TP3 PROGRAMACIÓN

1. ¿Qué es una relación involutiva en java?

   En Java, una relación involutiva es un tipo de relación entre clases donde una clase A contiene una referencia a otra clase B, y a su vez, la clase B contiene una referencia de vuelta a la clase A. Esto significa que ambas clases están conectadas mutuamente.

2. De un ejemplo de relación involutiva.

   Un ejemplo común de una relación involutiva es la relación entre dos clases como Person y Address. Supongamos que cada persona tiene una dirección asociada y cada dirección puede estar asociada a una o más personas. En este caso, la clase Person tendría una referencia a la clase Address, y la clase Address tendría una referencia a una lista o conjunto de objetos Person.

3. ¿Qué métodos comunes se encuentran en la interfaz Collection y qué funcionalidades proporcionan?

   Métodos para Agregar o Eliminar elementos:
   -add(E e): Agrega un elemento a la colección.
   -addAll(Collection<? extends E> c): Agrega todos los elementos de otra colección a esta colección.
   -remove(Object o): Elimina la primera aparición del elemento especificado de la colección.
   -removeAll(Collection<?> c): Elimina todos los elementos de la colección que están presentes en la colección especificada.

   Métodos para Consultar o Verificar elementos:
   -contains(Object o): Devuelve true si la colección contiene el elemento especificado.
   -containsAll(Collection<?> c): Devuelve true si la colección contiene todos los elementos de la colección especificada.
   -Empty(): Devuelve true si la colección no contiene elementos.
   -size(): Devuelve el número de elementos en la colección.

   Iteración sobre elementos
   -iterator(): Devuelve un iterador sobre los elementos en la colección.

   Operaciones en masa:
   -clear(): Elimina todos los elementos de la colección.
   -retainAll(Collection<?> c): Retiene solo los elementos en la colección que están presentes en la colección especificada.

   Convertir a array:

-toArray(): Devuelve un array que contiene todos los elementos de la colección.

Estos son algunos de los métodos más comunes de la interfaz Collection.

4. Explica el propósito de la interfaz Iterable en Java y cómo se utiliza.

   La interfaz Iterable en Java proporciona una manera de permitir que los objetos sean iterables, es decir, que puedan ser recorridos en un bucle for-each (también conocido como bucle for mejorado) u otras construcciones que requieren un conjunto de elementos.

   El propósito principal de la interfaz Iterable es proporcionar un mecanismo estándar para iterar sobre los elementos de un objeto, independientemente de su implementación específica. Esto permite que los objetos sean utilizados de manera uniforme en contextos donde la iteración es necesaria, como en bucles for-each, operaciones de streaming, y otros algoritmos de procesamiento de colecciones.

   Para implementar la interfaz Iterable, una clase debe proporcionar una implementación del método iterator(), que devuelve un objeto Iterator. Un Iterator es responsable de recorrer los elementos de la colección y proporcionar métodos para acceder a esos elementos.

5. ¿Qué ventajas ofrece el uso de la interfaz Iterable en comparación con simplemente iterar sobre una colección utilizando un bucle for estándar?

   - Sintaxis Simplificada: Utilizar un bucle for-each es más cómodo el poder controlar y ver las iteraciones que hace el array sin muchas complicaciones
   - Flexibilidad: Al devolver un iterador, una clase puede proporcionar iteraciones personalizadas, filtradas o transformadas según sea necesario o como lo requiera el diseñador.
   - Encapsulamiento: Permite a las clases controlar cómo se itera sobre ellas sin exponer la estructura de una esta en sí.