/\*  
Marla Peraza Ravelo, COP-3330C-17193, 10/6/2024  
  
The objective of this program is to demonstrate the use of Collections framework and the  
Collection interface. The program prompts the user to enter elements to create a list,  
then it performs a variety of operations with it, which include printing the list, sorting it  
alphabetically and using a custom sort, shuffling it, and other operations.  
  
The user is going to enter 8 unique words (cannot be duplicate) at the beginning. The entered  
words are going to be printed out, along with the total number of entries, the longest entry, and the  
shortest one. Then the list is going to be alphabetically sorted and printed out. After that, it is  
going to appear sorted by length (from longest to shortest). Next, it's elements are going to be  
shuffled. After that, the user will be prompted to enter a word to search. If the word exists inside  
the list, its position is going to be shown, and if not, there will be a not found text. Next the list  
is going to be reversed and rotated, and converted into an array and back into a list again.  
 \*/  
  
  
import java.util.ArrayList; //import to use ArrayList  
import java.util.Collections; //import to use Collections  
import java.util.Scanner; //import to use Scanner  
  
//Main class where the use of Collections is going to be demonstrated  
public class CollectionsInJava {  
  
 public static void main(String[] args) {  
 //Creating an ArrayList  
 ArrayList<String> myList = new ArrayList<>();  
  
 //Creating a Scanner object  
 Scanner scanner = new Scanner(System.*in*);  
  
 // 1- Prompt user to input 8 unique words to make a list  
 System.*out*.println("Enter 8 unique words. Do not enter duplicates, ignoring case. ");  
  
 //while loop to repeat the collection of input until myList has 8 entries  
 while (myList.size() < 8) {  
 String input = scanner.nextLine();  
  
 //if statement that checks whether the inputs are repeated using a method  
 if (!*checkForWord*(myList, input)) {  
 myList.add(input);  
 } else {  
 System.*out*.println("You entered a duplicate word, try again.");  
 }  
 }  
  
 //2: Print the list with an enhanced for loop  
 System.*out*.println("\nThis is the List entered:");  
 for (String s : myList) {  
 System.*out*.println(s);  
 }  
  
 // 3: Find total number, longest and shortest entries  
 //Find the longest entry  
 String longest = myList.get(0); // Start with the first string  
 for (String str : myList) {  
 if (str.length() > longest.length()) {  
 longest = str; // Update longest if a longer string is found  
 }  
 }  
  
 //Find the shortest entry  
 String shortest = myList.get(0); // Start with the first string  
 for (String str : myList) {  
 if (str.length() < shortest.length()) {  
 shortest = str; // Update shortest if a shortest string is found  
 }  
 }  
 //Print total entries, longest and shortest  
 System.*out*.println("Total Number of Entries: " + myList.size());  
 System.*out*.println("This is the longest entry: " + longest);  
 System.*out*.println("This is the shortest entry: " + shortest);  
  
 // 4: Sort the list alphabetically and print using enhanced for loop  
 Collections.*sort*(myList);  
 System.*out*.println("\nNow the list is sorted alphabetically:");  
 for (String s : myList) {  
 System.*out*.println(s);  
 }  
  
 // 5: Custom sort of the list by length (Longest to shortest)  
 myList.sort((a, b) -> b.length() - a.length());  
 System.*out*.println("\nThis is the list sorted by length (longest to shortest):");  
 for (String word : myList) {  
 System.*out*.println(word);  
 }  
  
 //6: Shuffle the list  
 Collections.*shuffle*(myList);  
 System.*out*.println("\nNow the list has been shuffled:");  
 for (String word : myList) {  
 System.*out*.println(word);  
 }  
  
 //7: Search for a word in the list  
 System.*out*.println("\nEnter a word that you want to search for inside the list:");  
 String searchWord = scanner.nextLine();  
 int index = *findIndexOfWord*(myList, searchWord);  
 if (index != -1) {  
 System.*out*.println("Word found at position: " + (index + 1));  
 } else {  
 System.*out*.println("Word not found.");  
 }  
  
 //8: Demonstrating other methods from Collections  
  
 // Reverse the list, prints out the list in reverse order using the reverse method from Collections  
 Collections.*reverse*(myList);  
 System.*out*.println("\nThis is the reversed list:");  
 //enhanced for loop  
 for (String s : myList) {  
 System.*out*.println(s);  
 }  
  
 // Rotate the list by two positions using the rotate method from Collections  
 Collections.*rotate*(myList, 2);  
 System.*out*.println("\nThis is the rotated list:");  
 //enhanced for loop  
 for (String s : myList) {  
 System.*out*.println(s);  
 }  
  
 // 9: Convert the list to an array  
 String[] arr = myList.toArray(new String[0]);  
 System.*out*.println("\nThe List has been converted to an Array:");  
 for (String s : arr) {  
 System.*out*.println(s);  
 }  
  
 // Step 10: Convert the array back into a list  
 ArrayList<String> arrayToList = new ArrayList<>();  
 //use addAll method from Collections  
 Collections.*addAll*(arrayToList, arr);  
 System.*out*.println("\nList Converted from Array:");  
 for (String s : arrayToList) {  
 System.*out*.println(s);  
 }  
  
 }  
  
 // Method to check if a word exists inside a list without case sensitivity  
 private static boolean checkForWord(ArrayList<String> list, String word) {  
 for (String entry : list) {  
 //The equalsIgnoreCase() method is used to compare two strings for equality,  
 // ignoring case differences. It returns true if the strings are equal,  
 // and false otherwise.  
 if (entry.equalsIgnoreCase(word)) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 // Method to find the index of a word inside the list  
 private static int findIndexOfWord(ArrayList<String> list, String word) {  
 for (int i = 0; i < list.size(); i++) {  
 if (list.get(i).equalsIgnoreCase(word)) {  
 return i;  
 }  
 }  
 return -1;  
 }  
}  
  
/\*  
 Differences between Collections and Collection:  
 - In Java, Collection is an interface that represents a group of objects, known as elements.  
 It provides basic operations like adding, removing, and querying elements, and is implemented by various  
 data structures such as ArrayList, HashSet, and LinkedList.  
 - Collections is a utility class in Java that provides static methods to operate on collections,  
 like sorting, reversing, and searching. It helps manipulate and perform common operations on  
 collection objects, like ArrayList, Set, etc.  
  
 Differences between0 class and interface:  
 - In Java, a class is like a blueprint from which objects are created. It can have attributes,  
 methods, and constructors. A class can provide concrete implementations of its methods.  
 - An Interface in Java is a reference type that can only contain abstract methods and static constants.  
 It cannot have any method implementation, and it allows classes to share common behaviors  
 by implementing the methods defined inside it.  
  
 Source: Oracle Java Documentation (https://docs.oracle.com/javase/8/docs/api/)  
\*/