**PROGRAMACIÓN 2: TRABAJO PRÁCTICO 3**

1. Es cuando un objeto está asociado consigo mismo mediante algún tipo de operación o función.
2. Por ejemplo mirarse en un espejo, cuando uno se mira al espejo ve su propia imagen reflejada de vuelta. No importa quien se pare frente, siempre verá su propia imagen reflejada: el resultado de la operación (mirarse en el espejo) sobre un objeto (una persona) es el mismo objeto reflejado.

* **‘add(E e)´**: Agrega un elemento a la colección si es posible.
* **‘addAll(Collection<? Extends E> c)’**: Agrega todos los elementos de la colección especificada a esta colección si son posibles.
* **‘clear()’**: Elimina todos los elementos de la colección.
* **‘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.
* **‘isEmpty()’**: Devuelve ‘true’ si la colección no contiene elementos.
* **‘iterator()’**: Devuelve un iterador sobre los elementos de la colección.
* **‘remove(Object o)’**: Elimina la primera instancia del elemento especificado de la colección, si está presente.
* **‘removeAll(Collection<?> c)’**: Elimina todos los elementos de la colección que estás contenidos en la colección especificada.
* **‘size()’**: Devuelve el número de elementos en la colección.
* **‘toArray()’**: Devuelve un array que contiene todos los elementos de la colección.
* **‘toArray(T[] a)’**: Devuelve un array que contiene todos los elementos de la colección; el tipo de tiempo de ejecución del array devuelto es el del array especificado.

1. La interfaz **‘Iterable’** se utiliza para permitir que una clase sea capaz de proporcionar un iterador sobre sus elementos. Para utilizarla primero hay que implementarla añadiendo **‘<T>’** a la definición de la clase y utilizando el método **‘iterator()’**. Luego, dentro del método, hay que crear y devolver un objeto que implemente la interfaz **‘iterator’**.
2. La interfaz ‘Iterable’ proporciona una abstracción de la iteración, lo que permite iterar sobre una colección sin preocuparse por los detalles de como se implementa internamente la iteración facilitando el entendimiento del código; también permite que las clases donde se implementa sean compatibles con los constructores y métodos que acepten objetos iterables; permite que los objetos se utilicen en bucles for-each (en lugar de tener que escribir un bucle for estándar con un índice de bucle, se puede iterar sobre los elementos de la colección directamente), lo que hace que el código sea más legible.