OTRAS OPERACIONES

COLECCIONES

OpenWebinars

MÉTODOS EN COLLECTIONS

- Esta clase provee de decenas de métodos estáticos para realizar diferentes operaciones con colecciones.
  - Obtener colecciones “especiales”
  - Buscar
  - Modificar
  - Descriptores estadísticos

OBTENER COLECCIONES ESPECIALES

- emptyXXXX
  - Devuelve una colección vacía e inmutable.
  - Útil si un método debe devolver una colección vacía en lugar de *null*.
  - Disponible para las interfaces: *List*, *Map*, *Set*, *NavigableMap*, *NavigableSet*, *SortedMap*, *SortedSet*.

OBTENER COLECCIONES ESPECIALES

- singletonXXX
  - Devuelve una colección inmutable con un único elemento.
  - Similar a usar el método *of* con un único argumento
  - Disponible para las interfaces: *List*, *Map*, *Set.*

OBTENER COLECCIONES ESPECIALES

- unmodifiableXXXX
  - Devuelve una vista inmutable de una colección
  - Útil si un método debe devolver una colección que no se puede modificar
  - Disponible para las interfaces: *Collection*, *List*, *Map*, *Set*, *NavigableMap*, *NavigableSet*, *SortedMap*, *SortedSet*.
  - Similar a *of*, pero éste devuelve una colección y no un vista.

OBTENER COLECCIONES ESPECIALES

- synchronizedXXXX
  - Devuelve una vista sincronizada de una colección
  - Útil si se va a trabajar en un entorno multi-hilo
  - Disponible para las interfaces: *Collection*, *List*, *Map*, *Set*, *NavigableMap*, *NavigableSet*, *SortedMap*, *SortedSet*.
  - Es responsabilidad del programador que las operaciones se realicen coherentemente: *synchronized(collection) { ... }*

OBTENER COLECCIONES ESPECIALES

- checkedXXXX
  - Devuelve una vista que asegura el tipo de dato de los elementos almacenados.
  - Si tratamos de insertar un dato de tipo incorrecto, provoca *ClassCastException*.
  - Útil para depurar errores.

BUSCAR

- binarySearch
  - Algoritmo de búsqueda binaria
  - Obliga a que los elementos de la lista estén ordenados ascendentemente.
  - Devuelve la posición o un número negativo.
  - Disponible también en la clase *Arrays*
- Se proporciona una segunda versión del método que recibe un *Comparator*.

BUSCAR

- *indexOf*, *lastIndexOf* están disponibles en List para buscar un elemento.
- *Collections.indexOfSubList*, *lastIndexOfSubList*
  - Buscan una lista dentro de otra
  - Si la encuentran, devuelven la posición de la primera / última ocurrencia.

MODIFICAR

- *shuffle(List)*
  - Desordena una lista aleatoriamente
- *swap(List, i, j)*
  - Permuta los elementos de la posiciones especificadas
- *replaceAll(List, old, new)*
  - Reemplaza todas las ocurrencias de *old* por *new*.

MODIFICAR

- *nCopies(n, obj)*
  - Devuelve una lista que tiene *n* copias del objeto *obj*.
- *fill(List, obj)*
  - Reemplaza todos los elementos de la lista especificada por *obj*.
- *copy(List, List)*
  - Copia los elementos de una lista a otra. En la lista de destino se copian en los mismos índices que en la lista de origen. Error si la lista de destino es de menor tamaño (*size*)

MODIFICAR

- *addAll(Collection, vararg)*
  - Añade una serie de elementos a una colección.
  - Es cómodo si se quieren añadir unos pocos elementos.

DESCRIPTORES ESTADÍSTICOS

- *max*, *min*
  - Trabaja con Collection.
  - Devuelve el elemento máximo (o mínimo) según el orden natural.
  - Otra implementación proporcionando un *Comparator*
- *frequency*
  - Nº de ocurrencias de un elemento en una lista.