Introducción a los genéricos en las colecciones de Java

Introducción

Las colecciones al ser formadas de objetos de tipo Object, tienen una serie de problemas que hasta la versión 1.5 fueron sufridas por los programadores/as:

- Nada impide crear listas heterogéneas de objetos (por ejemplo colecciones de objetos String junto con Arrays de enteros). Eso causa problemas evidentes de casting, además de posibles incoherencias.
- El que las listas sean siempre de tipo Object, provocan una cantidad grande de conversiones en el código (mediante casting), haciéndole más pesado de comprender.
- Los métodos de las clases e interfaces de las colecciones no se adaptan al tipo de datos que contienen dificultando su uso.

Por ello aparecieron los tipos genéricos en la versión 1.5 de Java consiguiendo solucionar los problemas anteriores en las colecciones. Así la interfaz Collection ahora se llama Collection <E> porque utiliza genéricos y sus métodos también les utiliza.

Lo que varía en definitiva es que al indicar ahora colecciones e iteradores del tipo que sean, hay que indicar también la traducción de sus genéricos. Dicho de otra forma, hay que indicar el tipo de lista o de iterador que estamos creando. Por ejemplo:

```
ArrayList<String> lista = new ArrayList<String>();

ListIterator<String> iterador = lista.listIterator();
```

A continuación, se enumeran los metódos de Collection <E> Hay que tener en cuenta, según lo comentado anteriormente que E es el tipo genérico de la colección

Método	Uso
boolean add(E o)	Añade el objeto a la colección. Devuelve true si se pudo completar la operación. Si no cambió la colección como resultado de la operación devuelve false
boolean remove(E o)	Elimina al objeto indicado de la colección.
int size()	Devuelve el número de objetos almacenados en la colección
boolean isEmpty()	Indica si la colección está vacía
boolean contains(Object o)	Devuelve true si la colección contiene al objeto indicado
void clear()	Elimina todos los elementos de la colección
boolean addAll(Collection E otra)	Añade todos los elementos de la colección otra a la colección actual. Sólo puede añadir objetos de colecciones cuyos elementos sean de tipos compatibles con el actual.
boolean removeAll(Collection E otra)	Elimina todos los objetos de la colección actual que estén en la colección otra
boolean retainAll(Collection E otra)	Elimina todos los elementos de la colección que no estén en la otra

boolean containsAll(Collection E otra)	Indica si la colección contiene todos los elementos de otra
Object[] toArray()	Convierte la colección en un array de objetos.
T[] toArray(T array)	Convierte la colección en un array de objetos. El array devuelto contiene todos los elementos de la colección, el array de devolución es del mismo tipo que el array que recibe de argumento (de hecho es la única utilidad que tiene este argumento, la de decir el tipo de array que se ha de devolver)
Iterator iterator()	Obtiene el objeto iterador de la colección, se explica en el punto siguiente

Ejemplo uso de recorrido de lista usando genéricos:

```
ArrayList <String> lista=new ArrayList<String>();
lista.add("Hola");
lista.add("Adiós");
lista.add("Hasta luego");
lista.add("Ciao");
Iterator<String>it=lista.iterator();
while(it.hasNext()){
   String s=it.next();
   System.out.println(s);
}
```

Se observa la ausencia de castings en el código