9.F. Iteradores.

2. Iteradores (II).

¿Qué inconvenientes tiene usar los iteradores sin especificar el tipo de objeto? En el siguiente ejemplo, se genera una lista con los números del 0 al 10. De la lista, se eliminan aquellos que son pares y solo se dejan los impares. En el ejemplo de la izquierda se especifica el tipo de objeto del iterador, en el ejemplo de la derecha no, observa el uso de la conversión de tipos en la línea 6.

Comparación de usos de los iteradores, con o sin conversión de tipos.

Ejemplo indicando el tipo de objeto de iterador	Ejemplo no indicando el tipo de objeto del iterador
`ArrayList <Integer> lista=new ArrayList<Integer>();`	`ArrayList <Integer> lista=new ArrayList<Integer>();`
`for (int i=0;i<10;i++) lista.add(i);`	`for (int i=0;i<10;i++) lista.add(i);`
`Iterator<Integer> it=lista.iterator();`	`Iterator it=lista.iterator();`
`while (it.hasNext()) {`	`while (it.hasNext()) {`
`Integer t=it.next();`	`Integer t=(Integer)it.next();`
`if (t%2==0) it.remove();`	`if (t%2==0) it.remove();`
`}`	`}`

Un iterador es seguro porque está pensado para no sobrepasar los límites de la colección, ocultando operaciones más complicadas que pueden repercutir en errores de software. Pero realmente se convierte en inseguro cuando es necesario hacer la operación de conversión de tipos. Si la colección no contiene los objetos esperados, al intentar hacer la conversión, saltará una incómoda excepción. Usar genéricos aporta grandes ventajas, pero usándolos adecuadamente.

Para recorrer los mapas con iteradores, hay que hacer un pequeño truco. Usamos el método entrySet que ofrecen los mapas para generar un conjunto con las entradas (pares de llave-valor), o bien, el método keySet para generar un conjunto con las llaves existentes en el mapa. Veamos cómo sería para el segundo caso, el más sencillo:

```
HashMap<Integer,Integer> mapa=new HashMap<Integer,Integer>test();

for (int i=0;i<10;i++) mapa.put(i, i); // Insertamos datos de prueba en el mapa.

for (Integer llave:mapa.keySet()) // Recorremos el conjunto generado por keySet, contendrá las llaves.

{

        Integer valor=mapa.get(llave); //Para cada llave, accedemos a su valor si es necesario.

}
```

Lo único que tienes que tener en cuenta es que el conjunto generado por keySet no tendrá obviamente el método `add` para añadir elementos al mismo, dado que eso tendrás que hacerlo a través del mapa.

Recomendación

Si usas iteradores, y piensas eliminar elementos de la colección (e incluso de un mapa), debes usar el método `remove` del iterador y no el de la colección. Si eliminas los elementos utilizando el método `remove` de la colección, mientras estás dentro de un bucle de iteración, o dentro de un bucle for-each, los fallos que pueden producirse en tu programa son impredecibles. ¿Logras adivinar porqué se pueden producir dichos problemas?

Los problemas son debidos a que el método `remove` del iterador elimina el elemento de dos sitios: de la colección y del iterador en sí (que mantiene interiormente información del orden de los elementos). Si usas el método `remove` de la colección, la información solo se elimina de un lugar, de la colección.

Autoevaluación

¿Cuándo debemos invocar el método `remove()` de los iteradores?

- ○ En cualquier momento.
- ○ Después de invocar el método next().
- ○ Después de invocar el método hasNext().
- ○ No es conveniente usar este método para eliminar elementos, es mejor usar el de la colección.