1. Create a simple thread which counts the number of words in an input text file & print the word count

In Main Thread.

Note: Testing his Threading skills

public class ThreadCount

{

public static void main(String[] args)

{

final int count;

String fileName="This is Text File";

FileNames fn = new FileNames();

fn.setFileNames();

count = fn.count();

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

{

fileName = fn.getFileNames(i);

if(i != (count))

{

Runnable r = new Threads(fileName, false);

Thread t = new Thread(r);

t.start();

}

else

{

Runnable r = new Threads(fileName, true);

Thread t = new Thread(r);

t.start();

}}}}

2. Write a utility method which accepts any Animal type like Dog, Cat, and Lion etc. as an argument & print the characteristic of it.

Note: Testing his Generics skills

3. Write a program which prompts for any number as input & print its table till 10.

Input: If input is 8

Output: 1x8=8....10x8=80

Note: To test his I/O skills

public class Table {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter Number");

String value = sc.nextLine();

int n=Integer.parseInt(value);

for(int i=1;i<=10;i++) {

System.out.println( n+ "X "+ i +" = "+n\*i);

}

}

}

4. Write a utility method which accepts day of a week as input [1, 2, 3, 4, 5, 6, and 7] & print corresponding Day as String

Input: 2

Output: Monday

Note: Check if he/she is using Enum for type safety in arguments

public class EnumWeek {

private enum Week {

SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY;

static Week get(Integer i) {

return values()[i % 7];

}

}

public static void main(String[] args) {

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

Scanner input = new Scanner(System.in);

int userInput = input.nextInt();

Week week = Week.get(userInput);

switch (week) {

case MONDAY:

System.out.println("That's Monday");

break;

case TUESDAY:

System.out.println("That's Tuesday");

break;

case WEDNESDAY:

System.out.println("That's Wednesday");

break;

case THURSDAY:

System.out.println("That's Thursday");

break;

case FRIDAY:

System.out.println("That's Friday");

break;

case SATURDAY:

System.out.println("That's Saturday");

break;

case SUNDAY:

System.out.println("That's Sunday");

break;

default:

break;

}

}

}

5. Write a simple program to Copy paste a binary file (Zip, Image file)

Note: Check if he/she is having enough understanding on File readers / byte streams in IO

public class CopyFiles {

public static void main(String[] args) {

String inputFile = "C:\\Users\\Santosh\\Desktop\\pshychotheraphy\\depression.JPG";

String outputFile = "C:\\Users\\Santosh\\Desktop\\pshychotheraphy\\i\\depress.JPG";

try (

InputStream inputStream = new FileInputStream(inputFile);

OutputStream outputStream = new FileOutputStream(outputFile);

) {

int byteRead;

while ((byteRead = inputStream.read()) != -1) {

outputStream.write(byteRead);

}

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

6. Write a program to clone a Java Object.

Note: To test his understanding on Shallow/Deep copy, Marker Interface

public class cloneCon {

private int[] data;

public cloneCon(int[] values) {

data = new int[values.length];

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

data[i] = values[i];

}

}

public void showData() {

System.out.println(Arrays.toString(data));

}

public static void main(String[] args) {

int[] vals = {3, 7, 9};

cloneCon e = new cloneCon(vals);

e.showData();

vals[0] = 13;

e.showData(); }}

7. Convert below Runnable using Lambda function

Runnable r = new Runnable(){

@Override

public void run(){

System.out.println("Hello world one!");

}

};

Runnable r = () -> System.out.println("hello world");

8. Write a Lambda function to filter the list of Person objects based on Age (Age >80 ) & print their names.

9. Write a program to reverse a string with least complexity O(n), O(n/2)

public class reverseStr

{

void stringReverse()

{

String str = "UseCase";

int length = str.length();

StringBuffer revString = new StringBuffer();

for (int i = length - 1;

i >= 0; i--)

{

revString.append(str.charAt(i));

}

System.out.println(revString);

}

public static void main(String []args)

{

reverseStr s= new reverseStr();

s.stringReverse();

}}

10. Map of <Employee, Address> should replace the duplicate entry based on employee id.

Ex: If 2 employee objects with same employee id in put in the Map, older one should be replaced.

Note: Test his knowledge in hashCode() & equals()

11.Given a string of parenthesis ‘(‘ , ‘[’ , ‘<’ , ‘)’ , ‘]’ and ‘>’ write a function which returns true if there are matching pairs and false if there are not. A

matching pair means, there should be a closing parenthesis for an opening one, in correct order.

ex.

Input => [<()>[](<>)]

Return true for this use case.

Input => (<<>)

Return false for this use case as these is a missing closing tag. Given a string of parenthesis ‘(‘ , ‘[’ , ‘<’ , ‘)’ , ‘]’ and ‘>’ write a function which returns true if there are matching pairs and false if there are not. A

matching pair means, there should be a closing parenthesis for an opening one, in correct order.

ex.

Input => [<()>[](<>)]

Return true for this use case.

Input => (<<>)

Return false for this use case as these is a missing closing tag. Given a string of parenthesis ‘(‘ , ‘[’ , ‘<’ , ‘)’ , ‘]’ and ‘>’ write a function which returns true if there are matching pairs and false if there are not. A

matching pair means, there should be a closing parenthesis for an opening one, in correct order.

ex.

Input => [<()>[](<>)]

Return true for this use case.

Input => (<<>)

Return false for this use case as these is a missing closing tag. Given a string of parenthesis ‘(‘ , ‘[’ , ‘<’ , ‘)’ , ‘]’ and ‘>’ write a function which returns true if there are matching pairs and false if there are not. A

matching pair means, there should be a closing parenthesis for an opening one, in correct order.

ex.

Input => [<()>[](<>)]

Return true for this use case.

Input => (<<>)

Return false for this use case as these is a missing closing tag.

import java.util.\*;

public class BalancedBrackets {

static boolean areBracketsBalanced(String expr)

{

Deque<Character> stack

= new ArrayDeque<Character>();

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

{

char x = expr.charAt(i);

if (x == '(' || x == '[' || x == '{')

{

stack.push(x);

continue;

}

if (stack.isEmpty())

return false;

char check;

switch (x) {

case ')':

check = stack.pop();

if (check == '{' || check == '[')

return false;

break;

case '}':

check = stack.pop();

if (check == '(' || check == '[')

return false;

break;

case ']':

check = stack.pop();

if (check == '(' || check == '{')

return false;

break;

}

}

return (stack.isEmpty());

}

public static void main(String[] args)

{

String expr = "([{}])";

if (areBracketsBalanced(expr))

System.out.println("Balanced ");

else

System.out.println("Not Balanced ");

}

}

12. 2-3 yrs exp

Find the count of each characters in a string / count of fruits in a basket

public class OccuranceOfCharInBasket {

public static void main(String[] args) {

String Basket = "Apple Banana Lemon Apple Grape";

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

String arr[] = Basket.split("");

for (String word : arr) {

Integer count = map.get(word);

if (count == null) {

map.put(word, 1);

} else {

map.put(word, count + 1);

}

}

System.out.println(map);

}

}

13. Algorithm for water tank overflow device. When Motor is ON, keep checking if water tank has reached full level. If yes, stop the Motor.

14. Algorithm to implement FLAMES game

public class Flames

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

StringBuffer s1=new StringBuffer(sc.next());

StringBuffer s2=new StringBuffer(sc.next());

char c,d,e='\0';

for(int i=0;i<s1.length();i++)

{

c=s1.charAt(i);

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

{

d=s2.charAt(j);

if(c==d)

{

s1.deleteCharAt(i);

s2.deleteCharAt(j);

i--;

j--;

break;

}

}

}

s1=s1.append(s2);

int k=6,l=0;

String s3="FLAMES";

while(k>1)

{

for(int m=1;m<s1.length();m++)

{

if(s3.charAt(l)!='X')

{

l++;

if(l==6)

l=0;

}

else

{

l++;

if(l==6)

l=0;

m--;

}

}

while(s3.charAt(l)=='X')

{

l++;

if(l==6)

l=0;

}

s3=s3.substring(0,l)+"X"+s3.substring(l+1);

k--;

}

for(int i=0;i<s3.length();i++)

{

if(s3.charAt(i)!='X')

e=s3.charAt(i);

}

switch(e)

{

case 'F':System.out.println("Friends");

break;

case 'L':System.out.println("Love");

break;

case 'A':System.out.println("Affection");

break;

case 'M':System.out.println("Marriage");

break;

case 'E':System.out.println("Enemy");

break;

case 'S':System.out.println("Sister");

break;

default :break;

}

}

}

15. Sort an array in order of recently used elements. If an element is searched, array should be re-arranged as this is the recently searched element

class GFG{

public static void fixArray(int ar[], int n)

{

int i, j, temp;

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

{

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

{

// Check is any ar[j]

// exists such that

// ar[j] is equal to i

if (ar[j] == i)

{

temp = ar[j];

ar[j] = ar[i];

ar[i] = temp;

break;

}

}

}

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

{

// If not present

if (ar[i] != i)

{

ar[i] = -1;

}

}

System.out.println("Array after Rearranging");

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

{

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

}

}

public static void main(String[] args)

{

int n, ar[] = { -1, -1, 6, 1, 9,

3, 2, -1, 4, -1 };

n = ar.length;

fixArray(ar, n);

}

}

**Java Stream API**

**Question1:The task is to group an array of employee records into a data map organized by job title**

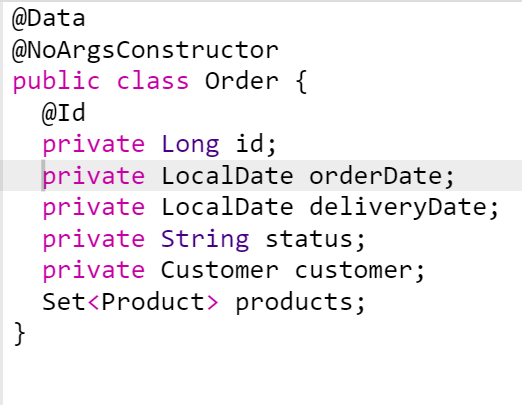
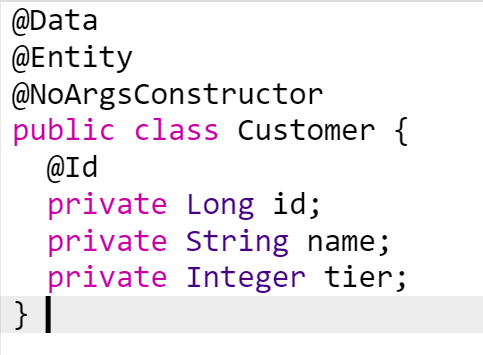
**Answer*employeeList.stream() .collect(Collectors.groupingBy(Employee::getTitle));***

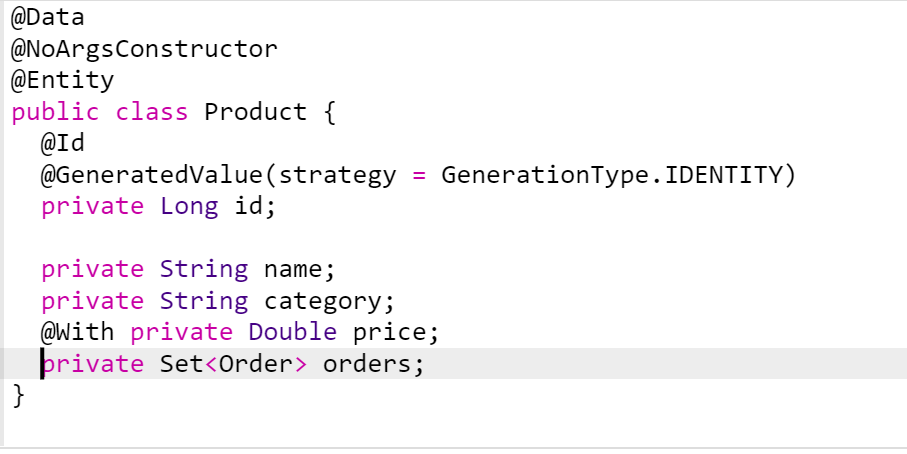
**Question 2: The task is to calculate average salary of all employee in the list**

**Answer employeeList.stream().mapToInt(employee->employee.getSalary()).average() .getAsDouble();**

***CUSTOMER, ORDER, Product Data Model Stream API Questions***

**The evaluations are based on a data model — customer, order and product. Customers can place multiple orders and so it is a one-to-many relationship while the relationship between products and orders is many-to-many**





**Question 3:Obtain a list of products belongs to category “Books” with price > 100**

**Answer**

List<Product>result=productlist.stream().filter(p-> p.getCategory().equalsIgnoreCase("Books")

.filter(p -> p.getPrice() > 100) .collect(Collectors.toList());

**Question 4:Obtain a list of order with products belong to category “Baby”**

**Answer**

List<Order> result = orderlist.stream().

filter (o -> o.getProducts(). stream ().anyMatch(p -> p.getCategory().equalsIgnoreCase("**Baby**"))). collect(Collectors.toList());

**Question 5:Obtain a list of products with category = “Toys” and then apply 10% discount**

**Answer**

List<Product> result = productlist.stream()

.filter(p -> p.getCategory().equalsIgnoreCase("**Toys**"))

.map(p -> p.withPrice(p.getPrice() \* 0.9))

.collect(Collectors.toList())

**Question 6: Obtain a list of products ordered by customer of tier 2 between 01-Feb-2021 and 01-Apr-2021**

**Answer**

List<Product> result = orderList. stream ()

. filter (o -> o.getCustomer().getTier() == 2)

. filter (o -> o.getOrderDate().compareTo(LocalDate.of(2021, 2, 1)) >= 0)

. filter (o -> o.getOrderDate().compareTo(LocalDate.of(2021, 4, 1)) <= 0)

.flatMap(o -> o.getProducts(). stream ())

. distinct ()

. collect(Collectors.toList());

**Question 7: Get the cheapest products of “Books” category**

**Answer**

Optional<Product> result = productlist

. stream()

. filter(p -> p.getCategory().equalsIgnoreCase("Books"))

. sorted(Comparator.comparing(Product::getPrice))

.findFirst();

**Answer 2:**

Optional<Product> result = productlist.stream()

.filter(p -> p.getCategory().equalsIgnoreCase("Books"))

.min(Comparator.comparing(Product::getPrice));

**Question 8: Get the 3 most recent placed order**

**Answer**

List<Order> result = ordeList.stream()

. sorted(Comparator.comparing(Order::getOrderDate).reversed())

. limit (3). collect (Collectors.toList());

**Question 9: Get a list of orders which were ordered on 15-Mar-2021, log the order records to the console and then return its product list**

**Answer**

List<Product> result = productList.stream()

.filter(o -> o.getOrderDate().isEqual(LocalDate.of(2021, 3, 15)))

.peek(o -> System.out.println(o.toString()))

.flatMap(o -> o.getProducts().stream())

.distinct()

.collect(Collectors.toList());

**Question 10: Calculate total lump sum of all orders placed in Feb 2021**

**Answer**

Double result = orderList .stream()

. filter (o -> o.getOrderDate().compareTo(LocalDate.of(2021, 2, 1)) >= 0)

. filter (o -> o.getOrderDate().compareTo(LocalDate.of(2021, 3, 1)) < 0)

.flatMap(o -> o.getProducts(). stream ())

.mapToDouble(p -> p.getPrice())

. sum ();

**Question 11: Calculate order average payment placed on 14-Mar-2021**

**Answer**

Double result = orderlist. stream()

.filter(o -> o.getOrderDate().isEqual(LocalDate.of(2021, 3, 14)))

.flatMap(o -> o.getProducts().stream())

.mapToDouble(p -> p.getPrice())

.average().getAsDouble();

**Question 12: Obtain a collection of statistic figures (i.e., sum, average, max, min, count) for all products of category “Books”**

**Answer**

DoubleSummaryStatistics statistics = productList.stream() . filter(p->p.getCategory().equalsIgnoreCase("Books")) .mapToDouble(p->p.getPrice()) .summaryStatistics();

System.out.println(String.format("count = %1$d, average = %2$f, max = %3$f, min = %4$f, sum = %5$f",

System.out.println(

String.format("count= %1$d, average = %2$f, max = %3$f, min = %4$f, sum = %5$f",

statistics.getCount(),statistics.getAverage(),statistics.getMax(),statistics.getMin(),statistics.getSum())));

**Question 13: Obtain a data map with order id and order’s product count**

**Answer**

Map<Long, Integer> result = orderList.stream().

collect(Collectors.toMap(order -> order.getId(),

order -> order.getProducts(). size ()));

**Question 14: Produce a data map with order records grouped by customer**

**Answer**

Map<Customer, List<Order>> result = orderList.stream()

. collect(Collectors.groupingBy(Order::getCustomer));

**Question 15: Produce a data map with order record and product total sum**

**Answer**

Map<Order, Double> result = orderRepo.findAll() .stream()

.collect(Collectors.toMap(Function.identity(),

order -> order.getProducts().stream().mapToDouble(p -> p.getPrice()).sum()));

Questions 16: Write a program (WAP) to swap two numbers without using third variable

Note: to test java core logic

Questions 17: WAP to find second max salary in a given list of employees with salary

Note: to test java 8 knowledge using stream api

Optional maxSal= employeeList.stream().max(Comparator.comparingDouble(Employee::getSalary));

Optional sec=employeeList.stream().filter( s -> s.getSalary() != maxSal.get().getSalary()).max(Comparator.comparing(Employee::getSalary));

System.out.println(“sec sal:”+sec.get());

Questions 18: Given an array print the elements in alternate order (descending & ascending)

Ex:

input: 1234567

Output: 7162534

class AlternativeString

{

static void alternateSort(int arr[], int n)

{

Arrays.sort(arr);

int i = 0, j = n-1;

while (i < j) {

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

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

}

if (n % 2 != 0)

System.out.print(arr[i]);

}

public static void main (String[] args)

{

int arr[] = {1, 2,3,4,5,6,7};

int n = arr.length;

alternateSort(arr, n);

}

}

Questions 19: Given two arrays, merge them without elements are not repeated

class Rextester

{

public static void main(String args[])

{

Scanner input=new Scanner(System.in);

System.out.println("Enter the size of array one:");

int arrayOneSize =input.nextInt();

System.out.println("Enter the size of array two:");

int arrayTwoSize =input.nextInt();

int mergedArray[]=new int[arrayOneSize+arrayTwoSize];

int mergedArrayTemp[]=new int[arrayOneSize+arrayTwoSize];

int FinalmergedArray[]=new int[arrayOneSize+arrayTwoSize];

System.out.println("Enter the elements of array one:");

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

{

mergedArray[i]=input.nextInt();

mergedArrayTemp[i]=mergedArray[i];

}

System.out.println("Enter the elements of array two:");

for(int i=arrayOneSize;i<arrayOneSize+arrayTwoSize;i++)

{

mergedArray[i]=input.nextInt();

mergedArrayTemp[i]=mergedArray[i];

}

/\*

for(int i=0;i<arrayOneSize+arrayTwoSize;i++)

System.out.print(mergedArray[i]);

\*/

for(int i=0;i<(arrayOneSize+arrayTwoSize)-1;i++)

{

for(int j=i+1;j<(arrayOneSize+arrayTwoSize);j++)

{

if(mergedArrayTemp[i]==mergedArrayTemp[j])

{

mergedArrayTemp[j]=-100;

}

}

}

System.out.println("Final merged array without duplicate:");

int finalArrayIteration=0;

for(int i=0;i<arrayOneSize+arrayTwoSize;i++)

{

if(mergedArray[i]==mergedArrayTemp[i])

{

FinalmergedArray[finalArrayIteration]=mergedArray[i];

finalArrayIteration++;

System.out.print(mergedArray[i]);

}

}

}

}