Estructuras de Datos

Tarea Iteradores y Comparadores

Los siguientes métodos deben ser implementados utilizando Iteradores y Comparadores. Cuando no se especifique, usted es libre de decidir los parámetros y tipos de dato de retorno de los métodos indicados. Cuando lo considere pertinente, incluya comentarios en sus respuestas para indicar cualquier consideración especial en cuanto a sus soluciones. Así mismo, usted puede decir el iterable o comparator que crea pertinente.

1. **public int sumArray(ArrayIterable iterable):** Suma los elementos de un arreglo.

2. **public ArrayIterable filterGreaterThan(ArrayIterable iterable, int threshold):** filtra los elementos de un arreglo para incluir sólo los elementos mayores a un valor dado (threshold).

3. **public void sortList(List<Integer> list, Comparator<Integer> comparator):** Ordena una lista.

4. **public boolean isSorted(List<Integer> list, Comparator<Integer> comparator):** Comprueba si una lista está ordenada según el comparator que se le defina.

5. **public ArrayIterable reverseIterable(ArrayIterable iterable):** Itera una lista en el sentido contrario. Notar que devuelve otro iterable, pues si aplico reverseIterable sobre el resultado, debe obtener el iterable original.

6. **public Integer findMin(List<Integer> list, Comparator<Integer> comparator):** Encuentra el mínimo de una lista. La comparación para saber si un elemento es menor que otro se debe hacer a través del comparator.

7. **public Integer findMax(List<Integer> list, Comparator<Integer> comparator):** Encuentra el máximo de una lista. La comparación para saber si un elemento es menor que otro se debe hacer a través del comparator.

8. **public List<Integer> filterEvenNumbers(Iterable<Integer> iterable):** Filtra los números pares de un Iterable. Observar que devuelve una Lista.

9. **public double calculateAverage(Iterable<Integer> iterable):** Calcula el promedio de los elementos de un Iterable.

10. **public int binarySearch(List<Person> list, Person target, Comparator<Person> comparator):** En base a una lista de objetos Person (pueden definir la clase como prefieran), el método busca si una Persona se encuentra en la lista de personas.