МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая Кибернетика и Информационные технологии»

Лабораторная работа №6

Выполнил: Студент группы

БВТ2203

Дворянчиков Даниил

Москва

2023

ConcurrentHashMap был разработан как альтернатива Hashtable и обеспечивает всю функциональность, предоставляемой Hashtable, с дополнительными возможностями, называемые уровень одновременности (concurrency level). ConcurrentHashMap позволяет множеству читателей одновременное чтение без использования блокировок. Это достигается разделением Map на различные части, основываясь на «уровне одновременности» и блокированием только части Map при обновлении. По умолчанию, уровень одновременности равен 16, и соответственно Map разделяется на 16 частей и каждая часть управляется отдельной блокировкой. Это означает, что 16 потоков могут работать с Map одновременно, пока они работают с разными частями Map. Это делает ConcurrentHashMap высокопроизводительным, в тоже время не ухудшая потоко-безопасность.

1. Написать программу, которая считывает текстовый файл и выводит на экран топ-10 самых часто встречающихся слов в этом файле. Для решения задачи использовать коллекцию Map, где ключом будет слово, а значением - количество его повторений в файле.
2. Написать обобщенный класс Stack, который реализует стек на основе массива. Класс должен иметь методы push для добавления элемента в стек, pop для удаления элемента из стека и peek для получения верхнего элемента стека без его удаления.
3. Необходимо разработать программу для учета продаж в магазине. Программа должна позволять добавлять проданные товары в коллекцию, выводить список проданных товаров, а также считать общую сумму продаж и наиболее популярный товар. Использовать ConcurrentHashMap и AtomicInteger для счетчика количества продаж каждого товара.

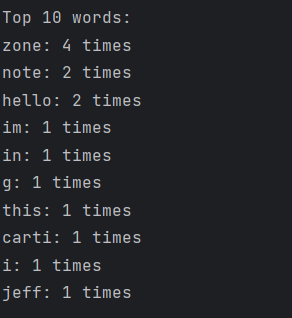
Выполнение:

public static void WordsCount(String filePath)  
{  
 File file = new File(filePath);  
  
 Scanner scanner = null;  
 try {  
 scanner = new Scanner(file);  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 Map<String, Integer> wordCountMap = new HashMap<>();  
  
 while (scanner.hasNext())  
 {  
 String word = scanner.next().toLowerCase();  
 word = word.replaceAll("[^a-zA-Zа-яА-Я]", "");  
 if(wordCountMap.containsKey(word)) wordCountMap.put(word, wordCountMap.get(word) + 1);  
 else wordCountMap.put(word, 1);  
 }  
  
 scanner.close();  
  
 List<Map.Entry<String, Integer>> list = new ArrayList<>(wordCountMap.entrySet());  
  
 Collections.*sort*(list, new Comparator<Map.Entry<String, Integer>>()  
 {  
 @Override  
 public int compare(Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2)  
 {  
 return o2.getValue().compareTo(o1.getValue());  
 }  
 });  
  
 int count = 0;  
 System.*out*.println("Top 10 words:");  
 for (Map.Entry<String, Integer> entry : list)  
 {  
 if (count < 10)  
 {  
 System.*out*.println(entry.getKey() + ": " + entry.getValue() + " times");  
 count++;  
 }  
 else break;  
 }  
}

Протестируем:

hello hello das Ads fill note jeff note head drill g carti put my glasses on i cant see this im in twilight zone zone zone zone

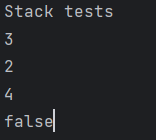
*WordsCount*("C:/Users/dvory/Desktop/итип/lab6/words.txt");



public class Stack<T> {  
 private T[] data;  
 private int size;  
  
 public Stack()  
 {  
 data = (T[]) new Object[10];  
 size = 0;  
 }  
  
 public Stack(int capacity)  
 {  
 data = (T[]) new Object[capacity];  
 size = 0;  
 }  
  
 public void push(T element)  
 {  
 ensureCapacity();  
 data[size] = element;  
 size++;  
 }  
  
 public T pop()  
 {  
 if (isEmpty()) throw new IllegalStateException("Stack is empty");  
 T element = data[size - 1];  
 size--;  
 return element;  
 }  
  
 public T peek()  
 {  
 if (isEmpty()) throw new IllegalStateException("Stack is empty");  
 return data[size - 1];  
 }  
  
 public boolean isEmpty()  
 {  
 return size == 0;  
 }  
  
 private void ensureCapacity()  
 {  
 if (size == data.length)  
 {  
 int newCapacity = data.length \* 2;  
 data = Arrays.*copyOf*(data, newCapacity);  
 }  
 }  
}

Протестируем:

Stack<Integer> stack = new Stack<>(10);  
stack.push(1);  
stack.push(2);  
stack.push(3);  
System.*out*.println(stack.pop());  
System.*out*.println(stack.peek());  
stack.push(4);  
System.*out*.println(stack.pop());  
System.*out*.println(stack.isEmpty());



class SalesManager  
{  
 private ConcurrentHashMap<String, AtomicInteger> salesCount;  
 private ConcurrentHashMap<String, Double> salesAmount;  
  
 public SalesManager()  
 {  
 this.salesCount = new ConcurrentHashMap<>();  
 this.salesAmount = new ConcurrentHashMap<>();  
 }  
  
 public void addSale(String productName, double amount)  
 {  
 // Используем compute методы для атомарных операций над AtomicInteger и Double  
 salesCount.compute(productName, (key, value) -> (value == null) ? new AtomicInteger(1) : new AtomicInteger(value.incrementAndGet()));  
 salesAmount.compute(productName, (key, value) -> (value == null) ? amount : value + amount);  
 }  
  
 public void displaySales()  
 {  
 System.*out*.println("Sales Report------------------");  
 for (String productName : salesCount.keySet())  
 {  
 AtomicInteger count = salesCount.get(productName);  
 Double totalAmount = salesAmount.get(productName);  
 System.*out*.println(productName + ": " + count.get() + " items, Total Amount: $" + totalAmount);  
 }  
 System.*out*.println("------------------------------");  
 }  
  
 public double getTotalSalesAmount()  
 {  
 return salesAmount.values().stream().mapToDouble(Double::doubleValue).sum();  
 }  
  
 public String getMostPopularProduct()  
 {  
 return salesCount.entrySet().stream()  
 .max((entry1, entry2) -> entry1.getValue().get() - entry2.getValue().get())  
 .map(Map.Entry::getKey)  
 .orElse("No sales yet");  
 }  
}

Протестиуем:

SalesManager salesManager = new SalesManager();  
salesManager.displaySales();  
System.*out*.println(salesManager.getTotalSalesAmount());  
salesManager.addSale("banana", 12);  
salesManager.addSale("banana", 12);  
salesManager.addSale("apple", 23);  
salesManager.getTotalSalesAmount();  
salesManager.displaySales();

