**ANAGARAM**

**package** swappingAndReversing;

**import** java.util.Arrays;

**public** **class** anagram {

**public** **static** **void** main(String[] args) {

//1st have to convert lowercase

// we have to convert both strings as character array4

//then both character array need to sort

// compare both character arrays using Arrays.equals()

String str1="belOW";

String str2="ELbow";

str1=str1.toLowerCase();

str2=str2.toLowerCase();

**if** (str1.length()==str2.length()) {

**char**[] ch1=str1.toCharArray();

**char**[] ch2=str2.toCharArray();

Arrays.*sort*(ch1);

Arrays.*sort*(ch2);

**boolean** result=Arrays.*equals*(ch1, ch2);

**if**(result) {

System.***out***.println("it is a anagram.......");

}**else** {

System.***out***.println("its not a anagram......");

}

} **else** {

System.***out***.println("it's not anagram........");

}

}

}

FACTORIAL

**public** **static** **void** main(String[] args) {

**int** num=6;

**int** fact=1;

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

fact=fact\*i;

System.***out***.println(fact);

}

System.***out***.println("==============================");

System.***out***.println(fact);

}

}

Fibanoic series

**public** **class** fibanoicSeries {

**public** **static** **void** main(String[] args) {

**int** i=0;

**int** j=1;

**int** k=1;

**int** count=10;

**for** (**int** l=1; l<=count; l++) {

System.***out***.println(i);

i=j;

j=k;

k=i+j;

}

}

}

Number palandrom

**public** **static** **void** main(String[] args) {

**int** num=1001;//It is the number variable to be checked for palindrome

**int** remainder;

**int** reversednumber=0;

**int** reversedValue=num;

**while**(num>0){

remainder=num%10; //getting remainder

reversednumber=(reversednumber\*10)+remainder;

num=num/10;

}

**if**(reversedValue==reversednumber)

System.***out***.println("palindrome number ");

**else**

System.***out***.println("not palindrome");

}

}

String palandrom

**public** **static** **void** main(String[] args) {

String str="NAVAN";

String rev="";

**for** (**int** i =str.length()-1; i >=0 ; i--) {

rev=rev+str.charAt(i);

}

**if** (str.equals(rev)) {

System.***out***.println("it is a palindrome..");

}**else** {

System.***out***.println("it is not a palindrome..");

}

System.***out***.println(rev);

}

}

Print 100 numbers without loop

**public** **class** printNumbersWithOutForLoop {

**private** **static** **void** printnum(**int** i) {

System.***out***.println(i);

**if**(i<100) {

*printnum*(i+1);

}

}

**public** **static** **void** main(String[] args) {

*printnum*(1);

}

}

Reverse

String str=”123456789”;

String reverse=””;

**for**(**int** i=str.length()-1; i>=0 ; i--) {

reverse=reverse+str.charAt(i);

System.***out***.println(reverse);

}

System.***out***.println(reverse);

}

}

String word reverse

**public** **static** **void** main(String[] args) {

String val="Welocome to NAVEEN IT";

String revData = Arrays.*asList*(val.split(" "))

.stream()

.map(x->**new** StringBuilder(x).reverse())

.collect(Collectors.*joining*(" "));

System.***out***.println(revData);

}

}

String logicals

**public** **static** **void** main(String[] args) {

List<String>list=Arrays.*asList*("sai","srinivas","vamsi","sai kiran","naveen","naveen");

**//find the count of elements in the list**

**long** elementscount=list.stream().count();

System.***out***.println(elementscount);

System.***out***.println("========**elementscount end**==========");

**long** particularElementCount=list.stream().filter(x->x.equals("naveen")).count();

System.***out***.println(particularElementCount);

System.***out***.println("========**particular element countend**======");

String name="saiRam";

**long** characterOccuranceCount=name.chars().filter(x->x=='s').count();

System.***out***.println(characterOccuranceCount);

System.***out***.println("==**character occurance count end=========");**

**//removing duplicates**

list.stream().distinct().forEach(System.***out***::println);

System.***out***.println("===**Distinct values end**==========");

**//print duplicate values**

Set<String>set=**new** HashSet<String>();

list.stream().filter(x>!set.add(x)).forEach(System.***out***::println);

System.***out***.println("=====print duplicate values end=====");

String str="Welcome to Teja it";

**long** charCount=str.chars().filter(x->x!=' ').count();

System.***out***.println(charCount);

System.***out***.println("=======**char count end**=========");

}

}

Vowels and consoling

**public** **static** **void** main(String[] args) {

String name="THis is NAVEEN it";

String input=name.toLowerCase();

List<Character>vowels=Arrays.*asList*('a','e','i','o','u');

**long** vowelsCount=input.chars().filter(ch->vowels.contains((**char**)ch)).count();

**long** consonantCount=input.chars().filter(ch->!vowels.contains((**char**)ch)).filter(x->x!=' ').count();

System.***out***.println(vowelsCount);

System.***out***.println(consonantCount);

}

}

Swapping

**public** **static** **void** main(String[] args) {

**int** i=20;

**int** j=40;

**int** k=i+j;

j=k-j;

i=k-i;

System.***out***.println(i+" "+j);

}

}

Funtctional interface

package swappingAndReversing;

import java.util.function.Consumer;

import java.util.function.Function;

import java.util.function.Predicate;

import java.util.function.Supplier;

public class FUnctionalInterfaces {

public static void main(String[] args) {

**//without lamda expression for consumer**

Consumer<String>cu=new Consumer<String>() {

@Override

public void accept(String t) {

System.out.println("name is.........."+t);

}

};

cu.accept("Naveen");//consumer having accept method , it take input doesnot return any return type.

//==========================================================================//

**//with lamda expression..for consumer**

Consumer<String>cu1= t->System.out.println("name is.........."+t);

cu1.accept("kumar");

//==================================================================//

**//without lamda expression for SUPPLIER**

Supplier<String>su=new Supplier<String>() {

@Override

public String get() {

return "naveen rayapudi";

}

};

System.out.println(su.get());

//===============================================================//

**//with lamda expression SUPPLIER**

Supplier<String>su1=()->"Naveen Kumar Rayapudi";

System.out.println(su1.get());

//=======================================================================//

//without lamda expression predicate

Predicate<String>pr=new Predicate<String>() {

@Override

public boolean test(String t) {

Boolean val=t.equals("NAVEEN");

return val;

}

};

Boolean data=pr.test("naveen");

System.out.println(data);

//====================================================================//

**//with lamda expression PREDICATE**

Predicate<String>pr1=t->t.equals("naveen");

Boolean data1=pr1.test("naveen");

System.out.println(data1);

//==================================================================//

**//without lamda expression FUNCTION**

Function<String,Double>f=new Function<String, Double>() {

@Override

public Double apply(String t) {

return 20000.45;

}

};

Double cost=f.apply("i phone cost is..");

System.out.println(cost);

//======================================================================//

**//with lamda expression FUNCTION**

Function<String,Double>f1=t->40000.45;

Double cost1=f1.apply("i phone cost is..");

System.out.println(cost1);

}

}

Enum

**package** featuresInterface;

**public** **enum** enumWeek {

***sunday***,***monday***,***tuesday***,***thursday***,***friday***,***saturday***;

**public** **static** **void** main(String[] args) {

System.***out***.println(***friday***);

}

}

Stream numbers

package featuresInterface;

import java.util.Arrays;import java.util.Comparator;

import java.util.List;

import java.util.Optional;

import java.util.OptionalDouble;

import java.util.stream.Collectors;

import java.util.stream.IntStream;

public class StreamNumberExamples {

public static void main(String[] args) {

// t->{body}

//Sequential stream

IntStream.range(1, 20).forEach(t->System.out.println("no.."+t));

IntStream.range(20, 40).parallel().forEach(s->System.out.println("pno:"+s));

//Parallel stream

IntStream.range(1,20).filter(t->t%2==0).forEach(t->System.out.println(t));

IntStream.range(1,20).filter(t->t%2!=0).forEach(t->System.out.println(t));

List<Integer>list=Arrays.asList(2,3,5,12,45,12,36,96,45,12,1,4);

list.stream()

.filter(t->t%2==0)//filter the data based on condition,filter having PREDICATE

.forEach(System.out::println);//iterate the elements and print

//className::MethodName=method Reference

//Integer::value of

System.out.println("=======================================================");

List<Integer>li=Arrays.asList(2,4,6,3,5,1,10,12,18,8,9,7,10,11,13,12,15);

li.stream()

.filter(t->t%2!=0)

.map(t->t\*5) //to apply some logic and internally having FUNCTION Method

//.distinct()//it will remove the duplicate data,or give unique data

//.sorted(Comparator.reverseOrder())//to sort the data

//.skip(1)//it will skip n number elements will display

//.limit(4)//it will show n number elements will display

.forEach(System.out::println);

System.out.println("==NAVEEN=====");

li.stream().forEach(System.out::println);

System.out.println("===================================");

List<Integer>distList=list.stream().distinct().collect(Collectors.toList());

System.out.println("=====================================");

distList.stream().filter(x->x%2==0).forEach(System.out::println);

System.out.println("=========AGGRIGATE FUNCTIONS=============");

Optional<Integer>max=list.stream().max(Integer::compare);

System.out.println("max values is........"+max);

Long count=list.stream().count();

System.out.println("count is .."+count

OptionalDouble avg=list.stream().mapToInt(x->x+5).average();

System.out.println("avg is.........."+avg);

int sum=list.stream().mapToInt(x->x\*3).sum();

System.out.println("sum is......."+sum);

Optional<Integer>findany=list.stream().findAny();

System.out.println("find any ....."+findany);

Integer findFirst=list.stream().findFirst().get();

System.out.println("find first ....."+findFirst);

Boolean anyMatch=list.stream().anyMatch(x->x==20);

System.out.println("all match....."+anyMatch);

Boolean nonMatch= list.stream().noneMatch(x->x==14);

System.out.println("non match is....."+nonMatch);

}

}

STREAM EXAMPLE

**public** **static** **void** main(String[] args) {

List<String>li=Arrays.*asList*("naveen R","sai kiran","sivaji","kesava","srinivas","surendra");

li.stream()

.filter(x->x.startsWith("s"))

//.map(x->x.concat(" from teja IT"))

.map(x->x.toLowerCase())

.distinct()

.sorted(Comparator.*reverseOrder*())

.skip(1)

.limit(2)

.forEach(System.***out***::println);

System.***out***.println("=======================");

li.stream().forEach(System.***out***::println);

System.***out***.println("=======================");

List<String>distList=li.stream().distinct().collect(Collectors.*toList*());

System.***out***.println("=========================");

Optional<String>findany=li.stream().findAny();

System.***out***.println("find any ....."+findany);

System.***out***.println("==========================");

String findFirst=li.stream().findFirst().get();

System.***out***.println("find first ....."+findFirst);

System.***out***.println("============================");

Boolean anyMatch=li.stream().anyMatch(x->x.equals("s"));

System.***out***.println("all match....."+anyMatch);

System.***out***.println("=============================");

Boolean nonMatch= li.stream().noneMatch(x->x.equalsIgnoreCase("siva"));

System.***out***.println("non match is....."+nonMatch);

}