***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.

**~SHAURRYA BAHETI**

*Index*

|  |  |  |
| --- | --- | --- |
| Sl. No. | Contents | Page No. |
| 1 | Introduction to OOPs and JAVA | 4 |
| 2 | System Specifications | 5 |
| 3 | Program 1 | 6 |
| 4 | Program 2 | 11 |
| 5 | Program 3 | 23 |
| 6 | Program 4 | 30 |
| 7 | Program 5 | 45 |
| 8 | Program 6 | 52 |
| 9 | Program 7 | 58 |
| 10 | Program 8 | 67 |
| 11 | Program 9 | 74 |
| 12 | Program 10 | 85 |
| 13 | Program 11 | 92 |
| 14 | Program 12 | 99 |
| 15 | Program 13 | 114 |
| 16 | Program 14 | 124 |
| 17 | Program 15 | 132 |
| 18 | Program 16 | - |
| 19 | Program 17 | - |
| 20 | Program 18 | - |
| 21 | Program 19 | - |
| 22 | Program 20 | - |
| 23 | Bibliography | 142 |

***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();

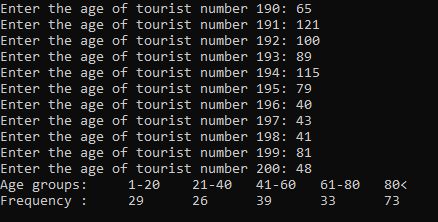
}

}

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();

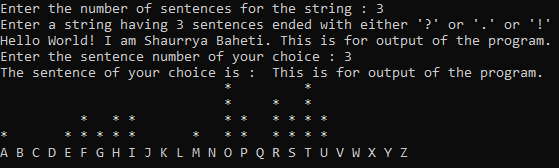
alpg.display();

}

}

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 a smaller 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 a smaller 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();

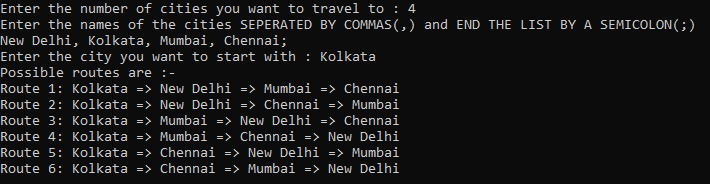
rts.displayRoutes();

}

}

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();

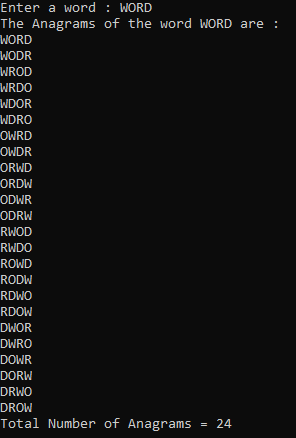
}

}

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 |

Output)

***Program 7***

Question)

Write a program to perform the following task: Input a number, say mak. Now create a number, say tip, by arranging the digits of the number in ascending order. Create another number, say tap, by arranging the arranging digits of the number in descending order. List all the Perfect Squares that lie in between tip and tap. [A perfect square is a number that has an integer square root.]

Algorithm)

* Start
* Take necessary inputs
* Use a recursive function to make tap
* Reverse Tap to get Tip
* Run a loop from tip up to Tap
* Check every number in between for perfect square
* If perfect square then print it
* End

Code)

import java.util.\*;

class TipTap

{

// Declaring class variables

int tip/\* stores the lower bound \*/, tap/\* stores the upper bound \*/, mak/\* Stores the number user enters \*/;

// Default constructor

TipTap()

{

// Initialising class variables to default values

tip = 0;

tap = 0;

mak = 0;

}

// Function that takes input

void input()

{

try

{

// Taking input

Scanner sc=new Scanner(System.in);

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

mak = sc.nextInt();// Getting mak

}

catch(Exception e)// Input mismatch

{

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

input();// Calling function again

}

}

// Function that removes a digit i from a number n

int rmDigit(int n, int i)

{

int a = 0;// Stores the position of the number

for(int j=n, k=1; j>0; j/=10, k++)

{

if(j%10==i)

{

a = k;// Storing the position of i

break;

}

}

if(a==1)

return n/10;// if i is the first digit from right returning n/10

else// else

{

int p = 1;

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

p \*= 10;// Place value of i

return (n/p)\*(p/10) + (n%(p/10));// Removing i from the number and returning the new number

}

}

// Sorting a number in descending order

int desc(int n)

{

if(n<10)

return n;// Returning the number if the number is only 1 digit

int min = 9;// To store the smallest digit in n, initially the highest possible digit

for(int i=n; i>0; i/=10)

{

if(i%10 <= min)

min=i%10;// storing the smallest digit in n

}

n = rmDigit(n, min);// Removing the min from the number

return desc(n)\*10 + min;// Recursively returning the number in descending order

}

// Reversing a number

int reverse(int n)

{

if(n < 10)

return n;// Returning the number if the number is only 1 digit

int c = 0;// Number of digits in the number

for(int i = n; i > 0; i/=10)

c++;

int p = 1;// Highest place value of the number

for(int i = 1; i < c; i++)

p\*=10;

return (n%10)\*p + reverse(n/10);// Recursively returning the reverse of the number

}

// Function to check if a number is a perfect square

boolean perfectSqr(int n)

{

double sqrt = Math.sqrt(n);// Square root with decimal

if(sqrt == Math.floor(sqrt))// Checking if there is nothing after the decimal

return true;// If yes then it is perfect square

else

return false;// else it is not a perfect square

}

// Function to display the required things

void display()

{

tap = desc(mak);// Getting tap

tip = reverse(tap);// Getting tip

// For better styling

String tp = Integer.toString(tip);

String tP = Integer.toString(tap);

while(tp.length()!=tP.length())

tp = "0" + tp;

// Printing tip and tap

System.out.println("New Number 1 : " + tp);

System.out.println("New Number 2 : " + tP);

// Printing the desired list

System.out.println("The desired list is -");

for(int i = tip; i <= tap; i++)

{

if(perfectSqr(i))

System.out.print(i + " ");// Printing a number if it is a perfect square

}

System.out.println();// Changing the line

}

// Main method to create objects and call functions accordingly

public static void main(String[] args)

{

TipTap tPt=new TipTap();// Creating object

// Calling functions accordingly

tPt.input();

tPt.display();

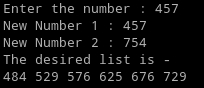
}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | int | tip | stores the lower bound |
| 2 | int | tap | stores the upper bound |
| 3 | int | mak | Stores the number user enters |
| 4 | - | TipTap() | Default Constructor |
| 5 | void | input() | To take input |
| 6 | int | rmDigit(int, int) | To remove a digit from a number |
| 7 | int | desc(int) | To sort a number's digits in descending order |
| 8 | int | reverse(int) | To reverse the digits of a number |
| 9 | boolean | perfectSqr(int) | To check if a number is a perfect square |
| 10 | void | display() | To display the required things |
| 11 | void | main(String[]) | To create object and call functions accordingly |

Output)



***Program 8***

Question)

Given the following series

S = m! / pm + (m-1)! / pm-1 + (m-2)! / pm-2 + … + (m-m)! / pm-m

A class called clSomeSeries has been defined to calculate the sum of the series. Some of the function/ methods in clSomeSeries are shown below:

Class name: clSomeSeries

Data members:

* int m
* int p
* double sum

Member functions/methods:

* clSomeSeries( ): Constructor
* int fnFact(int num): Calculates and returns the factorial of num
* long fnPower(int a, int p): Calculates and returns the value of ap , without using header file <math.h>
* void fnCalculate(void): Calculates the sum of the given series in “sum” using other member functions
* void fnInput(void): Inputs the value of m, p.
* void fnPrint(void): Shows the sum of the series.

Specify the class clSomeSeries, giving details of constructor( ) and mentioned functions. Write the main method also.

Algorithm)

* Start
* Take necessary inputs
* create recursive factorial function
* create recursive power function
* set sum initially to 1.0
* start loop from 1 up to m
* change sum to factorial of i divided by (ith power of p + previous value of sum)
* When the loop ends display the required things
* End

Code)

import java.util.\*;

class clSomeSeries

{

// Declaring class variables

int m/\* Storing the value of m \*/, p/\* Storing the value of p \*/;

double sum;// storing the sum of the series

// Default constructor

clSomeSeries()

{

// Initialising class variables to default values

m = p = 0;

sum = 0.0;

}

// Function that returns the factorial of a number num

int fnFact(int num)

{

if(num == 1 || num == 0)// base case

return 1;

return num \* fnFact(num-1);// recursive case

}

// Function that returns the a to the power p

long fnPower(int a, int p)

{

if(p==0 || a==1)// base case

return 1;

return a \* fnPower(a, p-1);// recursive case

}

// function that calculates the sum of the series

void fnCalculate()

{

sum = 1.0;

for(int i = 1; i <= m; i++)

{

sum = (double)fnFact(i)/(double)((fnPower(p, i)) + sum);

}

}

// Function that takes input

void fnInput()

{

// Taking input

Scanner sc=new Scanner(System.in);

while(true)// While loop used to valid input or else continue taking input

{

System.out.print("Enter the value of m : ");

try

{

m = sc.nextInt();

if(m > 0)// Checking if the input is valid

break;

else// Else continuing with an error message

System.out.println("m cannot be less than 1");

}

catch(Exception e)// Checking if the input was mismatched

{

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

}

}

while(true)// While loop used to valid input or else continue taking input

{

System.out.print("Enter the value of p : ");

try

{

p = sc.nextInt();

if(p > 0)// Checking if the input is valid

break;

else// Else continuing with an error message

System.out.println("p cannot be less than 1");

}

catch(Exception e)// Checking if the input was mismatched

{

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

}

}

}

// Function to print the sum of the series

void fnPrint()

{

System.out.println("The sum of the series for which m = " + m + " and p = " + p + " is : " + sum);// Printing sum

}

// Main method to create objects and call the functions accordingly

public static void main(String[] args)

{

clSomeSeries css=new clSomeSeries();// Creating object

// Calling functions accordingly

css.fnInput();

css.fnCalculate();

css.fnPrint();

}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | int | m | To store the value of m as entered by the user |
| 2 | int | p | To store the value of p as entered by the user |
| 3 | - | clSomeSeries() | Constructor |
| 4 | int | fnFact(int) | To calculate the factorial of a number |
| 5 | long | fnPower(int, int) | To calculate a power of a number |
| 6 | void | fnCalculate() | Uses other functions to calculate the sum of the series |
| 7 | void | fnInput() | To take input |
| 8 | void | fnPrint() | To display the required things |
| 9 | void | main(String[]) | To create object and call functions accordingly |

Output)

***Program 9***

Question)

Write a program to generate a 3X3 magic square i.e., a square in which the sum of all the numbers in each row, in each column. Now display the sum along with the array elements as a single unit.

Finally perform :

1. Row-wise sorting in ascending order
2. Column-wise sorting in descending order

Algorithm)

* Start
* Run a loop from 1 to 9
* place 1 at the middle of the first row
* now decrement the row and column by 1
* if the row becomes -1 shift to the last row and if the column becomes -1 shift to the last column
* try to place the next value in the loop at the new position
* if the position is occupied already, go to the previous column and drop down 1 row and place the loop value there
* continue the same logic until you reach 9
* after the loop ends, print the array
* sort the array row-wise in ascending order and print it
* then sort the array in column-wise descending and print it
* End

Code)

import java.util.\*;

class MagicSqr

{

// Declaring class variables

int[][] msqr;// Array to store the magic square

int n/\* int to store the number of rows \*/, sum/\* int to store the sum of each row or column \*/;

// Default constructor

MagicSqr()

{

// Initialising class variable to appropriate values

n = 3;

msqr=new int[3][3];

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

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

msqr[i][j] = 0;

}

// Function to fill the magic square

void generate()

{

int number = 1;// stores the current number to be placed in the magic square

int row = 0;// stores the row position at which number was stored

int column = n / 2;// stores the column position at which number was stored

int curr\_row;// stores current row position

int curr\_col;// stores current column position

while (number <= n \* n)// loop to fill the magic square

{

msqr[row][column] = number;// placing number at row X column

number++;// incrementing number

curr\_row = row;// updating current row position

curr\_col = column;// updating current column position

row -= 1;// decrementing row position

column += 1;// incrementing column position

// Taking care of out of bounds

if (row == -1)

row = n - 1;

if (column == n)

column = 0;

// if next position is filled shifting to next row

if (msqr[row][column] != 0)

{

row = curr\_row + 1;

column = curr\_col;

// Taking care of out of bounds

if (row == -1)

row = n - 1;

}

}

}

// function to get the sum of a row or column

void findSum()

{

sum = msqr[0][0] + msqr[0][1] + msqr[0][2];

}

// function to sort an array

int[] sort(int arr[], char c)

{

switch(c)

{

case 'a':// Sorting in ascending order using selection sort

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

{

int min = i;

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

{

if(arr[j] < arr[min])

min = j;

}

int temp = arr[min];

arr[min] = arr[i];

arr[i] = temp;

}

break;

case 'd':// Sorting in descending order using selection sort

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

{

int max = i;

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

{

if(arr[j] > arr[max])

max = j;

}

int temp = arr[max];

arr[max] = arr[i];

arr[i] = temp;

}

break;

}

return arr;// returning the sorted array

}

// Function to print arrays after sorting

void sortPrint()

{

System.out.println("\nRow wise sorted array : \n");

// Row wise sorting

int sorted[][] = new int[3][3];

for(int i = 0; i < 3; i++)

{

int temp[] = new int[3];

for(int k = 0; k < 3; k++)

temp[k] = msqr[i][k];

temp = sort(temp, 'a');

for(int k = 0; k < 3; k++)

sorted[i][k] = temp[k];

}

displayArr(sorted);// printing sorted array

System.out.println("\nColumn wise sorted array : \n");

// column wise sorting

for(int i = 0; i < 3; i++)

{

int temp[] = new int[3];

for(int k = 0; k < 3; k++)

temp[k] = msqr[k][i];

temp = sort(temp, 'd');

for(int k = 0; k < 3; k++)

sorted[k][i] = temp[k];

}

displayArr(sorted);// printing sorted array

}

// Function to print an array

void displayArr(int[][] arr)

{

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

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

System.out.println();

}

}

// Display function

void display()

{

findSum();// calling the findSum function

System.out.println("The 3X3 Magic Square(row-wise sum is mentioned at the end of each row"

+ "\nand column-wise sum is mentioned at the end of each column) : \n");

// Printing with sum of the rows and columns mentioned

for(int i = 0; i <= 3; i++)

{

for(int j = 0; j <= 3; j++)

{

if(i==3 && j != 3)

System.out.print("\_\t");

else if(j==3 && i != 3)

System.out.print("|" + sum + "\t");

else if(i==3 && j==3)

continue;

else

System.out.print(msqr[i][j] + "\t");

}

System.out.println();

}

System.out.println(sum + "\t" + sum + "\t" + sum);

// Printing sorted arrays

sortPrint();

}

// Main method to create objects and call functions accordingly

public static void main(String[] args)

{

MagicSqr msq=new MagicSqr();// Creating an object

// Calling functions accordingly

msq.generate();

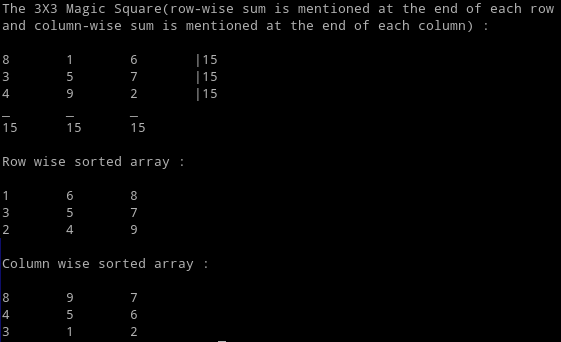
msq.display();

}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | int[][] | msqr | to store the magic square |
| 2 | int | n | to store the number of rows |
| 3 | int | sum | to store the sum of each row or column |
| 4 | - | MagicSqr() | Default Constructor |
| 5 | void | generate() | to fill the magic square |
| 6 | void | findSum() | to get the sum of a row or column |
| 7 | int[] | sort(int[], char) | to sort an array |
| 8 | void | sortPrint() | to print arrays after sorting |
| 9 | void | displayArr(int[][]) | to print an array |
| 10 | void | display() | To display the required things |
| 11 | void | main(String[]) | To create object and call functions accordingly |

Output)

***Program 10***

Question)

Write a program to implement the following: In a private detective department, the trainee detectives were given certain rules for making their passwords.

A password would be considered valid only if

1. It had odd number of characters less than 10.
2. The characters would be alternate alphabets and digits.
3. The alphabets would be only between J and T (both inclusive)
4. No even digit could be present after character J or T

Algorithm)

* Start
* Take input and check for odd characters less than 10 immediately
* check if the alphabets and digits are repeating or alternate
* check if the alphabets lie beyond J and T or not
* check if any J or T in the password is followed by an odd digit or not
* If every condition is satisfied print the validation statement
* If any condition was found false call invalid function with respective message and do System.exit(1)
* End

Code)

import java.util.\*;

class checkPass

{

// Declaring class variables

String pass;// String that stores the password

int l;// Int that stores the length of the password

// Default Constructor

checkPass()

{

// Initialising class variables to default values

pass = "";

l = 0;

}

// Function that takes input

void input()

{

// Taking input

Scanner sc=new Scanner(System.in);

System.out.print("Enter the password : ");

pass = sc.nextLine();

l = pass.length();

// Checking input and calling invalid() function with appropriate message

if(l>10 || l%2==0)

inValid("There are even number of digits or the password is longer than 10 characters");

}

// Function that checks if the alphabets and digits are alternate and also if 'J' or 'T' are followed by an even digit

void alternateCheck()

{

int what = 0;// Variable that checks alternate-ness

char prev = '\0';// Variable that stores the previous character

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

{

char c = pass.charAt(i);

c = Character.toUpperCase(c);// Converting for ease of access

if(c >= 'A' && c <= 'Z')// Checking if the character is alphabet

{

if(c < 'J' || c > 'T')// Checking if the characters beyond the range of 'J' and 'T'

inValid("There are alphabets outside the range of 'J' to 'T'");

else// else setting prev to that character

prev = c;

if(what != 1)// checking if previous character was a number or alphabet

what = 1;

else// If alphabet, password invalid

inValid("There are consecutive alphabets or numbers");

}

else if(c >= '0' && c <= '9')// Checking if the character is number

{

if(prev == 'J' || prev == 'T')// Checking if prev was a 'J' or a 'T'

if(!(c == '1' || c == '3' || c == '5' || c == '7' || c == '9'))// checking if the 'J' or the 'T' was followed by an odd digit

inValid("Alphabet " + prev + " is followed by an even digit");

else// else setting prev to c

prev = c;

if(what != 2)// Checking if previous character was an alphabet

what = 2;

else// If number then, password invalid

inValid("There are consecutive alphabets or numbers");

}

else// if character is neither a number nor an alphabet password invalid

inValid("There are characters other than alphabets or numbers");

}

}

// Function to print invalid with reason

void inValid(String s)

{

System.out.println(s);

System.exit(1);// Exiting program after printing the reason

}

// Main method to create object and call methods

public static void main(String[] args)

{

checkPass chp=new checkPass();// Creating object

// Calling functions accordingly

chp.input();

chp.alternateCheck();

// Printing the password is valid

System.out.println("The password is valid");

}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | String | pass | To store the password to be checked |
| 2 | int | l | To store the length of the password |
| 3 | - | checkPass() | Default Constructor |
| 4 | void | input() | To take input of the password |
| 5 | void | alternateCheck() | To check if the password is valid or not |
| 6 | void | inValid(String) | To print respective message for Invalid password if found |
| 7 | void | main(String[]) | To create object and call functions accordingly |

Output)

***Program 11***

Question)

In Bello Labs, a group of researchers were working on a string balancing project. Their team leader declared that a string would be called “Well Balanced String” if it satisfied the following conditions –

1. No character other than alphabet ‘a’ , ‘z’ would be present in the string
2. against every character ‘a’, character ‘z’ would be present
3. a pair of ‘a’ and ‘z’ would be such that in totality, no ’z’ appears without having a preceding ‘a’ for it.

Algorithm)

* Start
* Take necessary inputs
* Loop over the string
* check if there is any character other than a or z if yes return false
* count the number a and z
* if they are not equal, return false
* create a string array with each element being the character appended to the position and the number of that letter
* check with position if any number of a proceeds the same number of z or vice versa if yes return false
* If no return statement encountered return true
* print the necessary statement
* End

Code)

import java.util.\*;

class StringBalance

{

// Declaring class variables

String s;// String to be checked

int l;// Length of s

// Default constructor

StringBalance()

{

// Initialising class variable to default values

s = "";

l = 0;

}

// Function to take input

void input()

{

// Taking input

Scanner sc=new Scanner(System.in);

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

s = sc.nextLine();

l = s.length();// Storing length in l

}

// Function to check if the string is balanced or not

boolean isBalanced()

{

String str = s.toUpperCase();// Converting the string to uppercase for ease of use

int cA = 0/\* Number of a's in the string \*/, cZ = 0/\* Number of z's in the string \*/;

// Loop to count the number of a's and z's

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

{

char c = str.charAt(i);

if(c!='A' && c != 'Z')

return false;// Returning false if anything other than a or z is present

if(c=='A')

cA++;

if(c=='Z')

cZ++;

}

if(cA != cZ)

return false;// Returning false if number of a's is not equal to number of z's

String st[]=new String[l];// Store the a's and z's with positions

int ca=1/\* Number of a \*/, cz=1/\* number of z \*/;

// Loop to fill st

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

{

char c = str.charAt(i);

if(c=='A')

{

st[a] = Integer.toString(i)/\* Adding position of the character \*/ + c + Integer.toString(ca)/\* Adding the number of a \*/;

ca++;// Incrementing the number of a

}

else

{

st[a] = Integer.toString(i)/\* Adding position of the character \*/ + c + Integer.toString(cz)/\* Adding the number of z \*/;

cz++;// Incrementing the number of z

}

}

// Loop to check if all the nth a's precede the nth z's

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

{

String x = st[i];

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

{

String y = st[j];

if(x.charAt(1)!=y.charAt(1) && x.charAt(2)==y.charAt(2))

{

if(x.charAt(1)=='A' && y.charAt(1)=='Z' && x.charAt(0)>y.charAt(0))

return false;// Returning false if a comes after z

if(x.charAt(1)=='Z' && y.charAt(1)=='A' && x.charAt(0)<y.charAt(0))

return false;// Returning false if z comes before a

}

}

}

return true;// If no previous return statements are called the string is balanced

}

// Function to display the required things

void display()

{

System.out.println("Entered String : " + s);// Printing input string

// Printing if it is a well-balanced string or not

if(isBalanced())

System.out.println("It is a \"Well Balanced String\".");

else

System.out.println("It is NOT a \"Well Balanced String\".");

}

// Main method to create objects and call functions accordingly

public static void main(String[] args)

{

StringBalance stb=new StringBalance();// Creating objects

// Calling functions accordingly

stb.input();

stb.display();

}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | String | s | To store the entered string |
| 2 | int | l | To store the length of the entered string |
| 3 | - | StringBalance() | Default constructor |
| 4 | void | input() | To take input of the string |
| 5 | boolean | isBalanced() | To check if the string is balanced or not |
| 6 | void | display() | To display the required things |
| 7 | void | main(String[]) | To create object and call functions accordingly |

Output)



***Program 12***

Question)

Write a program to convert a roman number to decimal equivalent and vice-versa.

Algorithm)

* Start
* Take necessary inputs
* for roman to decimal
* convert a character to its decimal equivalent
* not check if the next characters decimal value is more than this value, if yes subtract this from the next value
* if no, then continue with adding this to the previous sum
* if all the characters are done converting and values are done adding return the sum
* The sum was the converted decimal, but we need to check if the entered roman was correct. to do so, we will convert this decimal to roman through our function
* if the roman user entered was same as our roman, the roman was correct and we will print it
* else print error and quit
* for decimal to roman
* check if the number is more than 3999, if yes print error message and quit
* else continue with conversion
* start a while loop until number is 0
* if the decimal is more than 1000 (number/1000) number of Ms and, update the number to number % 1000
* else if the number is more than 500 but less than 900, put a D and update the number to number % 500
* else if the number was more than 900, put CM, and update the number to number % 100
* else if the number is more than 100 but less than 400, put (number/100) number of Cs and update the number to number % 100
* else if the number was more than 400, put a CD, and update the number to number % 100
* else if the number is more than 50 but less than 90, put a L and update the number to number % 50
* else if the number was more than 90, put XC, and update the number to number % 10
* else if the number is more than 10 but less than 40, put (number/10) number of Xs and update the number to number % 10
* else if the number was more than 40, put a XL and update the number to number % 10
* else if the number is more than 5 but less than 9, put a V and update the number to number % 5
* else if the number was 9 put a IX and update the number to 0
* else if the number is more than 1 but less than 4, put (number) number of Is and update the number to 0
* else if the number was 4 put a IV and update the number to 0
* continue this until the number is 0 as stated in the while loop
* Once fully converted return the converted roman string, no checking is required
* display the required things
* End

Code)

import java.util.\*;

class Roman

{

// Declaring class variables

int n;// Stores the decimal number

String r;// Stores the roman number

// Default constructor

Roman()

{

// Initialising class variables to default values

n = 0;

r = "";

}

// Function that takes integer input from user

int get\_int(String s)

{

// Taking input

Scanner sc=new Scanner(System.in);

System.out.print(s);

try

{

int x = sc.nextInt();

return x;// Returning input integer

}

catch(Exception e)// Input mismatch

{

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

get\_int(s);// calling function again

}

return 0;

}

// Function to take input

void input()

{

// Taking input of choice

Scanner sc=new Scanner(System.in);

System.out.println("Choices :- \n\t1. Roman to Integer\n\t2. Integer to Roman");

int choice = get\_int("Enter choice : ");

// Going forward according to choice

switch(choice)

{

case 1:// Roman to decimal

System.out.print("Enter a roman numeral : ");

r = sc.next();

display('r');

break;

case 2:// Decimal to roman

n = get\_int("Enter an integer : ");

while(n <= 0)// Checking for valid input

{

System.out.println("Integer to convert must be greater than ZERO");

n = get\_int("Enter a POSITIVE integer : ");

}

display('i');

break;

default:// Wrong choice

System.out.println("Wrong choice");

input();

break;

}

}

// Function that returns the decimal value of a roman literal

int value(char r)

{

if (r == 'I')

return 1;

if (r == 'V')

return 5;

if (r == 'X')

return 10;

if (r == 'L')

return 50;

if (r == 'C')

return 100;

if (r == 'D')

return 500;

if (r == 'M')

return 1000;

return -1;

}

// Function to convert roman to decimal

int romanToDecimal(String str)

{

int res = 0;// resulting decimal number

// Converting

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

{

int s1 = value(str.charAt(i));

if (i + 1 < str.length())

{

int s2 = value(str.charAt(i + 1));

if (s1 >= s2)

{

res = res + s1;

}

else

{

res = res + s2 - s1;

i++;

}

}

else

{

res = res + s1;

}

}

return res;// Returning the converted number

}

// Function to convert decimal to roman

String decimalToRoman(int n)

{

String roman = "";// Resulting roman number

if(n >= 4000)// Validation

{

System.out.println("Numbers more than 3999 are not supported");

System.exit(1);

return "";

}

// Converting

while(n > 0)

{

if(n >= 1000)

{

for(int i = 0; i < n/1000; i++)

roman = roman + "M";

n = n%1000;

}

else if(n >= 500)

{

if(n < 900)

{

roman = roman + "D";

n = n%500;

}

else

{

roman = roman + "CM";

n = n%100;

}

}

else if(n >= 100)

{

if(n < 400)

{

for(int i = 0; i < n/100; i++)

roman = roman + "C";

n = n%100;

}

else

{

roman = roman + "CD";

n = n%100;

}

}

else if(n >= 50)

{

if(n < 90)

{

roman = roman + "L";

n = n%50;

}

else

{

roman = roman + "XC";

n = n%10;

}

}

else if(n >= 10)

{

if(n < 40)

{

for(int i = 0; i < n/10; i++)

roman = roman + "X";

n = n%10;

}

else

{

roman = roman + "XL";

n = n%10;

}

}

else if(n >= 5)

{

if(n < 9)

{

roman = roman + "V";

n = n%5;

}

else

{

roman = roman + "IX";

n = 0;

}

}

else

{

if(n < 4)

{

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

roman = roman + "I";

n = 0;

}

else

{

roman = roman + "IV";

n = 0;

}

}

}

return roman;// Returning converted number

}

// Function to display the required things

void display(char choice)

{

// Going according to user choice

switch(choice)

{

case 'r':// Roman to decimal

int convt = romanToDecimal(r);// Temporary storage of the value for validation

if(convt == romanToDecimal(decimalToRoman(convt)))// Validating the roman number entered

System.out.println(r + " converted to integer is " + romanToDecimal(r));// Printing if valid

else// Else exiting with message

{

System.out.println("It is not a valid Roman number");

System.exit(1);

}

break;

case 'i':// Decimal to roman

System.out.println(n + " converted to roman is " + decimalToRoman(n));// No checking required

break;

}

}

// Main method to create objects and call functions accordingly

public static void main(String args[])

{

Roman rmn = new Roman();// Creating object

// Calling functions accordingly

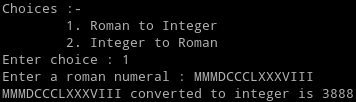
rmn.input();

}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | int | n | To store the decimal number |
| 2 | String | r | To store the roman number |
| 3 | - | Roman() | Default constructor |
| 4 | int | get\_int(String) | To take an integer input from user |
| 5 | void | input() | To take necessary inputs from the user |
| 6 | int | value(char) | to return the decimal equivalent of a roman character |
| 7 | int | romanToDecimal(String) | To convert a roman number to equivalent decimal number |
| 8 | String | decimalToRoman(int) | To convert a decimal number to equivalent roman number |
| 9 | void | display(char) | To display the required things |
| 10 | void | main(String[]) | To create objects and call the functions accordingly |

Output)

***Program 13***

Question)

Write a program to convert a decimal number (whole number/fractional number) into equivalent Binary, Octal and Hexadecimal number form in a menu driven logic.

Algorithm)

* Start
* Take necessary inputs
* Store the base to be converted to in a variable
* create an alphabet array for future use to make things easier
* if the number is negative make it positive but store the sign in another variable
* now take the integer part of the number entered
* start while loop until integer part is 0
* take mod of the integer part with the base that you stored, if mod < 10, concat it to the result from front
* if mod >= 10, concat the alphabet at the position mod - 10 of the array created before
* update the number to number / base
* after the loop is over, add the sign if any, check if the entered number had any decimal part if no return
* else take the fractional part and start converting that
* multiply the fractional part with the decimal to base that you stored, then take the integer part of the product
* if the integer part of the product is < 10, concat it to the result after decimal from back
* if the integer part is >= 10, concat the alphabet at the position integer part - 10 of the array created before
* continue the loop until the fractional part is 0 or the number of loops exceeds 10
* print the converted number
* End

Code)

import java.util.\*;

class Number

{

// Declaring class variables

double d;// To store the original decimal number

char choice;// To store the base to convert in

String converted;// To store the converted number

// Default Constructor

Number()

{

// Initialising the class variables to their default values

d = 0.0;

choice = '\0';

converted = "";

}

// Function to take input of the decimal number

void inputNum()

{

try

{

// Taking input

Scanner sc=new Scanner(System.in);

System.out.print("Enter a decimal number : ");

d = sc.nextDouble();

}

catch(Exception e)// Catching input mismatch exception and other exceptions

{

System.out.println("Enter a decimal NUMBER only");

inputNum();// Calling the function again if any exception was caught

}

}

// Function to take input of choice of base to convert to

void inputChoice()

{

// Taking input

Scanner sc=new Scanner(System.in);

System.out.println("Choices :-\n\t1. Decimal to Binary\n\t2. Decimal to Octal\n\t3. Decimal to Hexadecimal");

System.out.print("Enter choice : ");

try

{

int n = sc.nextInt();

switch(n)

{

case 1:

choice = 'B';

break;

case 2:

choice = 'O';

break;

case 3:

choice = 'H';

break;

default:// Default case for unavailable choice

System.out.println("Enter a valid choice");

inputChoice();// Calling the function again if there was a wrong choice

break;

}

}

catch(Exception e)// Catching input mismatch exception and other exceptions

{

System.out.println("Enter a valid choice");

inputChoice();// Calling the function again if any exception was caught

}

}

// Function to return the integer part of any double number n

int get\_int\_part(double n)

{

return (int)Math.floor(n);

}

// Function to convert the decimal number to desired base

void convert()

{

// Taking absolute value and checking sign.

char sign = '\0';

if(d < 0)

{

sign = '-';

d \*= -1;

}

// Storing a few values necessary for the conversion

int intPart = get\_int\_part(d);// The integer part of the entered number

int base = choice=='B'?2:choice=='O'?8:16;// Base of conversion

// Converting the integer part of the number

while(intPart > 0)

{

int mod = intPart % base;

if(mod<=9)

converted = Integer.toString(mod) + converted;

else

converted = (char)((int)'A' + mod%10) + converted;

intPart = intPart/base;

}

converted = sign + converted;// Adding the sign to the converted number

if(d - intPart == 0.0)

return;// Returning if the entered number was an integer

else

{

double fractPart = d - Math.floor(d);// Fractional part of the given number

converted = converted + ".";// Adding the decimal point

int f = 0;// Count of the number of decimal places

// Converting the fractional part up to 10 decimal places

while(f<10 && fractPart!=0.0)

{

f++;

fractPart \*= base;

int prod = get\_int\_part(fractPart);

if(prod <= 9)

converted += prod + "";

else

converted += (char)((int)'A' + (prod % 10)) +"";

fractPart=fractPart-Math.floor(fractPart);

}

}

}

// Function to display the original and converted number

void display()

{

if(d - get\_int\_part(d) == 0)

System.out.print(get\_int\_part(d));

else

System.out.print(d);

System.out.println(" Converted to " + (choice=='B'?"Binary":choice=='O'?"Octal":"Hexadecimal") + " is " + converted);

}

// Main method to create objects and call functions accordingly

public static void main(String[] args)

{

Number nbr=new Number();// Creating object

// Calling functions accordingly

nbr.inputNum();

nbr.inputChoice();

nbr.convert();

nbr.display();

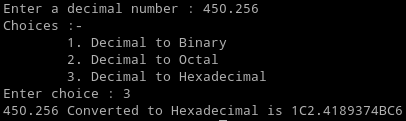
}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | double | d | To store the original decimal number entered by user |
| 2 | char | choice | To store the choice of base entered by user |
| 3 | String | converted | To store the converted number |
| 4 | - | Number() | Default constructor |
| 5 | void | inputNum() | To take input of a number |
| 6 | void | inputChoice() | To take input of the choice of base |
| 7 | int | get\_int\_part(double) | To get the integer part of a decimal number |
| 8 | void | convert() | To convert the decimal number to desired base |
| 9 | void | display() | To display the required things |
| 10 | void | main(String[]) | To create object and call functions accordingly |

Output)



***Program 14***

Question)

Given a time in numbers we can convert it into words. For example,

* 5 : 00 - five o’ clock
* 5 : 10 - ten minutes past five
* 5 : 15 - quarter past five
* 5 : 30 - half past five
* 5 : 40 - twenty minutes to six
* 5 : 45 - quarter to six
* 5 : 47 - thirteen minutes to six

Write a program which first inputs two integers, the first between 1 and 12 (both inclusive) and second between 0and 59 (both inclusive) and then prints out the time they represent, in words. Your program should follow the format of the examples above.

Algorithm)

* Start
* Take necessary inputs, and validate then and there
* For converting a number to words use switch case to use fall though and recursion to your advantage
* To convert the time to words, first convert the standard times, like half past some hour, some hour o' clock quarter past some-hour, quarter to some-hour
* For minutes less than 30, convert them as it is and print like some-minutes past some-hour
* for minutes more than 30, convert minutes - 30 to words and print like some-minutes to some-hour
* End

Code)

import java.util.\*;

class Time

{

// Declaring class variables

int hh/\* Store the hour \*/, mm/\* Store the minute \*/;

// Default constructor

Time()

{

// Initialising class variables to default values

hh = 0;

mm = 0;

}

// Function that takes integer input from user

int get\_int(String s)

{

try

{

// Taking input

Scanner sc=new Scanner(System.in);

System.out.print(s);

int x = sc.nextInt();

return x;// Returning the input

}

catch(Exception e)// Input mismatch

{

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

get\_int(s);// Calling the function again

}

return -1;// Return is required

}

// Function that takes input

void input()

{

// Getting hour

hh = get\_int("Enter the hour : ");

// Validating hour

while(hh > 12 || hh < 1)

{

System.out.println("Hour should be between 1 to 12 both inclusive");

hh = get\_int("Enter the hour : ");

}

// Getting minute

mm = get\_int("Enter the minute : ");

// Validating minute

while(mm > 59 || mm < 0)

{

System.out.println("Minute should be between 0 to 59 both inclusive");

mm = get\_int("Enter the minute : ");

}

}

// Function that converts a number to words

String convt(int i)

{

switch(i)

{

case 1: return "one";

case 2: return "two";

case 3: return "three";

case 4: return "four";

case 5: return "five";

case 6: return "six";

case 7: return "seven";

case 8: return "eight";

case 9: return "nine";

case 10: return "ten";

case 11: return "eleven";

case 12: return "twelve";

case 13: return "thirteen";

case 14:

case 16:

case 17:

case 18:

case 19: return convt(i - 10) + "teen";

case 20: return "twenty";

case 21:

case 22:

case 23:

case 24:

case 25:

case 26:

case 27:

case 28:

case 29: return "twenty-" + convt(i - 20);

}

return "";// Return is necessary

}

// Function to display the required things

void display()

{

if(mm==0)// Exactly an hour

System.out.println(convt(hh) + " o' clock");

else if(mm==30)// Half of an hour

System.out.println("half past " + convt(hh));

else if(mm==15)// Quarter of an hour

System.out.println("quarter past " + convt(hh));

else if(mm < 30)// Less than half but not a quarter

System.out.println(convt(mm) + " minutes past " + convt(hh));

else if(mm==45)// 3 Quarter of an hour

{

if(hh+1 == 13)// If the increment exceeds 12

System.out.println("quarter to " + convt(1));

else

System.out.println("quarter to " + convt(hh+1));

}

else// More than half but not 3 quarters

{

if(hh+1 == 13)// If the increment exceeds 12

System.out.println(convt(60-mm) + " minutes to " + convt(1));

else

System.out.println(convt(60-mm) + " minutes to " + convt(hh+1));

}

}

// Main method to create objects and call functions accordingly

public static void main(String[] args)

{

Time tme=new Time();// Creating object

// Calling functions accordingly

tme.input();

tme.display();

}

}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | int | hh | To store the hour |
| 2 | int | mm | To store the minutes |
| 3 | - | Time() | Default constructor |
| 4 | int | get\_int(String) | To get an integer input from user |
| 5 | void | input() | to take the required inputs |
| 6 | String | convt(int) | To convert a number to words |
| 7 | void | display() | To display the time in words |
| 8 | void | main(String[]) | To create object and call functions accordingly |

Output)



***Program 15***

Question)

A class Personal contains employee details and another class Retire calculates the employee’s Provident Fund and Gratuity. The details of the two classes are given below:

Class name : Personal

Instance variables:

* Name : stores the employee’s name
* Pan : stores the employee PAN
* basic\_pay : stores the employee basic salary ( in decimals )
* acc\_no : stores the employee bank account number

Member methods:

* Personal(….) : parameterized constructor to assign value to data members
* void display( ) : to display the employee details

Class name : Retire

Instance variables:

* yrs : stores the employee years of service
* pf : stores the employee provident fund amount (in decimals)
* grat : stores the employee gratuity amount (in decimals)

Member methods:

* Retire( ……. ) : parameterized constructor to assign value to data members of both the classes.
* void provident( ) : calculates the PF as (2% of the basic pay) \* years of service.
* void gratuity( ) : calculates the gratuity as 12 months’ salary, if the years of service is ≥ 10 years else the gratuity amount is nil.
* void display ( ) : Display the employee details along with the Provident Fund and gratuity amount.

Specify the class Personal giving details of the constructor and member functions void display( ). Using the concept of inheritance, specify the class Retire giving details of constructor, and the member functions void provident( ), void gratuity( ) and the void display( )

Algorithm)

* Start
* Create Personal class using the class description given
* Create Retire class extending Personal class using the class description
* for gratuity use ternary operator to check and then calculate the gratuity
* for pf, calculate according to the formula given
* Take inputs in main method and create object, call functions
* Display the required things
* End

Code)

import java.util.\*;

class Personal

{

// Declaring class variables

String Name/\* To store name of the employee \*/, Pan/\* To store the PAN number of the employee \*/;

long acc\_no;// To store the account number of the employee

double basic\_pay;// To store the Basic salary of the employee

// Parameterized constructor

Personal(String n, String p, long a, double b)

{

// Initialising the class variables

Name = n;

Pan = p;

acc\_no = a;

basic\_pay = b;

}

// Function to display the details

void display()

{

System.out.println("Employee Name : " + Name);

System.out.println("Employee PAN : " + Pan);

System.out.println("Employee Basic Salary : " + basic\_pay);

System.out.println("Employee Bank Account Number : " + acc\_no);

}

}

class Retire extends Personal

{

// Declaring class variables

int yrs;// To store the years of service

double pf/\* To store the provident fund \*/, grat/\* To store the gratuity \*/;

// Parameterised constructor

Retire(String name, String pan, long accNo, int years, double salary)

{

super(name, pan, accNo, salary);// Calling the constructor of the super class

// Initialising the class variables

yrs = years;

pf = 0.0;

grat = 0.0;

}

// Function to calculate provident fund

void provident()

{

pf = (0.02 \* basic\_pay) \* yrs;

}

// Function to calculate gratuity

void gratuity()

{

grat = yrs > 10 ? basic\_pay \* 12 : 0.0;

}

// Function to display the necessary things

void display()

{

String x = "";

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

x += "-";

System.out.println("-------------------------------" + x);

super.display();// Calling the display function of the super class

System.out.println("Provident Fund : " + pf);

System.out.println("Gratuity : " + grat);

System.out.println("-------------------------------" + x);

}

// Main method to take input, create object and call functions

public static void main(String[] args)

{

/\*

The try-catch blocks are so that input mis-match can be handled

and the while loops help int re-asking the user for input as long as he does not input a valid input

\*/

Scanner sc=new Scanner(System.in);

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

String n = sc.nextLine();

System.out.print("Enter Employee PAN number : ");

String p = sc.nextLine();

double b = 0.0;

while(true)

{

try

{

System.out.print("Enter Employee Basic Salary : ");

b = sc.nextDouble();

if(b > 0)

break;

else

System.out.println("Basic salary can't be negative");

}

catch(Exception e)

{

System.out.println("Please enter a number only");

}

}

long a = 0;

while(true)

{

try

{

System.out.print("Enter Employee Bank Account Number : ");

a = sc.nextLong();

if(a > 0)

break;

else

System.out.println("Bank Account number should be a positive number");

}

catch(Exception e)

{

System.out.println("Please enter a number only");

}

}

int y = 0;

while(true)

{

try

{

System.out.print("Enter Employee Years of Service : ");

y = sc.nextInt();

if(y > 0)

break;

else

System.out.println("Number of years should be more than Zero");

}

catch(Exception e)

{

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

}

}

Retire rtr=new Retire(n, p, a, y, b);// Creating object

// Calling functions accordingly

rtr.provident();

rtr.gratuity();

rtr.display();

}

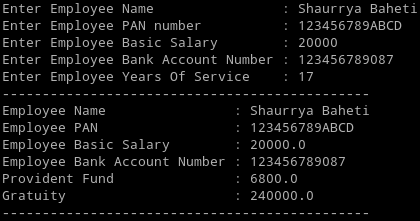
}

Variable Description Table)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | String | Name | stores the employee’s name |
| 2 | String | Pan | stores the employee PAN |
| 3 | long | acc\_no | stores the employee bank account number |
| 4 | double | basic\_pay | stores the employee basic salary (in decimals) |
| 5 | - | Personal(String, String, long, double) | parameterized constructor to assign value to data members |
| 6 | void | display() | to display the employee details |

|  |  |  |  |
| --- | --- | --- | --- |
| No. | DataType/ReturnType | Variable/Method | Description |
| 1 | int | yrs | stores the employee years of service |
| 2 | double | pf | stores the employee provident fund amount (in decimals) |
| 3 | double | grat | stores the employee gratuity amount (in decimals) |
| 4 | - | Retire(String, String, long, int, double) | parameterized constructor to assign value to data members of both the classes. |
| 5 | void | provident() | calculates the PF as (2% of the basic pay) \* years of service |
| 6 | void | gratuity() | calculates the gratuity as 12 months’ salary, if the years of service is ³ 10 years else the gratuity amount is nil. |
| 7 | void | display() | Display the employee details along with the Provident Fund and gratuity |
| 8 | void | main(String[]) | To take input, create object, and call the functions accordingly |

Output)



***Bibliography***

The book that helped me in completing the project is Computer Science with JAVA: A Textbook for Class XII by Sumita Arora.