|  |  |
| --- | --- |
| Method Name | calculateSum |
| Method Description | Calculate Sum |
| Argument | int n |
| Return Type | int-sum |
| Logic | Calculate the sum of first n natural numbers which are divisible by 3 or 5. |

**Exercise 1:** Create a class with a method which can calculate the sum of first n natural numbers

which are divisible by 3 or 5.

**CODE:**

**package** practice\_lab;

**import** java.util.Scanner;

**publicclass** Ex1 {

**int**i,sum=0;

**int** calculateSum(**int** n){

System.*out*.print("The numbers are :");

**for**(i=3;i<=n;i++){

**if**(i%3==0 || i%5==0){

System.*out*.print(i+",");

sum+=i;

}

}

**return**sum;

}

**publicstaticvoid** main(String[] args) {

Ex1 e=**new** Ex1();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the limit n :");

**int** n=s.nextInt();

System.*out*.println("\nThe sum is :"+e.calculateSum(n));

}

}

**OUTPUT:**

Enter the limit n :

9

The numbers are :3,5,6,9,

The sum is :23

**Exercise 2:** Create a class with a method to find the difference between the sum of the squares and the square of the sum of the first n natural numbers.

|  |  |
| --- | --- |
| Method Name | calculateDifference |
| Method Description | Calculate the difference |
| Argument | int n |
| Return Type | int - Sum |
| Logic | Find the difference between the sum of the squares of the first n natural numbers and the square of their sum.  For Example if n is 10,you have to find  (1^2+2^2+3^2+….9^2+10^2)-  (1+2+3+4+5…+9+10)^2 |

CODE:

**package** practice\_lab;

**import** java.util.\*;

//import java.math.\*;

**publicclass** Ex2 {

**int** calculateDifference(**int** n){

**int** i, squareOfSums=0, sumOfSquares=0;

**for**(i=1;i<=n;i++){

sumOfSquares+=Math.*pow*(i, 2);

squareOfSums+=i;

}

squareOfSums=(**int**)Math.*pow*(squareOfSums, 2);

System.*out*.println("sumOfSquares :"+sumOfSquares+"\nsquareOfSums :"+squareOfSums);

**return** sumOfSquares-squareOfSums;

}

**publicstaticvoid** main(String[] args) {

Ex2 e=**new** Ex2();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the limit n: ");

**int** n=s.nextInt();

System.*out*.println("The difference is :" + e.calculateDifference(n));

s.close();

}

}

OUTPUT

Enter the limit n:

5

sumOfSquares :55

squareOfSums :225

The difference is :-170

**Exercise 3:** Create a class containing a method to create the mirror image of a String. The method should return the two Strings separated with a pipe(|) symbol .

|  |  |
| --- | --- |
| Method Name | getImage |
| Method Description | Generate the mirror image of a String and add it to the existing string. |
| Argument | String |
| Return Type | String |
| Logic | Accepts One String  Find the mirror image of the String  Add the two Strings together separated by a pipe(|) symbol.  For Example  Input : EARTH  Output : EARTH|HTRAE  Hint: Use StringBuffer API (Ex: For this problem reverse method in Stringbuffer can be used)  Note: Learn the other APIs in StringBuffer |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex3 {

String getImage(String s){

StringBuffer image=**new** StringBuffer(s);

String mirrorImage=**new** StringBuffer().append(s).append(" | ").append(image.reverse()).toString();

**return** mirrorImage;

}

**publicstaticvoid** main(String[] args) {

Ex3 e=**new** Ex3();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the string :");

String st=s.nextLine();

System.*out*.println("The Mirror Image of the given String is :"+e.getImage(st));

s.close();

}

}

OUTPUT:

Enter the string :

hello world

The Mirror Image of the given String is :hello world | dlrow olleh

**Exercise 4:** Create a method to check if a number is an increasing number

|  |  |
| --- | --- |
| Method Name | checkNumber |
| Method Description | Check if a number is an increasing number |
| Argument | int number |
| Return Type | boolean |
| Logic | A number is said to be an increasing number if no digit is exceeded by the digit to its left.  For Example : 134468 is an increasing number |

CODE:

**package** practice\_lab;

**import** java.util.Scanner;

**publicclass** Ex4 {

Boolean checkNumber(**int** number){

Boolean b;

**int** i,flag=0;

String s=**new** StringBuffer().append(number).toString();

**for**(i=1;i<s.length();i++){

**if**(s.charAt(i-1)>s.charAt(i)){

flag=1;

**break**;

}

}

**if**(flag==1){

flag=0;

b=**false**;

}**else**{

b=**true**;

}

**return** b;

}

**publicstaticvoid** main(String[] args) {

Ex4 e=**new** Ex4();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the number:");

**int** number=s.nextInt();

**if**(e.checkNumber(number)){

System.*out*.println("It is an Increasing Number!");

}**else**{

System.*out*.println("Not an Increasing Number!");

}

s.close();

}

}

OUTPUT:

Enter the number:

7788

It is an Increasing Number!

Enter the number:

1293

Not an Increasing Number!

**Exercise 5:** Create a method to check if a number is a power of two or not

|  |  |
| --- | --- |
| Method Name | checkNumber |
| Method Description | Checks if the entered number is a power of two or not |
| Argument | int n |
| Return Type | boolean |
| Logic | Check if the input is a power of two.  Ex: 8 is a power of 2 |

CODE:

**package** practice\_lab;

**import**java.util.Scanner;

**publicclass** Ex5 {

**boolean** checkNumber(**int** n){

**int** i,flag=0;

**boolean** b;

**for**(i=0;i<n/2;i++){

**if**(n==(**int**)Math.*pow*(2,i)){

flag=1;

**break**;

}

}

**if**(flag==1){

flag=0;

b=**true**;

}**else**{

b=**false**;

}

**return** b;

}

**publicstaticvoid** main(String[] args) {

Ex5 e=**new** Ex5();

Scanner s=**new**Scanner(System.*in*);

System.*out*.println("Enter the number: ");

**int** n=s.nextInt();

**if**(e.checkNumber(n)){

System.*out*.println(n+" is a power of 2");

}**else**{

System.*out*.println(n+" is not a power of 2");

}

s.close();

}

}

OUTPUT:

Enter the number:

1000

1000 is not a power of 2

Enter the number:

1024

1024 is a power of 2

**Example 6:** A school offers medals to the students of tenth based on the following criteria

If(Marks>=90) : Gold

If(Marks between 80 and 90) : Silver

If(Marks between 70 and 80) : Bronze

Note: Marks between 80 and 90 means marks>=80 and marks<90

Write a function which accepts the marks of students as a Hashmap and return the details of the students eligible for the medals along with type of medal.

The input hashmap contains the student registration number as key and mark as value.

The output hashmap should contain the student registration number as key and the medal type as value.

|  |  |
| --- | --- |
| Method Name | getStudents |
| Method Description | Generate the list of students eligible for scholarship |
| Argument | Hashmap |
| Return Type | Hashmap |
| Logic | The method should return the details of the students eligible for the medals along with the medal type. |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex6 {

Map<Integer,String> getStudents(Map<Integer,Integer> inputMap){

**int** i,key,mark;

Map<Integer,String> outputMap=**new** HashMap<Integer,String>();

List<Integer> keys=**new** ArrayList<Integer>();

keys.addAll(inputMap.keySet());

**for**(i=0;i<keys.size();i++){

key=keys.get(i);

mark=inputMap.get(key);

**if**(mark>=90){

outputMap.put(key,"Gold");

}**else if**(mark>=80 && mark<90){

outputMap.put(key,"Silver");

}**else if**(mark>=70 && mark<80){

outputMap.put(key, "Bronze");

}

}

**return** outputMap;

}

**publicstaticvoid** main(String[] args) {

Ex6 e=**new** Ex6();

Map<Integer,Integer> inputMap=**new** HashMap<Integer,Integer>();

Map<Integer,String> outputMap=**new** HashMap<Integer,String>();

Scanner s=**new** Scanner(System.*in*);

**int** n,rno,mark;

System.*out*.println("Enter the number of Students:");

n=s.nextInt();

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

System.*out*.println("Enter the register number:");

rno=s.nextInt();

System.*out*.println("Enter the mark :");

mark=s.nextInt();

inputMap.put(rno, mark);

}

Set<Integer> key=inputMap.keySet();

**for**(**int** i : key){

System.*out*.println("Register Number: "+i+"\nMark: "+inputMap.get(i));

}

outputMap=e.getStudents(inputMap);

key=outputMap.keySet();

System.*out*.println("\nEligible Students");

**if**(outputMap.size()==0)

System.*out*.println("No students are Eligible");

**else**{

**for**(**int** i : key){

System.*out*.println("\nRegister Number: "+i+"\nScholarship: "+outputMap.get(i));

}

s.close();

}

}

}

OUTPUT:

Enter the number of Students:

4

Enter the register number:

1

Enter the mark :

90

Enter the register number:

2

Enter the mark :

80

Enter the register number:

3

Enter the mark :

70

Enter the register number:

4

Enter the mark :

60

Register Number: 1

Mark: 90

Register Number: 2

Mark: 80

Register Number: 3

Mark: 70

Register Number: 4

Mark: 60

Eligible Students

Register Number: 1

Scholarship: Gold

Register Number: 2

Scholarship: Silver

Register Number: 3

Scholarship: Bronze

**Example 7:** Create a method which accepts a String and replaces all the consonants in the String with the next alphabet.

|  |  |
| --- | --- |
| **Note**: Consonant refers to all alphabets excluding vowels Method Name | alterString |
| Method Description | Replace consonants |
| Argument | String |
| Return Type | String |
| Logic | Return the String replacing all the consonants with the next character.  For Example :JAVA should be changed as KAWA |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex7 {

String alterString(String oldString){

List<Character> vowels=**new** ArrayList<Character>();

vowels.add(' ');

vowels.add('a');vowels.add('e');vowels.add('i');vowels.add('o');vowels.add('u');

vowels.add('A');vowels.add('E');vowels.add('I');vowels.add('O');vowels.add('U');

**char**[] newString=oldString.toCharArray();

**for**(**int** i=0;i<newString.length;i++){

**if**(!vowels.contains(newString[i])){

newString[i]+=1;

}

}

StringBuffer sss=**new** StringBuffer();

String str=sss.append(newString).toString();

**return** str;

}

**publicstaticvoid** main(String[] args) {

Ex7 e=**new** Ex7();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the String :");

String str=s.nextLine();

System.*out*.println("Altered String is :"+e.alterString(str));

s.close();

}

}

OUTPUT:

Enter the String :

Hello World

Altered String is :Iemmo Xosme

**Example 8:** Create a method which accepts an array of integer elements and return the second smallest element in the array

|  |  |
| --- | --- |
| Method Name | getSecondSmallest |
| Method Description | Get the second smallest element in the array |
| Argument | int[] |
| Return Type | int |
| Logic | Sort the array and return the second smallest element in the array  Hint:  1. Convert to ArrayList  2. Use sort method in Collections class |

CODE:

**import** java.util.\*;

**publicclass** Ex8 {

**int** getSecondSmallest(**int**[] arr){

**int** i;

Set<Integer> arraySet=**new** TreeSet<Integer>();

Map<Integer,Integer> arrayMap=**new** TreeMap<Integer,Integer>();

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

arraySet.add(arr[i]);

}

Iterator<Integer> it=arraySet.iterator();

i=1;

**while**(it.hasNext()){

arrayMap.put(i,it.next());

i++;

}

**return** arrayMap.get(2);

}

**publicstaticvoid** main(String[] args) {

Ex8 e=**new** Ex8();

Scanner s=**new** Scanner(System.*in*);

**int** n,i;

System.*out*.println("Enter the size of the list:");

n=s.nextInt();

**int**[] array=**newint**[n];

System.*out*.println("Enter the Numbers :");

**for**(i=0;i<n;i++){

array[i]=s.nextInt();

}

System.*out*.println("The Second Largest Number is :"+e.getSecondSmallest(array));

s.close();

}

}

OUTPUT:

Enter the size of the list:

5

Enter the Numbers :

13

70

56

88

20

The Second Largest Number is :20

**Example 9:** Create a method which can perform the following operations on two String objects S1 and S2. The output of each operation should be added to an arraylist and the arraylist should be returned.(Assume S2 is of smaller size)

Examples for below statements are shown in the Logic part

1. Character in each alternate index of S1 should be replaced with S2

2. If S2 appears more than once in S1, replace the last occurrence of S2 in S1 with the reverse of S2, else return S1+S2

3. If S2 appears more than once in S1, delete the first occurrence of S2 in S1, else return S1

4. Divide S2 into two halves and add the first half to the beginning of the S1 and second half to the end of S1.

Note: If there are odd number of letters in S2, then add (n/2)+1 letters to the beginning and the remaining letters to the end. (n is the number of letters in S2)

5. If S1 contains characters that is in S2 change all such characters to \*

|  |  |
| --- | --- |
| Method Name | modifyStrings |
| Method Description | Perform the above mentioned actions on a String |
| Argument | String,String |
| Return Type | Arraylist |
| Logic | Do the above mentioned actions on the entered String.  For Example  S1=”JAVAJAVA”  S2=”VA’  1. **VA**A**VA**A**VA**A**VA**A (J replaced with VA, V replaced with VA etc.)  2. JAVAJAAV  3. JAJAVA  4. VJAVAJAVAA |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex9 {

List<String> modifyStrings(String s1, String s2){

**int** i,firstOccurance=0,flag=0;

StringBuffer s=**new** StringBuffer(s1);

List<String> modifiedStrings=**new** ArrayList<String>();

//1.

**for**(i=0;i<s.length();i=i+s2.length()+1){

s.replace(i, i+1, s2);

}

modifiedStrings.add(s.toString());

//2.

StringBuffer sbs1=**new** StringBuffer(s1);

StringBuffer sbs2=**new** StringBuffer(s2);

**if**(sbs1.substring(0,sbs1.lastIndexOf(s2)).lastIndexOf(s2)!=-1){

sbs1.replace(sbs1.lastIndexOf(s2), sbs1.lastIndexOf(s2)+s2.length(),sbs2.reverse().toString());

modifiedStrings.add(sbs1.toString());

}**else**{

modifiedStrings.add(s1.concat(s2));

}

//3.

StringBuffer sbs1a=**new** StringBuffer(s1);

StringBuffer temp=**new** StringBuffer(s1.substring(0, sbs1a.lastIndexOf(s2)));

**if**(temp.lastIndexOf(s2)!=-1){

flag=1;

**while**(temp.lastIndexOf(s2)!=-1){

firstOccurance=temp.lastIndexOf(s2);

temp.replace(0,temp.length(),temp.substring(0,temp.lastIndexOf(s2)));

}

}

**if**(flag==1){

flag=0;

sbs1a.delete(firstOccurance, firstOccurance+s2.length());

modifiedStrings.add(sbs1a.toString());

}

//4.

StringBuffer sbs1b=**new** StringBuffer(s1);

**if**(s2.length()%2==0){

sbs1b.replace(0, 0, s2.substring(0,s2.length()/2));

sbs1b.replace(sbs1b.length(), sbs1b.length(),s2.substring(s2.length()/2));

}**else**{

sbs1b.replace(0, 0, s2.substring(0,(s2.length()/2)+1));

sbs1b.replace(sbs1b.length(), sbs1b.length(),s2.substring((s2.length()/2)+1));

}

modifiedStrings.add(sbs1b.toString());

//5.

Set<Character> charset=**new** HashSet<Character>();

**char**[] ch=s2.toCharArray();

**for**(i=0;i<ch.length;i++){

charset.add(ch[i]);

}

**char**[] st=s1.toCharArray();

**for**(i=0;i<st.length;i++){

**if**(charset.contains(st[i])){

st[i]='\*';

}

}

modifiedStrings.add(**new** StringBuffer().append(st).toString());

**return** modifiedStrings;

}

**publicstaticvoid** main(String[] args) {

Ex9 e=**new** Ex9();

Scanner s=**new** Scanner(System.*in*);

List<String> modifiedStrings=**new** ArrayList<String>();

System.*out*.println("Enter the first String :");

String str1=s.next();

System.*out*.println("Enter the second String :");

String str2=s.next();

modifiedStrings=e.modifyStrings(str1, str2);

**for**(String str:modifiedStrings){

System.*out*.println(str);

}

s.close();

}

}

OUTPUT:

Enter the first String :

javajava

Enter the second String :

va

vaavaavaavaa

javajaav

jajava

vjavajavaa

j\*\*\*j\*\*\*

**Example 10:** Create a method that accepts a number and modifies it such that the each of the digit in the newly formed number is equal to the difference between two consecutive digits in the original number. The digit in the units place can be left as it is.

Note: Take the absolute value of the difference. Ex: 6-8 = 2

|  |  |
| --- | --- |
| Method Name | modifyNumber |
| Method Description | Accepts a number and modify it as per the requirement |
| Argument | int number1 |
| Return Type | int |
| Logic | Accept a number and modify it such that the each of the digit in the newly formed number is equal to the difference between two consecutive digits in the original number.  For example.  Input: 45862  Output:13242  **Algorithm:**   Convert number into String   Extract each char using charAt method   Convert char to int and find the difference   Create new StringBuffer object and keep adding the difference   Finally convert StringBuffer to int |

CODE: to be corrected

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex10 {

**int** modifyNumber(**int** number1){

**char**[] num=**new** StringBuffer().append(number1).toString().toCharArray();

StringBuffer modifiedNum=**new** StringBuffer();

**int** n1,n2,i;

**for**(i=1;i<num.length;i++){

n1=num[i];

n2=num[i-1];

modifiedNum.append(Math.*abs*(n1-n2));

}

modifiedNum.append(num[i-1]);

**return** Integer.*parseInt*(modifiedNum.toString());

}

**publicstaticvoid** main(String[] args) {

Ex10 e=**new** Ex10();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the number:");

**int** n=s.nextInt();

System.*out*.println("Modified Number is :"+e.modifyNumber(n));

s.close();

}

}

OUTPUT:

Enter the number:

845965

Modified Number is :414315

**Example 11:** Create a method which accepts the date of birth of person and date format and print the day (SUNDAY, MONDAY…) on which he was born.

Note: The output should be in upper case

|  |  |
| --- | --- |
| Method Name | getDayofWeek |
| Method Description | Finds the day of the week in which a person is born |
| Argument | String date, String dateFormat |
| Return Type | String – Day of week |
| Logic | Use Calendar API and switch case to get the day of the week  Ex: Input1 = 25/06/2012  Input2 = dd/MM/yyyy  Output= MONDAY |

package CollExercise;

import java.util.\*;

import java.text.\*;

public class Ex11 {

String getDayofWeek(String date, String dateFormat){

Date d=new Date();

SimpleDateFormat sdf=new SimpleDateFormat(dateFormat);

SimpleDateFormat sdf1=new SimpleDateFormat("EEEEE");

try{

d=sdf.parse(date);

}catch(Exception e){

e.printStackTrace();

}

return sdf1.format(d).toString().toUpperCase();

}

public static void main(String[] args) {

Ex11 e=new Ex11();

Scanner s=new Scanner(System.in);

System.out.println("Enter the date:");

String date=s.nextLine();

System.out.println("Enter the format:");

String dateFormat=s.nextLine();

System.out.println(e.getDayofWeek(date, dateFormat));

}

}

OUTPUT:

Enter the date:

20/02/1991

Enter the format:

dd/MM/yyyy

WEDNESDAY

**Example 12:** You are asked to create an application for registering the details of jobseeker. The requirement is:

Username should always end with **\_job** and there should be at least minimum of 8 characters to the left of **\_job**. Write a function to validate the same. Return true in case the validation is passed. In case of validation failure return false.

|  |  |
| --- | --- |
| Method Name | validateUserName |
| Method Description | Checks if the username is valid |
| Argument | String userName |
| Return Type | boolean |
| Logic | Checks if the username ends with \_job and contains at least 8 characters to the left of \_job. If valid return true. Else return false. |

CODE:

**package** CollExercise;

**import** java.util.\*;

**publicclass** Ex12 {

**boolean** validateUserName(String uname){

**boolean** b=**false**;

**if**(uname.lastIndexOf("\_iob")!=-1){

**if**(uname.substring(uname.lastIndexOf("\_iob")).compareTo("\_iob")==0){

**if**(uname.length()>=12){

b=**true**;

}

}

}

**return** b;

}

**publicstaticvoid** main(String[] args) {

Ex12 e=**new** Ex12();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the username:");

String uname=s.next();

**if**(e.validateUserName(uname)){

System.*out*.println("Valid username!!");

}**else**{

System.*out*.println("Not valid!!");

}

}

}

OUTPUT:

Enter the username:

Meyyammai\_iob

Valid username!!

Enter the username:

abhi\_1991

Not valid!!

Enter the username:

meyyammai

Not valid!!

**Example 13:** Create a method that can accept an array of String objects and sort in alphabetical order. The elements in the left half should be completely in uppercase and the elements in the right half should be completely in lower case. Return the resulting array.

|  |  |
| --- | --- |
| Note: If there are odd number of String objects, then (n/2)+1 elements should be in UPPPERCASE Method Name | getArrayList |
| Method Description | Converts the String array to ArrayList and sorts it |
| Argument | String []elements |
| Return Type | String [] modifiedArray |
| Logic | Load the elements in to an ArrayList ,sort it, convert the left half element to uppercase and right half elements to lower case .  Hint :  1. Use Collection  2. Use String API |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex13 {

String[] getArrayList(String[] elements){

**int** i;

String[] modifiedArray=**new** String[elements.length];

List<String> elementList=**new** ArrayList<String>();

**for**(i=0;i<elements.length;i++){

elementList.add(elements[i]);

}

String s1,temp;

Collections.*sort*(elementList);

**for**(i=0;i<elementList.size();i++){

s1=elementList.get(i);

**if**(s1.length()%2==0){

temp=s1.substring(0, s1.length()/2).toUpperCase().concat(s1.substring(s1.length()/2).toLowerCase());

}**else**{

temp=s1.substring(0, (s1.length()/2)+1).toUpperCase().concat(s1.substring((s1.length()/2)+1).toLowerCase());

}

modifiedArray[i]=temp;

}

**return** modifiedArray;

}

**publicstaticvoid** main(String[] args) {

Ex13 e=**new** Ex13();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the number of elements:");

**int** n=s.nextInt();

String[] elements=**new** String[n];

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

elements[i]=s.next();

}

String[] modifiedArray=e.getArrayList(elements);

**for**(String str: modifiedArray){

System.*out*.println(str);

}

s.close();

}

}

OUTPUT:

Enter the number of elements:

3

meyyammai

somasundaram

kumudini

KUMUdini

MEYYAmmai

SOMASUndaram

**Example 14:** Create a method which can remove a List from another List

|  |  |
| --- | --- |
| Method Name | removeElements |
| Method Description | Removes the elements in one list that is present in the second list also. |
| Argument | List<String> list1, List<String> list2; |
| Return Type | List- ArrayList contains the resulting List after the removal process. |
| Logic | Accept two List objects list1 and list2 and remove the elements from list1 that are present in list2. This should be done in single step process without using loop.  Hint: Use the List API which removes all the items in List1 which are contained in List2 |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex14 {

List<String> removeElements(List<String> list1, List<String> list2){

list1.removeAll(list2);

List<String> list=**new** ArrayList<String>();

list.addAll(list1);

**return** list;

}

**publicstaticvoid** main(String[] args) {

Ex14 e=**new** Ex14();

Scanner s=**new** Scanner(System.*in*);

**int** i,n1,n2;

List<String> list1=**new** ArrayList<String>();

List<String> list2=**new** ArrayList<String>();

System.*out*.println("Enter the number of elements for List 1:");

n1=s.nextInt();

System.*out*.println("Enter the elements of List 1 :");

s.nextLine();

**for**(i=0;i<n1;i++){

list1.add(s.nextLine());

}

System.*out*.println("Enter the number of elements for List 2:");

n2=s.nextInt();

System.*out*.println("Enter the elements of List 2 :");

s.nextLine();

**for**(i=0;i<n2;i++){

list2.add(s.nextLine());

}

List<String> list=**new** ArrayList<String>();

list=e.removeElements(list1, list2);

System.*out*.println(list);

s.close();

}

}

OUTPUT:

Enter the number of elements for List 1:

4

Enter the elements of List 1 :

ab

cd

ef

gh

Enter the number of elements for List 2:

2

Enter the elements of List 2 :

ab

ef

[cd, gh]

**Example 15:** Create a method which can remove all the elements from a list other than the list of elements specified.

|  |  |
| --- | --- |
| Method Name | removeElements |
| Method Description | Remove all the elements from a list other than the list of elements specified. |
| Argument | List<String> list1, List<String> list2; |
| Return Type | List- ArrayList contains the resulting List after the removal process. |
| Logic | Accept two List objects list1 and list2 and remove all the elements from list 1 other than the elements contained in list2.This should be done in single step process without using loop.  **Hint**: Use the List API method which can retain the elements available in the second list only |

CODE:

**package** practice\_lab;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

**publicclass** Ex15 {

List<String> removeElements(List<String> list1, List<String> list2){

list1.retainAll(list2);

List<String> list=**new** ArrayList<String>();

list.addAll(list1);

**return** list;

}

**publicstaticvoid** main(String[] args) {

Ex15 e=**new** Ex15();

Scanner s=**new** Scanner(System.*in*);

**int** i,n1,n2;

List<String> list1=**new** ArrayList<String>();

List<String> list2=**new** ArrayList<String>();

System.*out*.println("Enter the number of elements for List 1:");

n1=s.nextInt();

System.*out*.println("Enter the elements of List 1 :");

s.nextLine();

**for**(i=0;i<n1;i++){

list1.add(s.nextLine());

}

System.*out*.println("Enter the number of elements for List 2:");

n2=s.nextInt();

System.*out*.println("Enter the elements of List 2 :");

s.nextLine();

**for**(i=0;i<n2;i++){

list2.add(s.nextLine());

}

List<String> list=**new** ArrayList<String>();

list=e.removeElements(list1, list2);

System.*out*.println(list);

s.close();

}

}

OUTPUT:

Enter the number of elements for List 1:

4

Enter the elements of List 1 :

ab

cd

ef

gh

Enter the number of elements for List 2:

2

Enter the elements of List 2 :

ab

ef

[ab, ef]

**Example 16:** Create a method which accepts an array of numbers and returns the numbers and their squares in an HashMap

|  |  |
| --- | --- |
| Method Name | getSquares |
| Method Description | Accepts a list of numbers and return their squares |
| Argument | int[] |
| Return Type | Map |
| Logic | Iterate through the list, find the square of each number and add the elements to a map object with the number as the key and the square as the value. |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex16 {

Map<Integer, Integer> getSquares(**int**[] numbers){

**int** num,i;

Map<Integer, Integer> squares=**new** HashMap<Integer,Integer>();

**for**(i=0;i<numbers.length;i++){

num=numbers[i];

squares.put(num,(num\*num));

}

**return** squares;

}

**publicstaticvoid** main(String[] args) {

Ex16 e=**new** Ex16();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the number of elements :");

**int** n,i;

n=s.nextInt();

**int**[] numbers=**newint**[n];

System.*out*.println("Enter the elements:");

**for**(i=0;i<n;i++){

numbers[i]=s.nextInt();

}

Map<Integer, Integer> squares=**new** HashMap<Integer,Integer>();

squares=e.getSquares(numbers);

Set<Integer> keys=squares.keySet();

**for**(**int** key:keys){

System.*out*.println(key+"---"+squares.get(key));

}

s.close();

}

}

OUTPUT:

Enter the number of elements :

5

Enter the elements:

1

2

3

4

5

1---1

2---4

3---9

4---16

5---25

**Example 17:** Create a method which accepts the id and the age of people as a Map and decide if they are eligible for vote. A person is eligible for vote if his age is greater than 18. Add the IDs of all the eligible persons to list and return the list. (Assume date is in DD/MM/yyyy format)

|  |  |
| --- | --- |
| Method Name | votersList |
| Method Description | Generate the list of voters based on the ages of the people |
| Argument | Map |
| Return Type | List |
| Logic | Accept a map with ID as key and Date of Birth as value and check if the person is eligible to vote. A person is eligible for vote for if his age is greater than 18. If the person is eligible add his ID to the list.  Hint: Use Calendar API and SimpleDateFormat |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**import** java.text.\*;

**publicclass** Ex17 {

List<Integer> votersList(Map<Integer,String> hmap){

List<Integer> voterslist=**new** ArrayList<Integer>();

SimpleDateFormat sdf=**new** SimpleDateFormat("dd-MM-yyyy");

String age=**new** String();

Calendar c1=Calendar.*getInstance*();

Calendar c2=Calendar.*getInstance*();

Set<Integer> keys= hmap.keySet();

System.*out*.println("Today’s date : "+c2.getTime());

**for**(**int** key:keys){

age=hmap.get(key);

**try**{

c1.setTime(sdf.parse(age));

}

**catch**(Exception e){

e.printStackTrace();

}

**float** secs=(**float**) ((**float**)(c2.getTimeInMillis()-c1.getTimeInMillis())/1000);

**float** mins=(**float**)(secs/60);

**float** hours=(**float**)(mins/60);

**double** days=(**float**)(hours/24);

**float** years=(**float**)(days/365.30);

System.*out*.println("ID :"+key+"\nAge : "+years);

**if**(years>=18)

voterslist.add(key);

}

**return** voterslist;

}

**publicstaticvoid** main(String[] args) {

Ex17 e=**new** Ex17();

Scanner s=**new** Scanner(System.*in*);

Map<Integer,String> hmap=**new** HashMap<Integer,String>();

System.*out*.println("Enter the number of Entries :");

**int** n,i;

n=s.nextInt();

**for**(i=0;i<n;i++){

System.*out*.println("Enter the ID and DOB (dd-mm-yyyy) : ");

hmap.put(s.nextInt(), s.next());

}

System.*out*.println("Voter's List :"+e.votersList(hmap));

s.close();

}

}

OUTPUT:

Enter the number of Entries :

3

Enter the ID and DOB (dd-mm-yyyy) :

1

20-02-1991

Enter the ID and DOB (dd-mm-yyyy) :

2

25-12-1994

Enter the ID and DOB (dd-mm-yyyy) :

3

26-12-1994

Today’s date : Tue Dec 25 14:33:41 IST 2012

ID :1

Age : 21.843983

ID :2

Age : 18.000566

ID :3

Age : 17.99783

Voter's List :[1, 2]

**Example 18:** Create a method which accepts an integer array, reverse the numbers in the array and returns the resulting array in sorted order

|  |  |
| --- | --- |
| Method Name | getSorted |
| Method Description | Return the resulting array after reversing the numbers and sorting it |
| Argument | int [] |
| Return Type | int |
| Logic | Accept and integer array, reverse the numbers in the array, sort it and return the resulting array.  Hint  1. Convert the numbers to String to reverse it  2. Use Collection APIs to sort it  **Ex:** {12,23,96,45}  **Step 1:** Reverse numbers  {21,32,69,54}  **Step2:** Sort it  {21,32,54,69}  **Hint**: Use String to reverse number  To sort it, Convert array to ArrayList and use Collections.sort |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex18 {

**int**[] getSorted(**int**[] unsort){

**int** i, len=unsort.length;

**int**[] sorted=**newint**[len];

StringBuffer[] unsorts=**new** StringBuffer[len];

List<String> list=**new** ArrayList<String>();

**for**(i=0;i<len;i++){

unsorts[i]=**new** StringBuffer().append(unsort[i]);

unsorts[i].reverse();

list.add(unsorts[i].toString());

}

Collections.*sort*(list);

**for**(i=0;i<len;i++){

sorted[i]=Integer.*parseInt*(list.get(i));

}

**return** sorted;

}

**publicstaticvoid** main(String[] args) {

Ex18 e=**new** Ex18();

Scanner s=**new** Scanner(System.*in*);

**int** n;

System.*out*.println("Enter the number of Elements:");

n=s.nextInt();

**int**[] unsort=**newint**[n];

**int**[] sorted=**newint**[n];

System.*out*.println("Enter the elements :");

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

unsort[i]=s.nextInt();

}

sorted=e.getSorted(unsort);

System.*out*.println("Reversed and sorted order is :");

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

System.*out*.println(sorted[i]);

}

s.close();

}

}

OUTPUT:

Enter the number of Elements:

5

Enter the elements :

89

65

19

34

88

Reversed and sorted order is :

43

56

88

91

98

**Example 19:** Create a method which accepts an integer array and removes all the duplicates in the array. Return the resulting array in descending order

|  |  |
| --- | --- |
| Method Name | modifyArray |
| Method Description | Remove duplicates |
| Argument | int [] |
| Return Type | int [] |
| Logic | Remove the duplicate elements in the array and sort it in descending order  Hint:  1. Use Collection API (TreeSet) to remove duplicates and sort the result in ascending order  2. Create a new array, iterate through elements in TreeSet and add it in the reverse order |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex19 {

**int**[] modifyArray(**int**[] array){

**int** i,j,len=array.length;

Set<Integer> arraySet=**new** TreeSet<Integer>();

**for**(i=0;i<len;i++){

arraySet.add(array[i]);

}

List<Integer> listarray=**new** ArrayList<Integer>();

listarray.addAll(arraySet);

len=listarray.size();

**int**[] result=**newint**[len];

System.*out*.println(listarray);

**for**(i=len-1,j=0;i>=0;i--,j++){

result[j]=listarray.get(i);

}

**return** result;

}

**publicstaticvoid** main(String[] args) {

Ex19 e=**new** Ex19();

**int** i;

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the number of Elements:");

**int** n=s.nextInt();

**int**[] array=**newint**[n];

System.*out*.println("Enter the Elements:");

**for**(i=0;i<n;i++){

array[i]=s.nextInt();

}

**int**[] modifiedArray=e.modifyArray(array);

System.*out*.println("In descending order :");

**for**(i=0;i<modifiedArray.length;i++){

System.*out*.println(modifiedArray[i]);

}

s.close();

}

}

OUTPUT:

Enter the number of Elements:

5

Enter the Elements:

56

76

89

90

76

[56, 76, 89, 90]

In descending order :

90

89

76

56

**Example 20:** Create a method that accepts a character array and count the number of times each character is present in the array. Add how many times each character is present to a hash map with the character as key and the repetitions count as value

|  |  |
| --- | --- |
| Method Name | countCharacter |
| Method Description | Count the number of occurrence of each character in a Character array |
| Argument | char[] |
| Return Type | map |
| Logic | Count the number of times each character appears in the array. Add the details into a hash map with character as key and count as value.  Example:  {‘A’,’P’,’P’,’L’,’E’}  Output: Will be hashmap with the following contents{‘A’:1,’P’:2,’L’:1,’E’:1} |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex20 {

Map<Character,Integer> countCharacter(**char**[] carray){

**int** i,j,len=carray.length,count=0;

Map<Character, Integer> hmap=**new** HashMap<Character,Integer>();

**for**(i=0;i<len;i++){

count=1;

**if**(!hmap.containsKey(carray[i])){

**for**(j=i+1;j<len;j++){

**if**(carray[i]==carray[j]){

count++;

}

}

}**else**{

**continue**;

}

hmap.put(carray[i], count);

}

**return** hmap;

}

**publicstaticvoid** main(String[] args) {

Ex20 e=**new** Ex20();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the string :");

**char**[] carray=s.next().toCharArray();

Map<Character, Integer> hmap=**new** HashMap<Character,Integer>();

hmap=e.countCharacter(carray);

System.*out*.println(hmap);

s.close();

}

}

OUTPUT:

Enter the string :

madame

{d=1, e=1, a=2, m=2}

**Example 21:** A String contains a list of states and capitals. Write a method which can parse the string and return the states and capitals as map with state as key and capital as value.

The String is in the below format.

The state and capital is separated by a delimiter (del1). There will be multiple state-capital pairs and each state – capital pair is separated by another delimiter (del2).

Ex: Input will be **tamilnadu||chennai-karanataka||bengaluru.**

**Here, || will be provided as del1 and - will be provided as del2.**

|  |  |
| --- | --- |
| Method Name | getStates |
| Method Description | Accepts the states and capitals as a String and return a map |
| Argument | String data, char del1,char del l2 |
| Return Type | Map |
| Logic | Parse the string based on the delimiters and load it to a map with the state name as key and capital as value.  Hint: Use Stringtokenizer or split method in String class.  Try both the above ways to get familiarized with both APIs |

CODE:

**package** CollExercise;

**import** java.util.\*;

**publicclass** Ex21 {

//using string tokenizer

Map<String,String> getStates(String data, **char** del1, **char** del2){

Map<String,String> states=**new** HashMap<String,String>();

String del1s=**new** StringBuffer().append(del1).toString();

String del2s=**new** StringBuffer().append(del2).toString();

StringTokenizer st=**new** StringTokenizer(data,del2s);

**while** (st.hasMoreTokens()){

StringTokenizer st1=**new** StringTokenizer(st.nextToken(),del1s);

states.put(st1.nextToken(), st1.nextToken());

}

**return** states;

}

//using split--DONT GIVE | OR || AS DELIMITER .. IT WONT WORK !!

/\*Map<String,String> getStates(String data, char del1, char del2){

Map<String,String> states=new HashMap<String,String>();

String del1s=new StringBuffer().append(del1).toString();

String del2s=new StringBuffer().append(del2).toString();

for (String pair:data.split(del2s)){

String[] str=pair.split(del1s);

states.put(str[0], str[1]);

}

return states;

}\*/

**publicstaticvoid** main(String[] args) {

Ex21 e=**new** Ex21();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the delimiter 1 and delimiter 2:");

**char** del1;

**char** del2;

del1=s.next().charAt(0);

del2=s.next().charAt(0);

System.*out*.println("Enter the state and capital in the following format\n" +

"state||capital-state||capital-state||capital");

String data=s.nextLine();

data=s.nextLine();

Map<String,String> states=**new** HashMap<String,String>();

states=e.getStates(data, del1, del2);

System.*out*.println(states);

s.close();

}

}

OUTPUT:

Enter the delimiter 1 and delimiter 2:

|

-

Enter the state and capital in the following format

state||capital-state||capital-state||capital

tamil nadu|chennai-andra pradesh|hyderabad-karnataka|bengaluru

{andra pradesh=hyderabad, karnataka=bengaluru, tamil nadu=chennai}

**Example 22:** In a certain television game show, a couple is considered as a perfect couple if both the husband’s and wife’s name contains the same set of characters. Each couple is provided with an ID. Write a method which can accept a Hashmap with ID as key and the husband’s and wife’s name separated with “-” as value. The method should generate the list of perfect couples based on the above mentioned criteria and return their IDs as List object.

|  |  |
| --- | --- |
| Method Name | checkPerfectCouple |
| Method Description | Select the set of perfect couples |
| Argument | Map |
| Return Type | List |
| Logic | Accept the Map  Iterate through it  Separate the husband’s and wife’s names  If they contain the same characters, add the ID to the List object.  Ex: Assuming VIMAL-MALIV is the value, this is a perfect couple since both these names contains same characters (in different order). |

CODE:

**package** CollExercise;

**import** java.util.\*;

**publicclass** Ex22 {

List<Integer> checkPerfectCouple(Map<Integer,String> hmap){

List<Integer> perfectCouple=**new** ArrayList<Integer>();

String h=**new** String();

String w=**new** String();

**int** i,len,flag=0;

Set<Integer> ids=**new** HashSet<Integer>();

ids=hmap.keySet();

String name\_pair;

**for**(**int** id:ids){

name\_pair=hmap.get(id);

StringTokenizer st=**new** StringTokenizer(name\_pair,"-");

**while**(st.hasMoreTokens()){

h=st.nextToken();

w=st.nextToken();

}

**if**(h.length()==w.length()){

List<Character> husband=**new** ArrayList<Character>();

List<Character> wife=**new** ArrayList<Character>();

len=h.length();

**for**(i=0;i<len;i++)

husband.add(h.charAt(i));

**for**(i=0;i<len;i++)

wife.add(w.charAt(i));

Collections.*sort*(husband);

Collections.*sort*(wife);

**for**(i=0;i<len;i++){

**if**(wife.get(i)!=husband.get(i)){

flag=1;

**break**;

}

}

**if**(flag==1){

flag=0;

}**else**{

perfectCouple.add(id);

}

}**else**{

**continue**;

}

}

**return** perfectCouple;

}

**publicstaticvoid** main(String[] args) {

Ex22 e=**new** Ex22();

Scanner s=**new** Scanner(System.*in*);

Map<Integer,String> hmap=**new** HashMap<Integer, String>();

List<Integer> eligibleCouple=**new** ArrayList<Integer>();

System.*out*.println("Enter the number of couples:");

**int** n=s.nextInt();

**int** id;

String name\_pair;

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

System.*out*.println("Enter Id:");

id=s.nextInt();

System.*out*.println("Enter name in \"name1-name2\" format:");

name\_pair=s.next();

hmap.put(id, name\_pair);

}

eligibleCouple=e.checkPerfectCouple(hmap);

System.*out*.println("Perfect Couples :"+eligibleCouple);

}

}

Output:

Enter the number of couples:

4

Enter Id:

1

Enter name in "name1-name2" format:

meyy-yeem

Enter Id:

2

Enter name in "name1-name2" format:

meyy-yeym

Enter Id:

3

Enter name in "name1-name2" format:

abc-cba

Enter Id:

4

Enter name in "name1-name2" format:

acd-def

Perfect Couples :[2, 3]

**Example 23:** Create a method which can perform a particular String operation based on the user’s choice. The method should accept the String object and the user’s choice and return the output of the operation.

Options are

A: Add the String to itself

B: Replace alternate positions with \*

C: Remove duplicate characters in the String

|  |  |
| --- | --- |
| D: Change alternate characters to upper case Method Name | changeString |
| Method Description | Modify the string based on user choice |
| Argument | String string, char ch |
| Return Type | String |
| Logic | Perform the required operation based on the user choice and return the resulting string |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex23 {

String changeString(String string, **char** ch){

StringBuffer changedString=**new** StringBuffer(string);

**int** i,len=string.length();

**switch**(ch){

**case**'a':{

changedString.append(string);

**break**;

}

**case**'b':{

**for**(i=0;i<len;i=i+2)

changedString.replace(i, i+1, "\*");

**break**;

}

**case**'c':{

Set<Character> hashset=**new** HashSet<Character>();

**for**(i=0;i<len;i++)

hashset.add(string.charAt(i));

changedString.delete(0, len);

**for**(**char** c:hashset)

changedString.append(c);

**break**;

}

**case**'d':{

**for**(i=0;i<len;i=i+2)

changedString.replace(i, i+1,changedString.substring(i, i+1).toUpperCase().toString());

**break**;

}

**default**:{

changedString.replace(0, len, "Invalid Choice!");

**break**;

}

}

**return** changedString.toString();

}

**publicstaticvoid** main(String[] args) {

Ex23 e=**new** Ex23();

Scanner s=**new** Scanner(System.*in*);

String string;

**char** ch;

System.*out*.println("Enter the String:");

string=s.nextLine();

System.*out*.println("a) append the string to itself\nb) replace the alternative characters with \* \nc) remove duplicate characters\nd) change alternate character to uppercase\nEnter your choice:");

ch=s.next().charAt(0);

System.*out*.println(e.changeString(string, ch));

s.close();

}

}

OUTPUT:

Enter the String:

cognizant

a) append the string to itself

b) replace the alternative characters with \*

c) remove duplicate characters

d) change alternate character to uppercase

Enter your choice:

a

cognizantcognizant

Enter the String:

cognizant

a) append the string to itself

b) replace the alternative characters with \*

c) remove duplicate characters

d) change alternate character to uppercase

Enter your choice:

b

\*o\*n\*z\*n\*

Enter the String:

heellooo

a) append the string to itself

b) replace the alternative characters with \*

c) remove duplicate characters

d) change alternate character to uppercase

Enter your choice:

c

eolh

Enter the String:

cognizant

a) append the string to itself

b) replace the alternative characters with \*

c) remove duplicate characters

d) change alternate character to uppercase

Enter your choice:

d

CoGnIzAnT

Enter the String:

cts

a) append the string to itself

b) replace the alternative characters with \*

c) remove duplicate characters

d) change alternate character to uppercase

Enter your choice:

e

Invalid Choice!

**Example 24:** Create a method that accepts a String and checks if it is a positive string. A string is considered a positive string, if on moving from left to right **each** character in the String comes after the previous characters in the Alphabetical order.

For Example

ANT is a positive String (Since T comes after N and N comes after A)

APPLE is not positive since L comes before P in the alphabetical order.

|  |  |
| --- | --- |
| The method should return true if the entered string is positive Method Name | checkPositive |
| Method Description | Checks if a String is positive |
| Argument | String |
| Return Type | boolean |
| Logic | Check if a string is positive based on the above criteria and return true if positive.  Hint:  **Step 1:** Convert to Char array.  **Step 2:** Iterate through array, subtract 1st two characters (A-N). This will give the ASCII difference  **Step 3:** If result is negative, then return false and break. Else continue to next loop |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex24 {

Boolean checkPositive(String str){

Boolean b;

**int** i,flag=0;

String s=**new** StringBuffer().append(str).toString();

**for**(i=1;i<s.length();i++){

**if**(s.charAt(i-1)>s.charAt(i)){

flag=1;

**break**;

}

}

**if**(flag==1){

flag=0;

b=**false**;

}**else**{

b=**true**;

}

**return** b;

}

**publicstaticvoid** main(String[] args) {

Ex24 e=**new** Ex24();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the String: ");

String str=s.next();

**if**(e.checkPositive(str)){

System.*out*.println("It is a Positive String!");

}**else**{

System.*out*.println("Not a Positive String!!");

}

s.close();

}

}

OUTPUT:

Enter the String:

apple

Not a Positive String!!

Enter the String:

ANT

It is a Positive String!

**Example 25:** Create a method which accepts two Arraylist containing characters. Merge both arrays lists, sort the elements in the resulting list and return the resulting array.

|  |  |
| --- | --- |
| Method Name | mergeData |
| Method Description | Merge two arraylist , sort it and return the result as an integer array. |
| Argument | List, List |
| Return Type | char[] |
| Logic | Merge both arrays lists, sort the elements in the resulting list and return it as a char array. |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex25 {

**char**[] mergeData(List<Character> list1, List<Character> list2){

list1.addAll(list2);

Collections.*sort*(list1);

**char**[] carray=**newchar**[list1.size()];

**for**(**int** i=0;i<list1.size();i++){

carray[i]=list1.get(i);

}

**return** carray;

}

**publicstaticvoid** main(String[] args) {

Ex25 e=**new** Ex25();

Scanner s=**new** Scanner(System.*in*);

List<Character> list1=**new** ArrayList<Character>();

List<Character> list2=**new** ArrayList<Character>();

**int** i,n;

System.*out*.println("Enter the number of elements in list 1:");

n=s.nextInt();

System.*out*.println("Enter the elements of list 1:");

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

list1.add(s.next().charAt(0));

System.*out*.println("Enter the number of elements in list 2:");

n=s.nextInt();

System.*out*.println("Enter the elements of list 2:");

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

list2.add(s.next().charAt(0));

**char**[] carray=e.mergeData(list1, list2);

System.*out*.println(**new** StringBuffer().append(carray).toString());

s.close();

}

}

OUTPUT:

Enter the number of elements in list 1:

5

Enter the elements of list 1:

a

p

p

l

e

Enter the number of elements in list 2:

3

Enter the elements of list 2:

a

n

t

aaelnppt

**Example 26:** Create a method that searches for a particular String in a List. If found, the element should be replaced with a string having only half of the characters in the actual string

|  |  |
| --- | --- |
| Method Name | modifyElement |
| Method Description | Search for an element in the arraylist and modifies it. |
| Argument | List<String> arrayList , String element |
| Return Type | List |
| Logic | Accept an arraylist and search for an element in the list and replace with a string having only first half of the characters in the actual string.  For Example if a search was done for APPLE and if APPLE is found in the list, replace it with APP.  Return the modified list  Hint:  Iterate through list and find the index where the String is present.  Take the first half of the String and set it at that index in the arraylist. (Use set method) |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex26 {

List<String> modifyElement(List<String> arrayList, String element){

String str;

**for**(**int** i=0;i<arrayList.size();i++){

str=arrayList.get(i);

**if**(str.compareToIgnoreCase(element)==0){

arrayList.set(i, str.substring(0, str.length()/2));

}

}

**return** arrayList;

}

**publicstaticvoid** main(String[] args) {

Ex26 e=**new** Ex26();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the number of elements in the list:");

**int** n,i;

String element;

List<String> arrayList=**new** ArrayList<String>();

n=s.nextInt();

System.*out*.println("Enter the elements in the list:");

s.nextLine();

**for**(i=0;i<n;i++){

arrayList.add(s.nextLine());

}

System.*out*.println("Enter the string to be modified in the list:");

element=s.next();

System.*out*.println(e.modifyElement(arrayList, element));

s.close();

}

}

OUTPUT:

Enter the number of elements in the list:

5

Enter the elements in the list:

apple

banana

college

doctor

egg

Enter the string to be modified in the list:

college

[apple, banana, col, doctor, egg]

**Example 27:** Create a method to find the sum of the first n even numbers that are divisible by 3 and 8

|  |  |
| --- | --- |
| Method Name | findSum |
| Method Description | Find the sum of first n even numbers that are divisible by 3 and 8 |
| Argument | Int |
| Return Type | Int |
| Logic | Sum of the multiples of first n even numbers that are divisible by 3 and 8 |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex27 {

**int** findSum(**int** n){

**int** sum=0;

System.*out*.println("The numbers are :");

**for**(**int** i=2;i<=n;i=i+2){

**if**(i%3==0 || i%8==0){

System.*out*.println(i);

sum+=i;

}

}

**return** sum;

}

**publicstaticvoid** main(String[] args) {

Ex27 e=**new** Ex27();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the limit n :");

**int** n=s.nextInt();

System.*out*.println("The Sum is "+e.findSum(n));

s.close();

}

}

OUTPUT:

Enter the limit n :

24

The numbers are :

6

8

12

16

18

24

The Sum is 84

**Example 28:** Create a method to find the sum of the cubes of the digits of an n digit number

|  |  |
| --- | --- |
| Method Name | findSum |
| Method Description | Find the sum cubes of the digits of an n digit number |
| Argument | Int |
| Return Type | Int |
| Logic | Return the sum of cubes of the digits of an n digit number  Example  Input : 123  Output : 1^3+2^3+3^3= 1+8+27=36  Hint: Use %(mod) operator to separate each digit |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex28 {

**int** findSum (**int** n){

**int** sum=0,digit;

**while**(n>0){

digit=n%10;

sum+=Math.*pow*(digit,3);

n=n/10;

}

**return** sum;

}

**publicstaticvoid** main(String[] args) {

Ex28 e=**new** Ex28();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the number:");

**int** n=s.nextInt();

System.*out*.println("The sum of the cubes of digits is :"+e.findSum(n));s.close();

}

}

OUTPUT:

Enter the number:

123

The sum of the cubes of digits is :36

**Example 29:** Create a method which accepts a hash map and return the values of the map in sorted order as a List.

|  |  |
| --- | --- |
| Method Name | getValues |
| Method Description | Get the values of a map in sorted order |
| Argument | HashMap |
| Return Type | List |
| Logic | Return the values of a hash map in sorted order |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex29 {

List<String> getValues(Map<Integer,String> hmap){

Set<Integer> keys=**new** HashSet<Integer>();

List<String> values=**new** ArrayList<String>();

keys=hmap.keySet();

**for**(**int** key:keys)

values.add(hmap.get(key));

Collections.*sort*(values);

**return** values;

}

**publicstaticvoid** main(String[] args) {

Ex29 e=**new** Ex29();

Scanner s=**new** Scanner(System.*in*);

Map<Integer,String> hmap=**new** HashMap<Integer,String>();

System.*out*.println("Enter the number of elements :");

**int** n,i;

n=s.nextInt();

**for**(i=0;i<n;i++){

System.*out*.println("Enter the key and value:");

hmap.put(s.nextInt(), s.next());

}

System.*out*.println(e.getValues(hmap));

s.close();

}

}

OUTPUT:

Enter the number of elements :

3

Enter the key and value:

1

one

Enter the key and value:

2

two

Enter the key and value:

3

three

[one, three, two]

**Example 30:** A company requires each employee to maintain a secret code. The secret code needs to pass certain validation for getting accepted.

The validation rules are as given

1. The secret code should be six characters long

2. The first three characters should be cod (Use .startsWith method)

3. There should be at least one digit in the code (Use .isDigit)

4. The first character should always be an upper case letter(Use isUpperCase)

5. The code should contain only alphabets and digits.

6. The number of upper case letters should be greater than lower case letters.

Return true if the above validation is passed.

|  |  |
| --- | --- |
| Method Name | validateCode |
| Method Description | Validate the entered code as per the given validation rules |
| Argument | String code |
| Return Type | boolean |
| Logic | Validate the entered code  Hint: Use the String API methods to extract each character |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex30 {

**boolean** validateCode(String code){

**int** digit=0,upper=0,lower=0;

**if**(code.length()==6 && code.substring(0,3).compareToIgnoreCase("cod")==0 && Character.*isUpperCase*(code.charAt(0))){

**char**[] ch=code.toCharArray();

**for**(**int** i=0;i<code.length();i++){

**if**(Character.*isLetterOrDigit*(ch[i])){

**if**(Character.*isDigit*(ch[i]))

digit++;

**elseif**(Character.*isUpperCase*(ch[i]))

upper++;

**elseif**(Character.*isLowerCase*(ch[i]))

lower++;

}**elsereturnfalse**;

}

}**elsereturnfalse**;

**if**(digit>0 && upper>lower)

**returntrue**;

**elsereturnfalse**;

}

**publicstaticvoid** main(String[] args) {

Ex30 e=**new** Ex30();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the Code:");

String code=s.next();

**if**(e.validateCode(code))

System.*out*.println("Valid Code!!");

**else**

System.*out*.println("Invalid Code!!");

s.close();

}

}

OUTPUT:

Enter the Code:

CODEa3

Valid Code!!

Enter the Code:

codea3

Invalid Code!!

Enter the Code:

DODEa3

Invalid Code!!

Enter the Code:

codecode

Invalid Code!!

Enter the Code:

code12c

Invalid Code!!

Enter the Code:

cODEa9

Invalid Code!!

Enter the Code:

CodeA3

Invalid Code!!

**Example 31:** Write a method to find the sum of the factorials of the first n numbers in the Fibonacci series.

Fibonacci series is given by 0 1 1 2 3 5 8…… Factorial for a number m is given by factorial= m\*m-1\*m-2….1

|  |  |
| --- | --- |
| Method Name | sumOfFactorial |
| Method Description | Calculate sum |
| Argument | int n |
| Return Type | int |
| Logic | 1. Generate the first n elements in the Fibonacci series  2. Find the factorial of each element  3. Find the sum of the factorial |

CODE:

**package** CollExercise;

**import** java.util.\*;

**publicclass** Ex31 {

**int** sumOfFactorial(**int** n){

**int** i,a=0,b=1,fact,j,sumOfFactorial=0;

**int**[] fibo=**newint**[n];

fibo[0]=a;

fibo[1]=b;

System.*out*.println("The fibonacci Series :\n"+fibo[0]+"\n"+fibo[1]);

**for**(i=2;i<n;i++){

fibo[i]=a+b;

a=b;

b=fibo[i];

System.*out*.println(fibo[i]);

}

**for**(i=0;i<fibo.length;i++){

fact=1;

**for**(j=1;j<=fibo[i];j++){

fact\*=j;

}

System.*out*.println("Factorial of "+ fibo[i]+" is :"+ fact);

sumOfFactorial+=fact;

}

**return** sumOfFactorial;

}

**publicstaticvoid** main(String[] args) {

Ex31 e=**new** Ex31();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Etner the limit of fibonacci series :");

**int** n=s.nextInt();

System.*out*.println("The sum of factorial of fibonacci series is :"+e.sumOfFactorial(n));

}

}

OUTPUT:

Etner the limit of fibonacci series :

7

The fibonacci Series :

0

1

1

2

3

5

8

Factorial of 0 is :1

Factorial of 1 is :1

Factorial of 1 is :1

Factorial of 2 is :2

Factorial of 3 is :6

Factorial of 5 is :120

Factorial of 8 is :40320

The sum of factorial of fibonacci series is :40451

**Example 32:** A company transmits its String data over the network as encrypted data. The encryption logic is as shown given below.

For a String ad the logic is as given

aa+9=j

dd+9=m

So the encrypted word would be jm.

|  |  |
| --- | --- |
| If on addition of 9 results in a char greater than z (ASCII value 122) do the encryption in a cyclic manner starting from a. So if any character is ‘z’ it should be (z+9) which is equal to 127>122. In this case the character would be 9 character starting from ‘a’ which ‘i’ so for adz the encrypted value should be adi Method Name | encryptString |
| Method Description | Encrypt the entered string |
| Argument | String |
| Return Type | String |
| Logic | 1. Perform the arithmetic operation of char data.  2. For example  Assume  char a=’b’;  a++;  Now the value of a will be c. This is |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex32 {

String encryptString(String s){

**char** c;

**char**[] data=s.toCharArray();

**for**(**int** i=0;i<data.length;i++){

c=data[i];

**if**(Character.*isUpperCase*(c)){

c+=9;

**if**(c>90)

c-=26;

data[i]=c;

}**elseif**(Character.*isLowerCase*(c)){

c+=9;

**if**(c>122)

c-=26;

data[i]=c;

}

}

**returnnew** StringBuffer().append(data).toString();

}

**publicstaticvoid** main(String[] args) {

Ex32 e=**new** Ex32();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the data:");

String data=s.nextLine();

System.*out*.println("Encrypted Data : "+e.encryptString(data));

s.close();

}

}

OUTPUT:

Enter the data:

Cognizant

Encrypted Data : Lxpwrijwc

**Example 33:** A sales company keeps track of the product purchased and sold. The company needs to make sure that the sale date is always after the purchase date. Write a method to verify this

|  |  |
| --- | --- |
| Method Name | compareDates |
| Method Description | Comparing the purchase date and selling date |
| Argument | String purchaseDate, String sellingDate |
| Return Type | boolean |
| Logic | 1. Convert the string to Date objects  2. Return true if the selling date comes after the purchase date |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**import** java.text.\*;

**publicclass** Ex33 {

**boolean** compareDates(String pdate, String sdate){

SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");

Date purchaseDate=**new** Date();

Date sellingDate=**new** Date();

**try**{

purchaseDate=sdf.parse(pdate);

sellingDate=sdf.parse(sdate);

}

**catch**(Exception e){

e.printStackTrace();

}

**if**(sellingDate.compareTo(purchaseDate)>=0)

**returntrue**;

**elsereturnfalse**;

}

**publicstaticvoid** main(String[] args) {

Ex33 e=**new** Ex33();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter Purchase Date :");

String pdate=s.next();

System.*out*.println("Enter Selling Date :");

String sdate=s.next();

**if**(e.compareDates(pdate, sdate))

System.*out*.println("Yes ! Sales date falls after Purchase date!");

**else**

System.*out*.println("No! Sales date falls before Purchase date!");

s.close();

}

}

OUTPUT:

Enter Purchase Date :

20/02/2010

Enter Selling Date :

20/02/2010

Yes ! Sales date falls after Purchase date!

Enter Purchase Date :

20/02/2010

Enter Selling Date :

23/04/2012

Yes ! Sales date falls after Purchase date!

Enter Purchase Date :

20/03/2010

Enter Selling Date :

20/02/2010

No! Sales date falls before Purchase date!

**Example 34:** A company used to keep the record of the employees in two different branches separately. There are some employees who work in both the location. The company needs to keep track of the employee working in both the branches. Write a method to accept the two lists containing the names of the employees working in the two branches. The method should find out the names of employees present in both the list and return the names as a sorted array

|  |  |
| --- | --- |
| Method Name | getEmployees |
| Method Description | Get the names of employees working two different branches |
| Argument | List branch1, List branch2 |
| Return Type | String [] |
| Logic | Find the common names of the employees in both the lists |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex34 {

String[] getEmployees(List<String> list1, List<String> list2){

list1.retainAll(list2);

String[] employees=**new** String[list1.size()];

**for**(**int** i=0;i<list1.size();i++)

employees[i]=list1.get(i);

**return** employees;

}

**publicstaticvoid** main(String[] args) {

Ex34 e=**new** Ex34();

Scanner s=**new** Scanner(System.*in*);

**int** i,n;

List<String> list1=**new** ArrayList<String>();

List<String> list2=**new** ArrayList<String>();

System.*out*.println("Enter the number of employees in branch 1:");

n=s.nextInt();

System.*out*.println("Enter the Employees :");

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

list1.add(s.next());

System.*out*.println("Enter the number of employees in branch 2:");

n=s.nextInt();

System.*out*.println("Enter the Employees :");

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

list2.add(s.next());

String[] employees=e.getEmployees(list1, list2);

System.*out*.println("Employees working for both the branches :");

**for**(String str:employees)

System.*out*.println(str);

s.close();

}

}

OUTPUT:

Enter the number of employees in branch 1:

3

Enter the Employees :

apple

bat

cat

Enter the number of employees in branch 2:

2

Enter the Employees :

bat

cat

Employees working for both the branches :

bat

cat

**Example 35:** In a school there are some teachers who handle two subjects (Maths and English). When the feedback was taken their feedback was present in both Maths Feedback as well as English Feedback. Write a method to create a consolidated feedback for the teachers for English and Maths. For those taking both the subjects the highest feedback is taken. Write a method to accept two maps and return a Map object containing the feedbacks of all teachers in maths and English.

Note: HashMap contains

Key - Teacher’s name of type String

Value – Feedback of type Integer

|  |  |
| --- | --- |
| Method Name | getConsolidateFeedback |
| Method Description | Get the consolidated feedback |
| Argument | Map englishFeedback, Map mathsFeedBack |
| Return Type | Map consolidatedFeedBack |
| Logic | Hint :  1. Extract the keys of both the maps  2. For the teachers handling both the subjects find their maximum feedback and add to the output map.  3. Add the feed of other teachers to the output map. |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex35 {

Map<String,Integer> getConsolidatedFeedback(Map<String,Integer> eng, Map<String,Integer> math){

Map<String,Integer> consFB=**new** HashMap<String, Integer>();

Set<String> keys=eng.keySet();

**for**(String key:keys){

**if**(math.containsKey(key)){

consFB.put(key, Math.*max*(eng.get(key),math.get(key)));

math.remove(key);

}**else**{

consFB.put(key, eng.get(key));

}

}

keys=math.keySet();

**for**(String key:keys)

consFB.put(key, math.get(key));

**return** consFB;

}

**publicstaticvoid** main(String[] args) {

Ex35 e=**new** Ex35();

Scanner s=**new** Scanner(System.*in*);

Map<String,Integer> eng=**new** HashMap<String,Integer>();

Map<String,Integer> math=**new** HashMap<String,Integer>();

**int** n,i;

System.*out*.println("Enter the number of Feedbacks for English :");

n=s.nextInt();

System.*out*.println("Enter the Teacher name and Feedback between 1 to 10: ");

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

eng.put(s.next(), s.nextInt());

System.*out*.println("Enter the number of Feedbacks for Maths :");

n=s.nextInt();

System.*out*.println("Enter the Teacher name and Feedback between 1 to 10: ");

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

math.put(s.next(), s.nextInt());

System.*out*.println(e.getConsolidatedFeedback(eng, math));

s.close();

}

}

OUTPUT:

Enter the number of Feedbacks for English :

4

Enter the Teacher name and Feedback between 1 to 10:

aaa

5

bbb

6

ccc

7

ddd

8

Enter the number of Feedbacks for Maths :

3

Enter the Teacher name and Feedback between 1 to 10:

aaa

10

bbb

3

eee

9

{aaa=10, ddd=8, ccc=7, bbb=6, eee=9}

**Example 36:** Write a method which can find the sum of the first n prime numbers. Prime numbers are numbers which have only 1 and the number itself as its factors. 2 is the only even prime number. 1 is neither prime nor composite.

|  |  |
| --- | --- |
| Ex: the 1st 5 prime numbers are 2,3,5,7,11 and sum is 28 Method Name | getPrimeSum |
| Method Description | Get the sum of the first n prime numbers |
| Argument | int n |
| Return Type | Int |
| Logic | **Hint :**  1. Use for loop to iterate over numbers from 2 to n, say loop variable i.  2. Use an inner loop with loop variable j which loops from to 2 to i/2. If for any j the remainder on dividing i by j is zero, the number is non-prime. If it is prime add the number to the sum. |

CODE :

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex36 {

**int** getPrimeSum(**int** n){

**int** count=0, sum=0,number=2,i,flag=0;

**while**(count<=n){

**for**(i=2;i<=number/2;i++){

**if**(number%i==0){

flag=1;

**break**;

}

}

**if**(flag==1){

flag=0;

}**else**{

System.*out*.println("Prime number :"+number);

sum+=number;

count++;

}

number++;

}

**return** sum;

}

**publicstaticvoid** main(String[] args) {

Ex36 e=**new** Ex36();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the limit :");

**int** n=s.nextInt();

System.*out*.println("The sum of first "+n+" prime numbers is :"+e.getPrimeSum(n));

s.close();

}

}

OUTPUT:

Enter the limit :

5

Prime number :2

Prime number :3

Prime number :5

Prime number :7

Prime number :11

Prime number :13

The sum of first 5 prime numbers is :41

**Example 37:** Write a method which accepts a String and moves all the lower case ‘a’ to the beginning of the String.

|  |  |
| --- | --- |
| Method Name | rearrangeCharacters |
| Method Description | Move the all the lower case ‘a’ to the beginning of a String |
| Argument | String |
| Return Type | String |
| Logic | Hint :  1. Convert the string to a character array  2. Create a Stringbuffer object  3. Create a variable(**count**) to store the number of ‘a’ present  4. Iterate over the character array and if the character is ‘a’ increment **count** for ‘a’ else add the character to the StringBuffer object.  5. Finally insert the **count** number of ‘a’ to the beginning of the StringBuffer object |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex37 {

String rearrangeCharacters(String string){

**int** i;

StringBuffer aarray=**new** StringBuffer();

StringBuffer rearranged=**new** StringBuffer();

**char** ch;

**for**(i=0;i<string.length();i++){

ch=string.charAt(i);

**if**(Character.*isLowerCase*(ch) && ch=='a')

aarray.append(ch);

**else**

rearranged.append(ch);

}

rearranged.replace(0, 0, aarray.toString());

**return** rearranged.toString();

}

**publicstaticvoid** main(String[] args) {

Ex37 e=**new** Ex37();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the String:");

String string=s.next();

System.*out*.println("Rearranged String is :"+e.rearrangeCharacters(string));

s.close();

}

}

OUTPUT:

Enter the String:

HarAppa

Rearranged String is :aaHrApp

**Example 38:** Write a method which can find the factors of a number. Factor is an integer which evenly divides a number without leaving a remainder. Return the factors as an arraylist object.

For Example 1, 2 and 4 are the factor of 4

|  |  |
| --- | --- |
| Method Name | getFators |
| Method Description | Get the factors of a number n |
| Argument | int n |
| Return Type | List |
| Logic | Hint :  1. Create a loop starting from 1 to n with loop variable say i.  2. Check if for any i , dividing n by i gives zero as remainder. Then i is a factor of n .  3. Add i to the list object |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex38 {

List<Integer> getFactors(**int** n){

List<Integer> factors=**new** ArrayList<Integer>();

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

**if**(n%i==0)

factors.add(i);

**return** factors;

}

**publicstaticvoid** main(String[] args) {

Ex38 e=**new** Ex38();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the number:");

**int** n=s.nextInt();

System.*out*.println("The factors of "+n+" are "+e.getFactors(n));

s.close();

}

}

OUTPUT:

Enter the number:

45

The factors of 45 are [1, 3, 5, 9, 15, 45]

**Example 39:** Write a method which can accept an integer and return the binary, hexadecimal and octal equivalents of the number in a String array

|  |  |
| --- | --- |
| Method Name | getFormats |
| Method Description | Gets the binary,hexadecimal and octal formats of a number |
| Argument | int |
| Return Type | String |
| Logic | Hint :  1. Use Integer wrapper class methods |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex39 {

String[] getFormats(**int** n){

String[] formats=**new** String[3];

formats[0]=Integer.*toBinaryString*(n);

formats[1]=Integer.*toHexString*(n);

formats[2]=Integer.*toOctalString*(n);

**return** formats;

}

**publicstaticvoid** main(String[] args) {

Ex39 e=**new** Ex39();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the number:");

**int** n=s.nextInt();

String[] formats=e.getFormats(n);

System.*out*.println("Binary :"+formats[0]+"\nHexaDecimal:"+formats[1]+"\nOctal:"+formats[2]);

s.close();

}

}

OUTPUT:

Enter the number:

15

Binary :1111

HexaDecimal:f

Octal:17

**Example 40:** Write a method which accepts a double number and finds the sum of the digits to the left and right of the decimal point. It should return the sum as String in the following format

Left side sum:Right side sum

For example

Input :120.520

Output: 3:7

|  |  |
| --- | --- |
| Method Name | getSum |
| Method Description | Get the sum of digits on either sides of the decimal points in a double number |
| Argument | double |
| Return Type | String |
| Logic | Hint :  1. Convert the double number to aString  2. Separate the String to two parts based on the decimal point.  3. Find the sum of digits on each part |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex40 {

String getSum(**double** n){

String number=**new** StringBuffer().append(n).toString();

StringTokenizer st=**new** StringTokenizer(number,".");

String left=st.nextToken();

String right=st.nextToken();

**int** i,lsum=0,rsum=0;

**for**(i=0;i<right.length();i++){

rsum+=Integer.*parseInt*(**new** StringBuffer().append(right.charAt(i)).toString());

}

**for**(i=0;i<left.length();i++){

lsum+=Integer.*parseInt*(**new** StringBuffer().append(left.charAt(i)).toString());

}

String sum=**new** StringBuffer().append(lsum).append(":").append(rsum).toString();

**return** sum;

}

**publicstaticvoid** main(String[] args) {

Ex40 e=**new** Ex40();

Scanner s=**new** Scanner(System.*in*);

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

**double** n=s.nextDouble();

System.*out*.println("The resultant format is :"+e.getSum(n));

s.close();

}

}

OUTPUT:

Enter a decimal number :

456.987

The resultant format is :15:24

**Example 41:** Write a method to validate the age of a person. The person age is considered valid if it is above 21 years. Accept the date of birth of the person as String in date-month-year(Ex: 23-05-2012) format and return true if the age is greater than 21.

|  |  |
| --- | --- |
| Method Name | validateAge |
| Method Description | Validates if the person’s age is above 21 |
| Argument | String |
| Return Type | Boolean |
| Logic | Hint :Use Calendar, Date and DateFormat APIs |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**import** java.text.\*;

**publicclass** Ex41 {

**boolean** validateAge(String age){

SimpleDateFormat sdf=**new** SimpleDateFormat("dd-MM-yyyy");

//Date d2=new Date();

Calendar c1=Calendar.*getInstance*();

**try**{

c1.setTime(sdf.parse(age));

}

**catch**(Exception e){

e.printStackTrace();

}

Calendar c2=Calendar.*getInstance*();

**float** secs=(**float**) ((**float**)(c2.getTimeInMillis()-c1.getTimeInMillis())/1000);

**float** mins=(**float**)(secs/60);

**float** hours=(**float**)(mins/60);

**double** days=(**float**)(hours/24);

**float** years=(**float**)(days/365.30);

//System.out.println((c2.getTimeInMillis()-c1.getTimeInMillis())/(1000\*60\*60));

System.*out*.println("Birth Date : "+c1.getTime()+"\nToday's Date : "+c2.getTime()+"\nAge : "+years);

**if**(years>=21)

**returntrue**;

**else**

**returnfalse**;

}

**publicstaticvoid** main(String[] args) {

Ex41 e=**new** Ex41();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the DOB dd-mm-yyyy format :");

String dob=s.next();

**if**(e.validateAge(dob))

System.*out*.println("Valid Age !!");

**else**

System.*out*.println("Not Valid !!");

s.close();

}

}

**OUTPUT:**

Enter the DOB dd-mm-yyyy format :

20-02-1991

Birth Date :Wed Feb 20 00:00:00 IST 1991

Today's Date :Tue Dec 25 14:03:05 IST 2012

Age :21.843924

Valid Age !!

Enter the DOB dd-mm-yyyy format :

26-12-1991

Birth Date : Thu Dec 26 00:00:00 IST 1991

Today's Date : Tue Dec 25 14:05:26 IST 2012

Age : 20.998047

Not Valid !!

**Example 42:** Write a method which can return the current date in any of the following date formats based on the user choice

Choice 1 : Month-date-year(Ex: 05-26-2012)

Choice 2: Date-month-year(Ex: 26-MAY-12)

Choice 3: Date/month/year(Ex: 26/05/2012)

Choice 4: Month/date/year(Ex: 05/26/2012)

Choice 5: Return the current year

Choice 6: Return the current month as (Ex: APR)

Choice7: Return the date 10 days after the sysdate

Choice 8: Return 10 days prior to sysdate

|  |  |
| --- | --- |
| Method Name | getDate |
| Method Description | Format the entered date in the specified formats |
| Argument | int |
| Return Type | String |
| Logic | Use switch case(For choice) and DateFormat API |

CODE:

package practice\_lab;

import java.util.\*;

import java.text.\*;

public class Ex42 {

String getDate(int n){

Date d=new Date();

Calendar c=Calendar.getInstance();

String formatted=new String();

switch(n){

case 1:{

SimpleDateFormat sdf=new SimpleDateFormat("MM-dd-yyyy");

formatted=sdf.format(d);

break;

}

case 2:{

// Calendar c=Calendar.getInstance();

SimpleDateFormat sdf=new SimpleDateFormat("MMM");

SimpleDateFormat sdf1=new SimpleDateFormat("yy");

formatted=c.get(Calendar.DATE)+"-"+sdf.format(d).toUpperCase()+"-"+sdf1.format(d);

break;

}

case 3:{

SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");

formatted=sdf.format(d);

break;

}

case 4:{

SimpleDateFormat sdf=new SimpleDateFormat("MM/dd/yyyy");

formatted=sdf.format(d);

break;

}

case 5:{

SimpleDateFormat sdf=new SimpleDateFormat("yyyy");

formatted=sdf.format(d);

//formatted=new StringBuffer().append(c.get(Calendar.YEAR)).toString();

break;

}

case 6:{

SimpleDateFormat sdf=new SimpleDateFormat("MMM");

formatted=sdf.format(d).toUpperCase();

break;

}

case 7:{

Calendar c1=Calendar.getInstance();

c1.add(Calendar.DATE, 10);

formatted=c1.getTime().toString();

}

case 8:{

Calendar c2=Calendar.getInstance();

c2.add(Calendar.DATE, -10);

formatted=c2.getTime().toString();

}

}

return formatted;

}

public static void main(String[] args) {

Ex42 e=new Ex42();

Scanner s=new Scanner(System.in);

System.out.println("Choice 1 : Month-date-year(Ex: 05-26-2012)\nChoice 2: Date-month-year(Ex: 26-MAY-12)"+

"\nChoice 3: Date/month/year(Ex: 26/05/2012)\nChoice 4: Month/date/year(Ex: 05/26/2012)\nChoice 5: Return the current year"+

"\nChoice 6: Return the current month as (Ex: APR)\nChoice 7: Return the date 10 days after the sysdate\n"+

"Choice 8: Return 10 days prior to sysdate\nEnter the choice :");

int choice=s.nextInt();

System.out.println("Specified Format :"+e.getDate(choice));

s.close();

}

}

OUTPUT:

Choice 1 : Month-date-year(Ex: 05-26-2012)

Choice 2: Date-month-year(Ex: 26-MAY-12)

Choice 3: Date/month/year(Ex: 26/05/2012)

Choice 4: Month/date/year(Ex: 05/26/2012)

Choice 5: Return the current year

Choice 6: Return the current month as (Ex: APR)

Choice 7: Return the date 10 days after the sysdate

Choice 8: Return 10 days prior to sysdate

Enter the choice :

1

Specified Format :12-24-2012

Enter the choice :

2

Specified Format :24-DEC-12

Enter the choice :

3

Specified Format :24/12/2012

Enter the choice :

4

Specified Format :12/24/2012

Enter the choice :

5

Specified Format :2012

Enter the choice :

6

Specified Format :DEC

Enter the choice :

7

Specified Format :Thu Jan 03 00:29:04 IST 2013

Enter the choice :

8

Specified Format :Fri Dec 14 00:29:34 IST 2012

**Example 43:**

Consider two Hashmaps .First one containing the product name and product category code as key and value respectively. Second HashMap contains the product name and the units sold. Write a java function which accepts the two hash maps and return the names of products in each category which is having the highest number of units sold.

For example

Input1 :{“lux”:”soap”,”colgate”:”paste”, ”pears”:”soap”,”sony”:”electronics”,”samsung”:”electronics”}

Input 2:{“lux”:1000,”colgate”:500,”pears”:2000,”sony”:100,” samsung”,600}

Output: {“pears”,”colgate”,”samsung”}

|  |  |
| --- | --- |
| Method Name | getMaxSold |
| Method Description | Find the product maximum sold in each category |
| Argument | HashMap<String,String> productDetails, HashMap<String,Integer> salesDetails |
| Return Type | List |
| Logic | Algorithm:  1. Get product names from the productDetails map using the getValues method(Say categoryName which will be of data type Collection.  2. Remove the duplicate category names by converting categoryName to a Set object which can be done using the constructor of HashSet which accepts Collection object as argument  3. Obtain an iterator object(**iterator1**) to iterate over the category names and iterate over the set  4. Get first element in the set (**category**).  5. Declare two variables to maxSaleCount and maxSaleProduct with initial values 0 and null.  6. Inside the iteration create another iterator (**iterator2**) to iterate over the keys of the salesDetails map.  7. Iterate iterator2 and get the first element(**product**)  8. Check if the value for the key product in the productDetails map is equal to category. If found equal compare the sales of the product with the maxCount value and if the sales>maxCount set maxCount as sales and maxSaleProduct as **product.**  9. Completing the **iterator2** once will give the product which was sold maximum in a particular category  10. Completion of **iterator1** gives the names of the products which was sold maximum in each categories. |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex43 {

List<String> getMaxSold(Map<String,String> productDetails, Map<String,Integer> salesDetails){

List<String> products=**new** ArrayList<String>();

**int** maxSale=0;

String maxPdt=**new** String();

Set<String> keys=productDetails.keySet();

Set<String> category=**new** HashSet<String>();

**for**(String key:keys)

category.add(productDetails.get(key));

**for**(String categ:category){

**for**(String key:keys){

**if**(productDetails.get(key).compareToIgnoreCase(categ)==0 && salesDetails.get(key)>maxSale){

maxSale=salesDetails.get(key);

maxPdt=key;

}

}

maxSale=0;

products.add(maxPdt);

}

**return** products;

}

**publicstaticvoid** main(String[] args) {

Ex43 e=**new** Ex43();

Scanner s=**new** Scanner(System.*in*);

Map<String,String> productDetails=**new** HashMap<String,String>();

Map<String, Integer> salesDetails=**new** HashMap<String, Integer>();

**int** n,i;

System.*out*.println("Enter the number of entries in productDetails :");

n=s.nextInt();

String name,category;

**int** sales;

**for**(i=0;i<n;i++){

System.*out*.println("Enter the Product Name, Category, Number of Items Sold :");

s.nextLine();

name=s.nextLine();

category=s.nextLine();

sales=s.nextInt();

productDetails.put(name, category);

salesDetails.put(name, sales);

}

System.*out*.println("Maximum sold product in each Category :"+e.getMaxSold(productDetails, salesDetails));

s.close();

}

}

OUTPUT:

Enter the number of entries in productDetails :

5

Enter the Product Name, Category, Number of Items Sold :

persona

soap

5000

Enter the Product Name, Category, Number of Items Sold :

samsung

electronics

8000

Enter the Product Name, Category, Number of Items Sold :

pears

soap

4000

Enter the Product Name, Category, Number of Items Sold :

nokia

electronics

4000

Enter the Product Name, Category, Number of Items Sold :

herbal essence

shampoo

6000

Maximum sold product in each Category :[persona, samsung, herbal essence]

**Example 44:** Write a method which accepts a number and return it in words.

For Example 123 One Two Three Method

|  |  |
| --- | --- |
| Name | getNumber |
| Method Description | Get the number in words |
| Argument | int |
| Return Type | String |
| Logic | Use mod(%) operator, StringBuffer and switch case |

CODE:

**package** practice\_lab;

**import** java.util.Scanner;

**public class** Ex44 {

String getNumber(**int** n){

**int** digit;

StringBuffer string=**new** StringBuffer();

**while**(n>0){

digit=n%10;

**switch**(digit){

**case** 0:

string.replace(0,0,"zero ");

**break**;

**case** 1:

string.replace(0,0,"one ");

**break**;

**case** 2:

string.replace(0,0,"two ");

**break**;

**case** 3:

string.replace(0,0,"three ");

**break**;

**case** 4:

string.replace(0,0,"four ");

**break**;

**case** 5:

string.replace(0,0,"five ");

**break**;

**case** 6:

string.replace(0,0,"six ");

**break**;

**case** 7:

string.replace(0,0,"seven ");

**break**;

**case** 8:

string.replace(0,0,"eight ");

**break**;

**case** 9:

string.replace(0,0,"nine ");

**break**;

}

n=n/10;

}

**return** string.toString();

}

**publicstaticvoid** main(String[] args) {

Ex44 e=**new** Ex44();

Scanner s=**new** Scanner(System.*in*);

System.*out*.println("Enter the number:");

**int** n=s.nextInt();

System.*out*.println("Number in Words :"+e.getNumber(n));

s.close();

}

}

OUTPUT:

Enter the number:

1991

Number in Words :one nine nine one

**Example 45:** Consider two Hashmaps .First one containing the product name and product category code as key and value respectively. Second HashMap contains the product name and price. Write a java function which accepts the two hash maps , price hike rate and the product category and updates the prices of the product in the entered category by the hike rate

For example

Input1 :{“lux”:”soap”,”colgate”:”paste”, ”pears”:”soap”,”sony”:”electronics”,”samsung”:”electronics”}

Input 2:{“lux”:1000,”colgate”:500,”pears”:2000,”sony”:100,” samsung”,600}

Input 3:10

Input4: “electronics”

Output1: :{“lux”:1000,”colgate”:500,”pears”:2000,”sony”:110,” samsung”,660}

|  |  |
| --- | --- |
| Method Name | updatePrices |
| Method Description | Get the number in words |
| Argument | HashMap<String,String> productDetails, HashMap<String,Integer> salesDetails, int rate, String category |
| Return Type | String |
| Logic | Hint 1:  1. Iterate over the productDetails Map and get the names of the products in the entered category. The product names can be added to a Set  2. Iterate over the set whose elements will be present as key in the salesDetails tables. For each productEntry in the set update the values of the salesDetails map with rate%. |

CODE:

**package** practice\_lab;

**import** java.util.\*;

**publicclass** Ex45 {

Map<String,Integer> updatePrice(Map<String,String> pdDetails, Map<String,Integer> slDetails, **int** rate, String category){

Map<String,Integer> updatedPrice=**new** HashMap<String,Integer>();

Set<String> keys=pdDetails.keySet();

**int** hike=0;

**for**(String key: keys){

**if**(pdDetails.get(key).compareToIgnoreCase(category)==0){

hike=(slDetails.get(key)\*rate)/100;

}

updatedPrice.put(key, (slDetails.get(key)+hike));

hike=0;

}

**return** updatedPrice;

}

**publicstaticvoid** main(String[] args) {

Ex45 e=**new** Ex45();

Scanner s=**new** Scanner(System.*in*);

Map<String,String> productDetails=**new** HashMap<String,String>();

Map<String, Integer> salesDetails=**new** HashMap<String, Integer>();

**int** n,i;

System.*out*.println("Enter the number of entries in productDetails :");

n=s.nextInt();

String name,category;

**int** sales;

**for**(i=0;i<n;i++){

System.*out*.println("Enter the Product Name, Category, Number of Items Sold :");

s.nextLine();

name=s.nextLine();

category=s.nextLine();

sales=s.nextInt();

productDetails.put(name, category);

salesDetails.put(name, sales);

}

System.*out*.println("Enter the hike rate %:");

**int** rate=s.nextInt();

System.*out*.println("Enter the category:");

s.nextLine();

category=s.nextLine();

System.*out*.println("Updated Price List :"+e.updatePrice(productDetails, salesDetails,rate,category));

s.close();

}

}

OUTPUT:

Enter the number of entries in productDetails :

5

Enter the Product Name, Category, Number of Items Sold :

persona

soap

300

Enter the Product Name, Category, Number of Items Sold :

samsung

mobile

10000

Enter the Product Name, Category, Number of Items Sold :

pears

soap

100

Enter the Product Name, Category, Number of Items Sold :

nokia

mobile

8000

Enter the Product Name, Category, Number of Items Sold :

herbal essence

shampoo

500

Enter the hike rate %:

10

Enter the category:

mobile

Updated Price List :{nokia=8800, herbal essence=500, pears=100, samsung=11000, persona=300}