Trabajo practico N°3

1. Una relación involutiva en Java se refiere a una relación entre dos clases en la que ambas clases están fuertemente acopladas entre sí. Esto significa que cada clase depende directamente de la otra, lo que puede llevar a un diseño poco flexible y difícil de mantener.

2. Un ejemplo de una relación involutiva podría ser una clase “Cliente” y una clase “CuentaBancaria”. Si la clase Cliente tiene una referencia a la clase “CuentaBancaria” y viceversa, esto crearía una relación involutiva. Por ejemplo:

```
public class Cliente {
   private CuentaBancaria cuenta;

   public Cliente() {
      cuenta = new CuentaBancaria(this);
   }
}

public class CuentaBancaria {
   private Cliente propietario;

   public CuentaBancaria(Cliente propietario) {
      this.propietario = propietario;
   }
}
```

   En este ejemplo, la clase “Cliente” tiene una referencia a “CuentaBancaria”, y la clase “CuentaBancaria” tiene una referencia al “Cliente”, lo que crea una relación involutiva.

3. La interfaz “Collection” en Java define varios métodos comunes que proporcionan funcionalidades para trabajar con colecciones de elementos. Algunos de los métodos más comunes son:
   - add(): Agrega un elemento a la colección.
   - remove(): Elimina la primera ocurrencia del elemento especificado de la colección, si está presente.
   - contains(): Devuelve true si la colección contiene el elemento especificado.
   - isEmpty(): Devuelve true si la colección no contiene elementos.
   - size(): Devuelve el número de elementos en la colección.
   - iterator(): Devuelve un iterador sobre los elementos de la colección.

4. La interfaz Iterable en Java se utiliza para permitir que una clase sea iterable, lo que significa que sus instancias pueden ser recorridas usando un bucle for-each o un iterador. La interfaz Iterable declara un único método llamado iterator(), que devuelve un iterador sobre los elementos de la instancia de la clase que implementa la interfaz.

5. El uso de la interfaz Iterable en Java ofrece varias ventajas sobre simplemente iterar sobre una colección utilizando un bucle for estándar:
   - Permite el uso del bucle for-each, que es más legible y conciso que un bucle for estándar.
   - Abstrae el proceso de iteración, lo que permite a la clase que implementa Iterable cambiar la forma en que se itera sobre sus elementos sin afectar al código cliente.
   - Facilita la integración con API de Java que esperan objetos iterables, como los métodos Collections.addAll() o Arrays.asList().
   - Fomenta el principio de diseño de la programación orientada a objetos, ya que la clase puede proporcionar su propio iterador para encapsular la estructura de datos subyacente.