1. **WORD ANALYZER USING COLLECTIONS**

**Problem statement:1**

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**

Transaction.java

* package com.tasks.problem;
* import java.time.LocalDate;
* import com.opencsv.bean.CsvBindByName;
* import com.opencsv.bean.CsvDate;
* public class Transaction {
* @CsvBindByName(column = "Type")
* private String type;
* @CsvBindByName(column = "Amount")
* private Double amount;
* @CsvBindByName(column = "Date")
* @CsvDate(value = "yyyy-MM-dd")
* private LocalDate date;
* public String getType() {
* return type;
* }
* public void setType(String type) {
* this.type = type;
* }
* public Double getAmount() {
* return amount;
* }
* public void setAmount(double amount) {
* this.amount = amount;
* }
* public LocalDate getDate() {
* return date;
* }
* public void setDate(LocalDate date) {
* this.date = date;
* }
* }

TransactionService

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();

        }

        //@todo Write code to assign total transaction amount to amt variable

        double totalAmount = 0d;

        for(Transaction transaction:transactions){

            totalAmount=totalAmount+transaction.getAmount();

        }

        return totalAmount;

    }

    public Transaction getTransactionWithHighestAmount() {

        if(transactions == null) {

            transactions=getAllTransactions();

        }

        //@todo Write code to get the transaction object with highest amount

        double highestAmount= Double.MIN\_VALUE;

        Transaction highestTransaction = null;

         for(Transaction transaction:transactions){

            if(transaction.getAmount()>highestAmount){

                highestAmount=transaction.getAmount();

                highestTransaction=transaction;

            }

         }

    return highestTransaction;

    }

public Transaction getTransactionWithLowestmount() {

        if(transactions == null) {

            getAllTransactions();

        }

        //@todo Write code to figure out transaction with lowest amount

        double lowestAmount = Double.MAX\_VALUE;

        Transaction lowestTransaction = null;

            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();

    }

}

1. **Transaction service in java**

**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

1. **product service**

**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.

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

Product.java

package com.tasks.problem;

import java.time.LocalDate;

import com.opencsv.bean.CsvBindByName;

import com.opencsv.bean.CsvDate;

public class Product {

    @CsvBindByName(column = "Name")

    private String name;

    @CsvBindByName(column = "Category")

    private String category;

    @CsvBindByName(column = "Price")

    private Double price;

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getCategory() {

        return category;

    }

    public void setCategory(String category) {

        this.category = category;

    }

    public Double getPrice() {

        return price;

    }

    public void setPrice(Double price) {

        this.price = price;

    }

}

Productservice.

package com.tasks.problem;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.Reader;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.DoubleSummaryStatistics;

import java.util.List;

import java.util.stream.Collector;

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

        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

        return products.stream().max(Comparator.comparing(Product::getPrice)).orElse(null);

    }

}

1. **substring in java**

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.
5. package com.tasks.problem;
6. public class LongestSubstring {
8. public static int lengthOfLongestSubstring(String str) {
9. String test = "";
11. // Result
12. int maxLength = -1;
14. // Return zero if string is empty
15. if (str.isEmpty()) {
16. return 0;
17. }
18. // Return one if string length is one
19. else if (str.length() == 1) {
20. return 1;
21. }
22. for (char c : str.toCharArray()) {
23. String current = String.valueOf(c);
25. // If string already contains the character
26. // Then substring after repeating character
27. if(test.contains(current)){
28. test=test.substring(test.indexOf(current)+1);
29. }
30. //@todo
31. test = test + String.valueOf(c);
32. //@todo Evaluate the value for maxLength
33. maxLength=Math.max(maxLength,test.length());
34. }
35. //@todo return maxLength
36. return maxLength;
37. }
39. }
40. **JAVA LRU CACHE**

Implement a thread safe Least Recently Used (LRU) cache that supports the following operations:

1. **get(key)**
2. **put(key, value)**

The cache should have a specified capacity, and when the number of unique keys exceeds the capacity, the least recently used key should be removed. Make the cache **thread**-**safe** using synchronization or concurrency utilities.

**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 leastKeyused=concurrentLinkedQueue.poll();

           if(leastKeyused !=null){

            concurrentHashMap.remove(leastKeyused);

           }

        }

        //@todo Add the key

            concurrentLinkedQueue.add(key);

            concurrentHashMap.put(key, value);

        //return value;

        } finally{

            readWriteLock.writeLock().unlock();

        }

    }

}

1. **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 (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) {

            printPreOrderTraversal(node.left);

            printPreOrderTraversal(node.right);

        }

    }

}

**DECEMBER BATCH**

**🡪Entity**

**BOOK.java**

**package com.ust.websecurity.entity;**

**import lombok.AllArgsConstructor;**

**import lombok.Data;**

**import lombok.NoArgsConstructor;**

**import javax.persistence.Entity;**

**import javax.persistence.Id;**

**@Entity**

**@Data**

**@AllArgsConstructor**

**@NoArgsConstructor**

**public class Book {**

**@Id**

**private long id;**

**private String name;**

**}**

**ISSUE.java**

**package com.ust.websecurity.entity;**

**import lombok.AllArgsConstructor;**

**import lombok.Data;**

**import lombok.NoArgsConstructor;**

**import javax.persistence.\*;**

**import java.time.LocalDate;**

**@Entity**

**@Data**

**@AllArgsConstructor**

**@NoArgsConstructor**

**public class Issue {**

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

**private long id;**

**private LocalDate issueDate;**

**private LocalDate returnDate;**

**private Integer period;**

**private Integer fine;**

**// Define the many-to-one relationship with the User entity**

**@ManyToOne**

**@JoinColumn(name = "user\_id") // Replace "user\_id" with the actual name of the column in the issue table referencing the User entity's primary key**

**private User user;**

**// Other issue fields/columns**

**// ...**

**}**

**User.java**

**package com.ust.websecurity.entity;**

**import lombok.AllArgsConstructor;**

**import lombok.Data;**

**import lombok.NoArgsConstructor;**

**import javax.persistence.\*;**

**import java.util.List;**

**@Entity**

**@Data**

**@AllArgsConstructor**

**@NoArgsConstructor**

**public class User {**

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

**private long id;**

**private String name;**

**private Boolean subscribed=false;**

**// Define the relationship with the issue table**

**@OneToMany(mappedBy = "user") // Assuming "user" is the name of the column in the issue table referencing the User entity**

**private List<Issue> issues;**

**}**

**🡪🡪Repository**

**IssueRepository**

**package com.ust.websecurity.repository;**

**import com.ust.websecurity.entity.Issue;**

**import org.springframework.data.jpa.repository.JpaRepository;**

**import org.springframework.stereotype.Repository;**

**@Repository**

**public interface IssueRepository extends JpaRepository<Issue,Long> {**

**}**

**UserRepository**

**package com.ust.websecurity.repository;**

**import com.ust.websecurity.entity.User;**

**import org.springframework.data.jpa.repository.JpaRepository;**

**import org.springframework.stereotype.Repository;**

**@Repository**

**public interface UserRepository extends JpaRepository<User,Long> {**

**}**

**🡪🡪Controller**

**package com.ust.websecurity.controller;**

**import com.ust.websecurity.entity.Issue;**

**import com.ust.websecurity.entity.User;**

**import com.ust.websecurity.exception.UserNotSubscribedException;**

**import com.ust.websecurity.repository.IssueRepository;**

**import com.ust.websecurity.repository.UserRepository;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.http.ResponseEntity;**

**import org.springframework.stereotype.Repository;**

**import org.springframework.web.bind.annotation.\*;**

**import javax.swing.text.html.Option;**

**import java.util.List;**

**import java.util.Optional;**

**@RestController**

**@RequestMapping("/api/v1")**

**public class LibraryController {**

**@Autowired**

**private UserRepository userRepository;**

**@Autowired**

**private IssueRepository issueRepository;**

**@PostMapping("/issue-book")**

**public ResponseEntity<Issue> issuebook(@RequestBody Issue issue){**

**final var user = userRepository.findById(issue.getUser().getId());**

**if(user.isEmpty()){**

**return ResponseEntity.noContent().build();**

**}**

**if(user.get().getSubscribed()==false){**

**throw new UserNotSubscribedException("fghj");**

**}**

**return ResponseEntity.ok().body(issueRepository.save(issue));**

**}**

**@PostMapping("/user")**

**public ResponseEntity<User> createUser(@RequestBody User user){**

**return ResponseEntity.ok().body(userRepository.save(user));**

**}**

**@GetMapping("renew-user-subscription/{id}")**

**public ResponseEntity<User> renewUserSubcription(@PathVariable Long id){**

**final var user = userRepository.findById(id);**

**if(user.isEmpty()){**

**return ResponseEntity.noContent().build();**

**}**

**user.get().setSubscribed(true);**

**return ResponseEntity.ok().body(userRepository.save(user.get()));**

**}**

**@PutMapping("/user")**

**public ResponseEntity<User> updateUser(@RequestBody User user){**

**User updated = null;**

**Optional<User> u = userRepository.findById(user.getId());**

**if(u.isPresent()){**

**updated = u.get();**

**updated.setId(user.getId());**

**updated.setSubscribed(user.getSubscribed());**

**updated.setName(user.getName());**

**return ResponseEntity.ok(userRepository.save(updated));**

**}**

**else{**

**return ResponseEntity.notFound().build();**

**}**

**}**

**@PutMapping("/issue-book")**

**public ResponseEntity<Issue> updateIssue(@RequestBody Issue issue){**

**Issue issu = null;**

**Optional<Issue> op = issueRepository.findById(issue.getId());**

**if(op.isPresent()){**

**issu = op.get();**

**issu.setId(issue.getId());**

**issu.setFine(issue.getFine());**

**issu.setPeriod(issue.getPeriod());**

**issu.setIssueDate(issue.getIssueDate());**

**issu.setReturnDate(issue.getReturnDate());**

**return ResponseEntity.ok(issueRepository.save(issu));**

**}**

**else{**

**return ResponseEntity.noContent().build();**

**}**

**}**

**@DeleteMapping("/usr/{id}")**

**public ResponseEntity<String> deleteUser(@PathVariable long id){**

**Optional<User> usr = userRepository.findById(id);**

**if(usr.isEmpty()){**

**ResponseEntity.notFound().build();**

**}**

**else{**

**userRepository.deleteById(id);**

**}**

**return ResponseEntity.ok("deleted");**

**}**

**@DeleteMapping("/issue/{id}")**

**public ResponseEntity<String> deleteIssue(@PathVariable long id){**

**Optional<Issue> op = issueRepository.findById(id);**

**if(op.isEmpty()){**

**ResponseEntity.noContent().build();**

**}**

**else{**

**issueRepository.deleteById(id);**

**}**

**return ResponseEntity.ok("deleted");**

**}**

**}**

**🡪Config**

**WebSecurityConfigurer**

**package com.ust.websecurity.config;**

**import org.springframework.context.annotation.Configuration;**

**import org.springframework.http.HttpStatus;**

**import org.springframework.security.authentication.dao.DaoAuthenticationProvider;**

**import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;**

**import org.springframework.security.config.annotation.web.builders.HttpSecurity;**

**import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;**

**import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;**

**import org.springframework.security.web.authentication.HttpStatusEntryPoint;**

**@Configuration**

**@EnableWebSecurity**

**public class WebSecurityConfigurer extends WebSecurityConfigurerAdapter {**

**@Override**

**protected void configure(AuthenticationManagerBuilder auth)throws Exception{**

**auth.authenticationProvider(new DaoAuthenticationProvider());**

**}**

**@Override**

**protected void configure(HttpSecurity http)throws Exception{**

**http.authorizeRequests().antMatchers("/api/v1/issue-book").permitAll().anyRequest().authenticated()**

**.and().exceptionHandling().authenticationEntryPoint(new HttpStatusEntryPoint(HttpStatus.UNAUTHORIZED));**

**}**

**}**

🡪🡪EXCEPTION

UserNotSubscribedException

package com.ust.websecurity.exception;

public class UserNotSubscribedException extends RuntimeException{

public UserNotSubscribedException(String message) {

super(message);

}

}