Ejercicios: Colecciones.

1. Comparar los tiempos de ejecución de las distintas formas de recorrer una List. En concreto, habrá que medir el tiempo que se tarda en recorrer un ArrayList y un LinkedList del mismo tamaño, siguiendo distintos procedimientos y con distinto número de elementos:
    - Bucle que accede a cada elemento por su posición.
    - Bucle for-each
    - Iterator

    Completar la siguiente tabla, observar los resultados obtenidos y responder a las preguntas:

	Tamaño x	Tamaño 2 * x
ArrayList con for		
ArrayList con foreach		
ArrayList con iterator		
LinkedList con for		
LinkedList con foreach		
LinkedList con iterator		

    a) ¿Hay diferencias significativas entre el uso de for, foreach e iterator?
    b) ¿Los tres tipos de recorrido funcionan igual de bien/mal en las dos clases? ¿o hay diferencias entre ArrayList y LinkedList?
    c) Justificar las razones teniendo en cuenta cómo están implementadas internamente

2. (QuitarElementos) Implementar un método que reciba como parámetro un Collection<String> y un String prefijo. El método debe eliminar todos los elementos de la colección que comiencen por el prefijo indicado. Usar el método main para probar el método.

3. (OperacionesMasivas) Implementar los siguientes métodos que se harán apoyándose en clases del API Collections de Java. Probar los métodos desde el main
    - int[] quitarDuplicados (int v[]), que dado un array de enteros devuelva otro array con los mismos valores que el original pero sin duplicados. Sugerencia: Volcar v a un Set y devolver un array con el contenido del Set
    - int[] unión(int v1[], int v2[]), que dados dos arrays v1 y v2 devuelva otro array con los elementos que están en v1 o que están en v2, sin que ningún elemento se repita. Sugerencia: Volcar los dos arrays a un mismo Set y devolver un array con el contenido del Set
    - int[] intersección(int v1[], int v2[]), que dados dos arrays v1 y v2 devuelva otro array con los elementos que aparecen en ambos arrays. Cada elemento común aparecerá una sola vez en el resultado. Sugerencia: Volcar los arrays a dos Set y

usar retainAll para mantener los comunes. Devolver un array con el contenido del Set.

- int[] diferencia (int v1[], int v2[]), que dados dos arrays v1 y v2 devuelva otro array con los elementos que de v1 que no están en v2. En caso de haber elementos duplicados en v1 estos se mantendrán en el resultado. Sugerencia: Volcar los arrays a dos List. Usar removeAll para eliminar los elementos comunes. Devolver un array con el resultado.

4. (Futbol) Para gestionar una liguilla de futbol disponemos de:
   - Un array con los nombres de los equipos participantes (String[] equipos)
   - Una matriz con los partidos disputados (String[][] partidosDisputados). Tiene dos columnas, la primera con el nombre del equipo local y la segunda con el visitante

   Escribir un método que, dados esos dos parámetros, devuelva un array con el nombre de los equipos que ya han jugado en casa (han sido locales) en algún partido. Ayudarse de algún tipo de colección.

   Escribir un método que, dados esos dos parámetros, devuelva un array con el nombre de los equipos que nunca hay jugado fuera de casa (como visitantes). Ayudarse de algún tipo de colección.

5. (PalabrasTexto) Escribe un programa lea un fichero de texto cuyo nombre indique el usuario y muestre qué palabras contiene. Supondremos que el fichero no contiene signos de puntuación (puntos, comas, etc). Ayudarse de un Set

6. (ContarPalabras) Escribe un programa que abra el fichero de texto que indique el usuario y muestre cuántas veces se repite cada palabra que contiene. Supondremos que el fichero no contiene signos de puntuación (puntos, comas, etc.) Ayudarse de un Map.

7. (TraductorSimple) Escribir un programa que solicite al usuario una frase y la muestre traducida a inglés, palabra a palabra. Para ello, se dispone de un fichero palabras.txt que contiene parejas (palabra en español, palabra en inglés) separadas por un tabulador. Cada pareja se encuentra en una línea del fichero. El proceso será el siguiente:

   - Leer el fichero y cargar sus datos en una estructura en un Map.

   - Solicitar al usuario una frase. Traducir, usando la estructura de datos anterior, cada palabra de la frase y formar con ellas la frase traducida.

   - Mostrar la frase traducida al usuario.

8. (Diccionario) Disponemos de un diccionario español – inglés, en forma de Map<String,Set<String>>, que contiene parejas formadas por (palabra en castellano, conjunto de traducciones a inglés). Implementar los siguientes métodos:

   - anyadirTraduccion(Map <String, Set <String>> diccionario ,String cast, String ingl): Añade la pareja (cast, ingl) al diccionario de forma que:

- Si la palabra cast no estaba en el diccionario la añade, junto con su traducción a inglés.
- Si la palabra cast estaba ya en el diccionario pero no aparecía como traducción la palabra *ing,* añade *ing* a su conjunto de traducciones.
- Si la palabra cast estaba y la traducción *ing* también, no se realizarán cambios.
- El método devuelve true si se han realizado cambios en el diccionario y false en caso contrario.

- quitarTraduccion (Map <String, Set <String>> diccionario ,String cast, String ingl): Quita la traducción *ingl* a la palabra *cast.* Si la palabra en castellano se queda sin traducciones, se elimina del diccionario. Si se han producido cambios se devuelve true y en caso contrario false.
- traduccionesDe(Map <String, Set <String>> diccionario, String cast): Devuelve una colección con las traducciones de la palabra indicada o null si la palabra no está en el diccionario.

9. En un partido de baloncesto, cada vez que un jugador comete una falta hemos añadido su nombre a un List<String>. La lista contiene, por tanto, quienes han cometido falta y el orden en que se han cometido. Mostrar los nombres de los jugadores que han sido expulsados en el orden en que han sido expulsados. Se expulsa a un jugador que comete 5 faltas