پویا امید 14014421005

پیشنهاد کلمه درست با وارد کردن کلمه

وارد کردن کتابخانه‌ها

```
import org.apache.commons.text.similarity.LevenshteinDistance;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;
```

برای محاسبه فاصله بین دو کلمه استفاده میشه LevenshteinDistance

برای کار با فایل‌ها هستن FileNotFoundException و File

تعریف کلاس اصلی

```
public class Main {
```

تعریف متغیرها و سازنده کلاس

```
private Set<String> wordset;

public Main(String[] w) {
wordset = new HashSet<String>(Arrays.asList(w));
}
```

یه مجموعه از کلمات هست که از فایل خونده میشه wordset

یه آرایه از کلمات رو می‌گیره و اون‌ها رو تبدیل به HashSet توی سازنده کلاس می‌کنه

متدی برای دریافت پیشنهادات اصلاح

```
public List<String> getCorrections(String input) {
List<String> corrections = new ArrayList<String>();
if (wordset.contains(input)) {
corrections.add(input);
return corrections;
}
List<WordDistance> wordDistances = new ArrayList<>();
for (String w : wordset) {
int distance = LevenshteinDistance.getDefaultInstance().apply(input, w);
if (distance < 3) {
wordDistances.add(new WordDistance(w, distance));
}
}
Collections.sort(wordDistances, new Comparator<WordDistance>() {
public int compare(WordDistance wd1, WordDistance wd2) {
return Integer.compare(wd1.distance, wd2.distance);
}
});
for (WordDistance wd : wordDistances) {
corrections.add(wd.word);
}
return corrections;
}
```

getCorrections متدی هست که پیشنهادات اصلاح رو برمی‌گردونه

اگه کلمه ورودی توی wordset. باشه، همون کلمه رو برمی‌گردونه

اگه نباشه، فاصله هر کلمه با ورودی رو حساب می‌کنه و اگه کمتر از ۳ بود، به لیست wordDistances اضافه می‌کنه

بعد لیست wordDistances . رو مرتب می‌کنه و کلمات رو به لیست اضافه می‌کنه و برمی‌گردونه

اینجا از Collections.sortاستفاده می‌کنیم که یک تابع برای مرتب‌سازی لیست‌ها است.

این تابع دو ورودی می‌گیرد: لیستی که می‌خواهیم مرتب کنیم (wordDistances) و یک مقایسه‌گر (Comparator) که به تابع می‌گوید چگونه مقایسه و مرتب‌سازی کند.

یک مقایسه‌گر جدید می‌سازیم که وظیفه دارد تعیین کند که دو شیء `WordDistance` چگونه با هم مقایسه شوند.

یک متد `compare`تعریف می‌کنیم که دو شیء `WordDistance` را به عنوان ورودی می‌گیرد و باید تعیین کند کدام یکی کوچکتر، بزرگتر یا برابر است.

`Integer.compare`یک تابع کمکی است که دو عدد صحیح (اینجا فاصله‌ها) را با هم مقایسه می‌کند:

- اگر `wd1.distance`کوچکتر باشد، مقدار منفی برمی‌گرداند.
- اگر `wd1.distance`بزرگتر باشد، مقدار مثبت برمی‌گرداند.
- اگر برابر باشند، صفر برمی‌گرداند.

```
private static class WordDistance {
String word;
int distance;

WordDistance(String word, int distance) {
this.word = word;
this.distance = distance;
}
}
```

.برای نگهداری کلمه و فاصله‌ش از کلمه ورودی استفاده میشه WordDistance

متد اصلی برنامه

```
public static void main(String[] args) {
try {
File f = new File("wordlist.txt");
```

```
Scanner fileScanner = new Scanner(f);
ArrayList<String> file = new ArrayList<String>();
while (fileScanner.hasNext()) {
file.add(fileScanner.nextLine().trim());
}
fileScanner.close();

String[] data = file.toArray(new String[0]);
Main spellChecker = new Main(data);
Scanner inputScanner = new Scanner(System.in);

while (true) {
System.out.print("Enter the word (or type 'exit' to quit): ");
String word = inputScanner.nextLine().trim();
if (word.equalsIgnoreCase("exit")) {
break;
}
List<String> corrections = spellChecker.getCorrections(word);
if (corrections.isEmpty()) {
System.out.println("No suggestions found.");
} else {
System.out.println("The most likely answer: " + corrections.get(0));
if (corrections.size() > 1) {
System.out.println("Other suggestions:");
for (int i = 1; i < corrections.size(); i++) {
System.out.println(corrections.get(i));
}
}
}
```

```
}
inputScanner.close();
} catch (FileNotFoundException e) {
System.out.println("The word list file was not found.");
}
}
```

در این بخش فایل `wordlist.txt`را باز می‌کنیم و با استفاده از اسکنر (`Scanner`) آن را خط به خط می‌خوانیم و کلمات را در یک لیست (`ArrayList`) ذخیره می‌کنیم.

لیست کلمات را به آرایه تبدیل می‌کنیم و سپس یک شیء از کلاس `Main` با استفاده از این آرایه می‌سازیم.

یک حلقه بی‌نهایت داریم که از کاربر کلمه می‌گیرد

اگر کاربر `exit` را وارد کند، حلقه و برنامه خاتمه پیدا می‌کند.

برای کلمه وارد شده، پیشنهادات اصلاح را با استفاده از متد `getCorrections` دریافت می‌کنیم.
اگر پیشنهادی پیدا نشود، پیغام "No suggestions found" نمایش داده می‌شود.

اگر پیشنهاداتی وجود داشته باشد، اولین پیشنهاد به عنوان محتمل‌ترین اصلاح نمایش داده می‌شود و دیگر پیشنهادات نیز اگر وجود داشته باشند، نمایش داده می‌شوند.

خلاصه

برنامه یک فایل کلمات رو می‌خونه

از کاربر یک کلمه می‌گیره

فاصله هر کلمه از کلمات موجود با کلمه ورودی رو حساب می‌کنه

کلماتی که فاصله کمی دارن رو به عنوان پیشنهاد اصلاح نشون می‌ده