**Day 1 Que 1-----Java word analyzer using collections**

**Problem Description:**

Write a Java program that reads a text file containing a list of words and uses Java Collections to perform the following operations:

    1. Read the text file and create a list of words, where each word is a 'String' object.

    2. Create a set of all unique words in the list.

    3. Create a map that associates each word with the number of times it appears in the list.

    4. Find the three most common words in the list and display them along with their frequency.

**Your implementation should conform to the following requirements:**

    1. The program should use java collections to perform the various operations on the list of words.

    2. The program should use the **BufferedReader** class to read the text file.

    3. The program should handle errors gracefully (e.g., file not found, invalid data).

    4. The program should have a clean and user-friendly interface.

    5. The skeleton implementation already has a **words.txt** file populated with sample data.

**Your task is to complete the methods defined in the file below:**

1. **../src/main/java/com/tasks/problem/WordAnalyzerService.java**

**Notes:**

1. Ensure that the structure and datatype of the components are followed as specified in the comments to ensure that the code is evaluated correctly.

2. Read the Java doc comments for the public methods in the **WordAnalyzerService** class. You need to implement your code in line with these comments

package com.tasks.problem;

import java.io.BufferedReader;

import java.io.FileReader;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.HashMap;

import java.util.HashSet;

import java.util.LinkedHashMap;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import java.util.Set;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.stream.Collector;

import java.util.stream.Collectors;

public class WordAnalyzerService {

    String fileData;

    String[] words;

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

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

    /\*\*

     \*

     \* @return number of words present in the file words.txt

     \* @throws Exception

     \*/

    public long readFileAndReturnNoOfWords()  {

            //@todo Use BufferedReader to read the file and store the words into words variable

            // delete the last new line separator

        List<String> listOfWords = new ArrayList<>();

        try{

            BufferedReader br = new BufferedReader(new FileReader("words.txt"));

            while((fileData = br.readLine()) != null){

                String[] temp = fileData.split(" ");

                for(String word:temp){

                    listOfWords.add(word);

                }

            }

        }

        catch(FileNotFoundException e){

            e.printStackTrace();

        }

        catch(IOException e){

            e.printStackTrace();

        }

        words = listOfWords.toArray(new String[0]);

        return words.length;

    }

    /\*

     \*

     \* @return the unique words present in the file. These words should be populated in the set variable declared above.

     \*/

    public long createSetOfUniqueWordsAndReturnUniqueCount() {

            if(words == null) {

                readFileAndReturnNoOfWords();

            }

        //@todo  Add words to the collection uniquely

        for(String word:words){

            set.add(word);

        }

    return set.size();

    }

    /\*\*

     \* Populate the map variable with key-value mapping of word-count, count representing how many times the word appeared in the file.

     \*/

    public void createMapOfWord\_Count() {

        if(words == null) {

            readFileAndReturnNoOfWords();

        }

        //@todo Populate the map variable by writing appropriate code

        for(String word:words){

            map.put(word,map.getOrDefault(word, 0L)+1);

        }

    }

    /\*\*

     \*

     \* @param word - input word

     \* @return the number of times the input word appeared in the file

     \*/

    public long getOccurrencesOf(String word) {

        if(map.keySet().size() == 0) {

            createMapOfWord\_Count();

        }

        //@todo Get the count

        return map.get(word);

    }

    /\*\*

     \*

     \* @return topp 3 words sorted (desc) by number of occurrences in the file

     \*/

    public List<String> findThreeMostCommonWords(){

        if(map.keySet().size()==0){

            createMapOfWord\_Count();

        }

        Map<String,Long> sortedMap = sortByValue(map);

        List<String> topThreeWords = new ArrayList<>();

        int size = sortedMap.size();

        int count = 0;

        for(Map.Entry<String,Long> entry:sortedMap.entrySet()){

            if(count >= size -3){

                topThreeWords.add(entry.getKey());

            }

            count++;

        }

        List<String> keys = map.entrySet().stream().sorted(Map.Entry.

        <String,Long> comparingByValue().reversed()).limit(3).map(Map.Entry::getKey).collect(Collectors.toList());

        return keys;

    }

    /\*\*

     \* Sort the map keys based on key value with most commonly used word at the top.

     \* @param hm

     \* @return

     \*/

    private static Map<String, Long>

    sortByValue(Map<String, Long> hm)

    {

        //@todo Sort the map on the basis of value of the key in the Map.

        // Creating a list from elements of HashMap

        System.out.println(hm.toString());

        List<Map.Entry<String,Long>> unsortedList = new ArrayList<>(hm.entrySet());

        System.out.println(unsortedList.toString());

        Collections.sort(unsortedList,new Comparator<Map.Entry<String,Long>>() {

            @Override

            public int compare(Map.Entry<String,Long> first,Map.Entry<String,Long>second){

                return first.getValue().compareTo(second.getValue());

            }

        });

        // Sorting the list using Collections.sort() method

        // using Comparator

        // putting the  data from sorted list back to hashmap

        Map<String,Long> sortedMap = new LinkedHashMap<>();

        for(Map.Entry<String,Long> item:unsortedList){

            sortedMap.put(item.getKey(),item.getValue());

        }

        // returning the sorted HashMap

        return sortedMap;

    }

}

**Day 1 Que 2----Writing a transaction service in Java**

**Problem statement:**

You need to write a Java program that reads a CSV file containing a list of bank transactions and performs the following operations:

1. Read the CSV file and create a list of "Transaction' objects, where each object represents a transaction in the file.
2. Calculate the total balance of the account by summing up all the transaction amounts.
3. Find the transaction with the highest amount.
4. Find the transaction with the lowest amount.
5. Find the average transaction amount.
6. Handle errors gracefully by catching and handling any exceptions that may occur during file reading or transaction processing.

**Your implementation should conform to the following requirements:**

1. The program should use Exception Handling to handle any potential errors during file reading or transaction processing.
2. The program should use the "CsvToBean" class from the 'opencsv'" library to read the CSV file.
3. The program should use appropriate exception handling techniques, such as try-catch blocks, to catch and handle any exceptions that may occur.
4. The program should have a clean and user-friendly interface.

**Your task is to complete the methods defined in the file below:**

1. ../src/main/java/com/tasks/problem/TransactionService.java

**Notes:**

* Ensure that the structure and datatype of the components are followed as specified in the comments to ensure that the code is evaluated correctly.
* You can add a helper method in the same service class or a helper class in the same/other package.
* You need to provide complete implementation to all the public service methods in **TransactionService.java**
* package com.tasks.problem;
* import java.io.BufferedReader;
* import java.io.FileReader;
* import java.io.Reader;
* import java.util.List;
* import com.opencsv.bean.CsvToBean;
* import com.opencsv.bean.CsvToBeanBuilder;
* public class TransactionService {
* List<Transaction> transactions;

* public List<Transaction> getAllTransactions() {
* try {
* Reader reader = new BufferedReader(new FileReader("transactions.csv"));
* //@todo You can initialize CsvBean<T> class and then call the parse method to get the list
* CsvToBean<Transaction> csvToBean = new CsvToBeanBuilder<Transaction>(reader)
* .withType(Transaction.class)
* .withIgnoreLeadingWhiteSpace(true)
* .build();
* transactions = csvToBean.parse();
* return transactions;
* }

* // TODO Auto-generated catch block
* catch(Exception e){
* e.printStackTrace();
* return null;
* }
* }
* public Double getTotalTransactionAmount() {
* if(transactions == null) {
* transactions=getAllTransactions();
* }
* double totalAmount = 0d;
* //@todo Write code to assign total transaction amount to amt variable
* for(Transaction transaction:transactions){
* totalAmount = totalAmount + transaction.getAmount();
* }
* return totalAmount;
* }

* public Transaction getTransactionWithHighestAmount() {
* if(transactions == null) {
* transactions = getAllTransactions();
* }
* double highestAmount = Double.MIN\_VALUE;
* Transaction highTransaction = null;
* //@todo Write code to get the transaction object with highest amount
* for(Transaction transaction:transactions){
* if(transaction.getAmount()>highestAmount){
* highestAmount=transaction.getAmount();
* highTransaction=transaction;
* }
* }
* return highTransaction;
* }
* public Transaction getTransactionWithLowestmount() {
* if(transactions == null) {
* getAllTransactions();
* }
* double lowestAmount = Double.MAX\_VALUE;
* Transaction lowestTransaction = null;
* //@todo Write code to figure out transaction with lowest amount
* for(Transaction transaction:transactions){
* if(transaction.getAmount()<lowestAmount){
* lowestAmount=transaction.getAmount();
* lowestTransaction=transaction;
* }
* }
* return lowestTransaction;
* }
* public Double getAverageTransactionAmount() {
* if(transactions == null) {
* transactions = getAllTransactions();
* }
* //@todo Write code to get the average of transaction amounts & return it.
* double totalAmount = getTotalTransactionAmount();
* return totalAmount/transactions.size();
* }
* }

**Day 2 Que 1----Writing a product service in Java**

**Problem Description:**

Write a Java program that reads a CV file containing a list of bank transactions and performs the following operations:

    1. Read the CSV file and create a list of "Product' objects, where each object represents a product in the file.

    2. Filter the list of products to include only those with a price greater than a specified value.

    3. Group the products by category and calculate the total value of products in each category.

    4. Calculate the average price of all products in the list.

    5. Find the product with the highest price in the list.

    6. The public methods service class ProductService.java has method skeletons, that you need to update to fulfill above expectations

**Your implementation should conform to the following requirements:**

   1. The program should use Exception Handling to handle any potential errors during file reading or transaction processing.

    2. The program should use the "CsvToBean" class from the 'opencsv'" library to read the CSV file.

    3. The program should use appropriate exception handling techniques, such as try-catch blocks, to catch and handle any exceptions that may occur.

    4. The program should have a clean and user-friendly interface.

**Your task is to complete the methods defined in the file below:**

    1. ../src/main/java/com/tasks/problem/ProductService.java

**Notes:**

    1. Ensure that the structure and datatype of the components are followed as specified in the comments to ensure that the code is evaluated correctly.

    2. You can add a helper method in the same service class or a helper class in the same/other package.

package com.tasks.problem;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.Reader;

import java.util.ArrayList;

import java.util.List;

import java.util.Comparator;

import java.util.DoubleSummaryStatistics;

import java.util.stream.Collectors;

import com.opencsv.bean.CsvToBean;

import com.opencsv.bean.CsvToBeanBuilder;

public class ProductService {

    List<Product> products;

    public List<Product> getAllProducts() {

        try {

            //@todo write code using OpenCSV to read the file products.csv and marshalling it to list of products.

            if(products==null)

            {

                Reader reader = new BufferedReader(new FileReader("products.csv"));

                CsvToBean<Product> csvToBean = new CsvToBeanBuilder<Product>(reader)

                .withType(Product.class).withIgnoreLeadingWhiteSpace(true)

                .build();

                products = csvToBean.parse();

                reader.close();

            }

            return products;

        } catch (Exception e) {

            // TODO Auto-generated catch block

            throw new RuntimeException("FAILURE\_TO\_PROCESS\_CSV");

        }

    }

    public List<Product> getProductsWithPriceGreaterThan(Double price){

        if(this.products == null) {

            getAllProducts();

        }

        //@todo Write code to filter products having price greater than the passed price agrument

        return products.stream()

        .filter(product -> product.getPrice() > price).collect(Collectors.toList());

    }

    public Double groupByCategoryAndAggregateValue(String category){

        if(this.products == null) {

            getAllProducts();

        }

        //@todo Write code to group by category argument apssed as method parameter and then retutn the aggregated price of products belonging to the category.

        double tot = products.stream()

        .filter(product -> product.getCategory().equals(category))

        .mapToDouble(Product::getPrice).sum();

        return tot;

    }

    public Double calculateAverageOfAllProducts(){

        if(this.products == null) {

            getAllProducts();

        }

        //@todo Write code to evaluate the average of prices of all products

        // double tot = 0d;

        // return null;

        DoubleSummaryStatistics stats = products.stream()

        .mapToDouble(Product::getPrice)

        .summaryStatistics();

        return stats.getAverage();

    }

    public Product findProductWithHighestPrice(){

        if(this.products == null) {

            getAllProducts();

        }

        //@todo Write code to fetch the product with highest price

        // double tot = 0d;

        //  }

        return products.stream()

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

        .orElse(null);

    }

}

**Day 2 Que 2----Working with substrings in Java**

**Task Description:**

You are given a string **str**, find the length of the longest substring in **str** without repeating characters. Implement an efficient solution with a time complexity of O(n).

**Example:**

Let the given string be “ABDEFGABEF”, the longest substrings without repeating characters here are “BDEFGA” and “DEFGAB”, with length 6. Similarly, for “BBBB” the longest substring is “B”, with length 1.

**Your task is to complete the helper methods defined in the file below:**

1. **../src/main/java/com/tasks/problem/LongestSubstring.java**

**Notes:**

1. Ensure that the structure and datatype of the components are followed as specified in the comments to ensure that the code is evaluated correctly.
2. Use **Test App & Submit** option often so you will be guided by test error messages.
3. When you delete or edit something you shouldn’t have, test messages will give an error accordingly
4. You will receive a congratulations message upon successful completion of the task.

package com.tasks.problem;

import java.util.HashSet;

public class LongestSubstring {

    public static int lengthOfLongestSubstring(String str) {

        String test = "";

        HashSet<Character> set = new HashSet<>();

        // Result

        int maxLength = 0;

        // Return zero if string is empty

        // if (str.isEmpty()) {

        // //     return 0;

        // }

        // Return one if string length is one

        // else if (str.length() == 1) {

        //     return 1;

        // }

        // for (char c : str.toCharArray()) {

        //     String current = String.valueOf(c);

            // If string already contains the character

            // Then substring after repeating character

            //@todo

            // test = test + String.valueOf(c);

            //@todo Evaluate the value for maxLength

        // }

        //@todo return maxLength

        // return -1;

        int left = 0;

        int right = 0;

        while(right<str.length()){

            char c =str.charAt(right);

            if(!set.contains(c)){

                set.add(c);

                maxLength=Math.max(maxLength,right-left+1);

                right++;

            }

            else{

                set.remove(str.charAt(left));

                left++;

            }

        }

        return maxLength;

    }

}

**Day 3 Que 1----Java LRU Cache**

**Example**

Let the requests made be as follows:

Put one-one

Put two-two

Put three-three

Get one

put four-four

Now, if you try a get("two), you should get null since two is the least used key and hence is removed.

**Your task is to complete the following files**

* **../src/main/java/com/tasks/problem/LRUCache.java**

**Notes:**

* Ensure that the structure and datatype of the components are followed as specified in the comments to ensure that the code is evaluated correctly.
* Use **Test App & Submit** option often so you will be guided by test error messages.
* When you delete or edit something you shouldn’t have, test messages will give an error accordingly
* You will receive a congratulations message upon successful completion of the task.
* package com.tasks.problem;
* import java.util.concurrent.ConcurrentHashMap;
* import java.util.concurrent.ConcurrentLinkedQueue;
* import java.util.concurrent.locks.ReadWriteLock;
* import java.util.concurrent.locks.ReentrantReadWriteLock;
* public class LRUCache<K,V> {
* private  ConcurrentLinkedQueue<K> concurrentLinkedQueue = new ConcurrentLinkedQueue<K>();
* private  ConcurrentHashMap<K,V> concurrentHashMap = new ConcurrentHashMap<K, V>();
* private ReadWriteLock readWriteLock = new ReentrantReadWriteLock();
* int maxSize=0;
* public LRUCache(final int MAX\_SIZE){
* this.maxSize=MAX\_SIZE;
* }
* public V get(K key){
* readWriteLock.readLock().lock();
* try {
* V v=null;
* if(concurrentHashMap.containsKey(key)){
* concurrentLinkedQueue.remove(key);
* v= concurrentHashMap.get(key);
* concurrentLinkedQueue.add(key);
* }
* //@todo return the appropriate object
* return v;
* }finally{
* readWriteLock.readLock().unlock();
* }
* }
* public int size() {
* readWriteLock.readLock().lock();
* try {
* return concurrentHashMap.size();
* }
* finally{
* readWriteLock.readLock().unlock();
* }
* }
* public void put(K key,V value){
* readWriteLock.writeLock().lock();
* try {
* if(concurrentHashMap.containsKey(key)){
* concurrentLinkedQueue.remove(key);
* }
* while(concurrentLinkedQueue.size() >=maxSize){
* //@todo Get the least used key and delete it
* K leastUsedKey = concurrentLinkedQueue.poll();
* if(leastUsedKey != null){
* concurrentHashMap.remove(leastUsedKey);
* }
* }
* //@todo Add the key
* concurrentLinkedQueue.add(key);
* concurrentHashMap.put(key, value);
* //return value;
* } finally{
* readWriteLock.writeLock().unlock();
* }
* }
* }

**Day 3 Que 2----Java Binary tree serialization**

**Serialize and Deserialize a Binary Tree**

Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or transmitted across a network and later reconstructed. Deserialization is the reverse process of recreating the original data structure or object from the serialized format.

Write two methods to serialize and deserialize a binary tree.

**Your task is to complete the helper methods defined in the file below:**

1. **../src/main/java/com/tasks/problem/SerializeDeserializeBinaryTree.java**

**Example**

Input binary tree

      TreeNode root = new TreeNode(1);

    root.left = new TreeNode(2);

    root.right = new TreeNode(3);

    root.left.left = new TreeNode(4);

    root.left.right = new TreeNode(5);

Serialization Output - 1,2,4,null,null,5,null,null,3,null,null

Deserialization Output - Same Java structure (root node) with same left/right/.. associations as above.

**Notes:**

1.Ensure that the structure and datatype of the components are followed as specified in the comments to ensure that the code is evaluated correctly.

2.Use Test App & Submit option often so you will be guided by test error messages.

3.When you delete or edit something you shouldn’t have, test messages will give an error accordingly

4.You will receive a congratulations message upon successful completion of the task.

package com.tasks.problem;

import java.util.ArrayList;

import java.util.List;

import java.util.Stack;

public class SerializeDeserializeBinaryTree {

    public static class TreeNode {

        public int val;

        public TreeNode left;

        public TreeNode right;

        public TreeNode(int x) {

            val = x;

        }

    }

    public static String serialize(TreeNode root)

    {

        if (root == null) {

            return null;

        }

        Stack<TreeNode> s = new Stack<>();

        s.push(root);

        List<String> l = new ArrayList<>();

        while (!s.isEmpty()) {

            TreeNode t = s.pop();

            // If current node is NULL, store marker

            if (t == null) {

                l.add("null");

            }

            else {

                // Else, store current node

                // and recur for its children

                l.add("" + t.val);

                //@todo push the nodes (left and right) onto the stack

                s.push(t.right);

                s.push(t.left);

            }

        }

        //@todo return the String output after using the join function

        return String.join(",",l);

    }

    static int t;

    // Decodes your encoded data to tree.

    public static TreeNode deserialize(String data)

    {

        if (data == null)

            return null;

        t = 0;

        String[] arr = data.split(",");

        return helper(arr);

    }

    public static TreeNode helper(String[] arr)

    {

        if (t>=arr.length||arr[t].equals("null"))

            return null;

        // Create node with this item

        // and recur for children

        TreeNode root

            = new TreeNode(Integer.parseInt(arr[t]));

        t++;

        //@todo Evaluate root.left

        root.left=helper(arr);

        t++;

        //@todo Evualate root.right

        root.right=helper(arr);

        return root;

    }

    private static void printPreOrderTraversal(TreeNode node) {

        if (node != null) {

            System.out.println(node.val+"");

            printPreOrderTraversal(node.left);

            printPreOrderTraversal(node.right);

        }

    }

}