can be of any length. Also print the total number of words displayed.

Algorithm)

* Start
* Take necessary inputs
* Call the recursive function and give the parameters an empty String and the word
* Base case of the recursive function, first checks if the formed anagram was already printed before if yes it just returns, if no it prints the first string which initially was empty, and the second string if it is not empty, and adds the printed anagram to the anagram collection for future use. If the second string’s length is less than or equal to 1, base case is fulfilled.
* In the recursive case there is loop that loops over the word, takes a character and joins the left and right side of the word from that character, adds the character to the empty string position parameter, and then calls itself again, with the parameters being the empty string position parameter + the character, and the left + the right of that word from the character.
* After the complete recursion is over print the number of anagrams formed
* End

Code)

import java.util.\*;

class Anagram

{

// Declaring class variables

String word; // To store the word entered by user

int count; // To store the number of anagrams generated

String anag[]; // To store the anagrams generated

// Default constructor to initialise class variables to default values

Anagram()

{

count = 0;

word = "";

anag = new String[0];

}

// Function to take input

void input()

{

// Taking input for the word with whose letters we need to print the anagrams

Scanner sc = new Scanner(System.in);

System.out.print("Enter a word : ");

word = sc.next();

}

// Function to display the possible anagrams recursively

void display(String s1, String s2)

{

if (s2.length() <= 1)

{

// Checking if the anagram generated is already printed or not

String newanag = (s1 + s2).toUpperCase();

boolean found = false; // Initialising found to false to denote not found

for(int i = 0; i < anag.length; i++)

{

if(anag[i].equalsIgnoreCase(newanag)) // Checking if found or not

{

found = true; // Changing the found to true

break; // Breaking out of loop if found

}

}

if(found==false) // When not printed already

{

count++; // Increase the count of the anagrams by 1

System.out.println(newanag); // Print the anagram

// Add the new anagram to the previous list

String temp[]=new String[anag.length]; // Initialising a temporary array

for(int i = 0; i < temp.length; i++)

temp[i] = anag[i]; // Filling the temporary array

anag=new String[temp.length + 1]; // Re-initialising the original array

anag[0] = newanag; // Adding the anagram just printed

for(int i = 1; i < anag.length; i++)

{

anag[i]=temp[i-1]; // Re-filling the original array using the temporary array

}

}

}

else

{

// Loop to generate anagrams

for (int i = 0; i < s2.length(); i++)

{

String x = s2.substring(i, i + 1); // Getting the i th character as a String

String y = s2.substring(0, i);

String z = s2.substring(i + 1);

display(s1 + x, y + z); // Calling the function again with changed parameters

}

}

}

// Function to display whatever is required

void display()

{

System.out.println("The Anagrams of the word "+ word +" are : ");

display("", word);

System.out.println("Total Number of Anagrams = " + count);

}

// Main method to declare object and call required functions

public static void main(String args[])

{

Anagram ang = new Anagram();

ang.input();

ang.display();

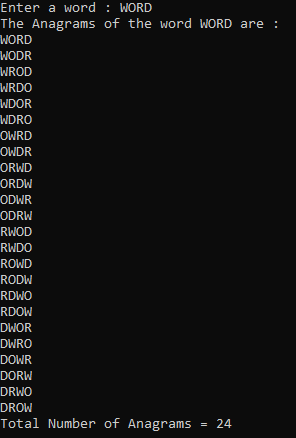
}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | String | word | To store the word entered by user |
| 2 | int | count | To store the number of anagrams generated |
| 3 | String[] | anag | To store the anagrams generated |
| 4 | - | Anagram() | Default constructor |
| 5 | void | input() | to take input |
| 6 | void | display(String, String) | to display the possible anagrams recursively |
| 7 | void | display() | to display whatever is required |
| 8 | void | main(String[]) | to declare object and call required functions |

Output)



Program 6

Question)

A prime number is a number that is divisible by 1 and that number. Twin prime numbers are the pair of 2 prime numbers whose difference is 2, e.g. (3,5), (5,7), (11,13) etc. The sum of reciprocals of the twin primes converges to a sum, known as Brun’s Constant. Declare a class named “Primes” with one data member double sum, and three member functions, int primeCheck( int, int ), double sumTwinPrime( int ) and void BrunConstant ( int ). WAP to declare the above class with its member functions. Use recursive technique in primeCheck( ) function. Write the main method also.

Algorithm)

* Start
* Take necessary inputs
* Check for prime in the recursive function
* Add the twin primes in another function
* Run a loop from 1 to the number user enters, increment only if there was something added to the sum.
* Check for primes from 3, find twin prime and pass the first number in the twin prime pair to the sumTwinPrime function which will give the sum of the twin primes, add that sum to the sum of the previous twin primes.
* Once the loop is over, display the sum as Brun’s Constant.
* End

Code)

import java.util.\*;

class Primes

{

// Declaring class variables

double sum;// To store the Brun constant

// Function to check if the number is prime or not

int primeCheck(int n, int i)

{

// Base cases

if(n==3)

return n;

if(n % i==0)

return 0;

if(i>=(int)Math.sqrt(n))

return n;

// Recursive case

return primeCheck(n, i+1);

}

// Function to calculate the sum to the reciprocal of twin primes

double sumTwinPrime(int n)

{

return (1.0/n) + (1.0/(n+2));

}

// Function to calculate the Brun’s constant

void BrunConstant(int n)

{

sum = 0.0;// Sum at the start

int a = 1;// Term number

int p = 3;// Number to check

// Calculating the constant

while(a <= n)

{

if(primeCheck(p, 2)==p && primeCheck(p+2, 2)==p+2)

{

a++;

sum += sumTwinPrime(p);

}

p++;

}

}

// Main method to create object and call functions accordingly

public static void main(String[] args)

{

Primes prm=new Primes();// Creating object

// Taking input and calling functions

Scanner sc=new Scanner(System.in);

System.out.print("Enter the number of terms : ");

int n = 0;

try

{

n = sc.nextInt();

// Validating the input

if(n<=0)

{

System.out.println("Number of terms should be more than 0");

main(args);// Calling function again if invalid input detected

return;

}

else

{

prm.BrunConstant(n);

System.out.println("The Brun's Constant for " + n + " is " + prm.sum);

}

}

catch(Exception e)// Input mismatch

{

System.out.println("Enter an integer only please");

main(args);// Calling function again if invalid input detected

return;

}

}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | double | sum | To store the Brun constant |
| 2 | int | primeCheck(int, int) | to check if the number is prime or not |
| 3 | double | sumTwinPrime(int) | to calculate the sum to the reciprocal of twin primes |
| 4 | void | BrunConstant(int n) | to calculate the Brun’s constant |
| 5 | void | main(String[]) | to create object and call functions accordingly |