APRENDERAPROGRAMAR.COM

TIPO Y MÉTODO ITERATOR. ERROR JAVA.UTIL.CONCURRENT MODIFICATIONEXCEPTION. RESUMEN DE TIPOS DE BUCLES EN JAVA. (CU00667B)

Sección: Cursos
Categoría: Curso “Aprender programación Java desde cero”

Fecha revisión: 2029

Resumen: Entrega nº67 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

TIPO ITERATOR Y MÉTODO ITERATOR EN JAVA

El uso del bucle for-each tiene algunos inconvenientes. Uno de ellos, que para recorrer la colección nos basamos en la propia colección y por tanto no podemos (o al menos no debemos) manipularla durante su recorrido. Supongamos que vamos recorriendo una lista de 20 objetos y que durante el recorrido borramos 5 de ellos. Probablemente nos saltará un error porque Java no sabrá qué hacer ante esta modificación concurrente.

Sería como seguir un camino marcado sobre unas baldosas y que durante el camino nos movieran las baldosas de sitio: no podríamos seguir el camino previsto. Este tipo de problemas se suelen presentar con un mensaje de error del tipo *java.util.ConcurrentModificationException*.

El uso de operaciones sobre colecciones usando un for tradicional también puede dar lugar a resultados no deseados. En general, operar sobre una colección al mismo tiempo que la recorremos puede ser problemático. Este problema queda salvado mediante un recurso del API de Java: los objetos tipo Iterator. **Un objeto de tipo Iterator funciona a modo de copia para recorrer una colección,** es decir, el recorrido no se basa en la colección "real" sino en una copia. De este modo, al mismo tiempo que hacemos el recorrido (sustentado en la copia) podemos manipular la colección real, añadiendo, eliminando o modificando elementos de la misma. Para poder usar objetos de tipo Iterator hemos de declarar en cabecera *import java.util.Iterator;*. La sintaxis es la siguiente:

```
Iterator <TipoARecorrer> it = nombreDeLaColección.iterator ();
```

Esta sintaxis resulta un tanto confusa. Por ello vamos a tratar de analizarla con detenimiento. En primer lugar, declaramos un objeto de tipo Iterator. La sentencia para ello es Iterator <TipoARecorrer> it; donde it es el nombre del objeto que estamos declarando. El siguiente paso es inicializar la variable para que efectivamente pase a contener un objeto. Pero Iterator carece de constructor, así que no podemos usar una sentencia de tipo *new Iterator<TipoARecorrer>();* porque no resulta válida. Iterator es una construcción un tanto especial. De momento nos quedamos con la idea de que es algo parecido a una clase. Dado que no disponemos de constructor, usamos un método del que disponen todas las colecciones denominado iterator() y que cuando es invocado devuelve un objeto de tipo Iterator con una copia de la colección. Por tanto hemos de distinguir:

a) Iterator con mayúsculas, que define un tipo.

b) El método iterator (con minúsculas) que está disponible para todas las colecciones y que cuando es invocado devuelve un objeto de tipo Iterator con una copia de la colección.

¿Cómo sabemos cuándo nos referimos al tipo Iterator y cuándo al método iterator()? Prestando atención a las mayúsculas/minúsculas y al contexto de uso. Si el nombre de una colección va seguido de .iterator() sabremos que estamos utilizando un método para obtener el iterador de la colección. Por las

minúsculas, por los paréntesis, y por ir a continuación del nombre de la colección. Iterator es un tipo genérico o parametrizado, porque requiere que definamos un tipo complementario cuando declaramos objetos de tipo Iterator. Los objetos de tipo Iterator tienen como métodos:

hasNext(): devuelve true si el objeto no es el último de la colección (the object has next).

next(): hace que el objeto pase a referenciar al siguiente elemento de la lista.

remove(): elimina de la colección que se está recorriendo el último objeto devuelto por next().

Uno de los métodos disponibles para la clase String es *contains*. La sintaxis para su uso es "Cadena".contains ("textoabuscar"), que devuelve true si *Cadena* contiene *textoabuscar*. Consideremos el siguiente ejemplo:

```
//Ejemplo aprenderaprogramar.com
if (coleccion.next().contains(cadena) ) { System.out.println ("Cadena encontrada en " + coleccion.next() );    }
```

Este código presenta un problema. ¿Cuál es? Que con la primera referencia a coleccion.next() se evalúa si un ítem de la colección contiene la variable *cadena*, y cuando le decimos que imprima que se ha encontrado la cadena en coleccion.next() no nos devuelve el ítem deseado. ¿Por qué? Porque cada vez que aparezca el método next() se devuelve el siguiente objeto dentro de la colección.

Por eso, en el siguiente ejemplo, donde queremos evaluar y mostrar el String que se analiza, utilizamos otro String temporal para almacenar el que nos devuelve cada llamada al método next(). Escribe y compila este código:

```
import java.util.Iterator;
import java.util.ArrayList;

public class TestUsoIterator {    //Ejemplo uso iterator aprenderaprogramar.com

   public static void main (String [ ] Args) {
      ArrayList <String> listaDeNombres = new ArrayList <String> ();
      listaDeNombres.add("Juan Pérez Sánchez");
      listaDeNombres.add("José Alberto Reverón Montes");
      String cadenaBuscar = "Alberto";
      System.out.println ("La cadena que buscamos es " + cadenaBuscar);

      Iterator<String> it = listaDeNombres.iterator();   //Creamos el objeto it de tipo Iterator con String
      String tmpAnalizando;
      while ( it.hasNext() ) {          //Utilizamos el método hasNext de los objetos tipo Iterator
         tmpAnalizando = it.next();  //Utilizamos el método next de los objetos tipo Iterator
         System.out.println ("Analizando elemento... " + tmpAnalizando);
         if (tmpAnalizando.contains(cadenaBuscar) ) {
            System.out.println ("Cadena encontrada!!!");

         } else { }   //else vacío. No hay acciones a ejecutar.
      } //Cierre del while
   } //Cierre del main
} //Cierre de la clase
```

Además de las ventajas propias de trabajar con una copia en vez de con la colección original, otro aspecto de interés de la clase Iterator y el método iterator radica en que no todas las colecciones de objetos en Java tienen un índice entero asociado a cada objeto, y por tanto no se pueden recorrer basándonos en un índice. En cambio, siempre se podrán recorrer usando un iterador.

EJERCICIO

Crea una clase denominada ListaCantantesFamosos que disponga de un atributo ArrayList listaCantantesFamosos que contenga objetos de tipo CantanteFamoso. La clase debe tener un método que permita añadir objetos de tipo CantanteFamoso a la lista. Un objeto de tipo CantanteFamoso tendrá como atributos nombre (String) y discoConMasVentas (String), y los métodos para obtener y establecer los atributos. Crea una clase test con el método main que inicialice un objeto ListaCantantesFamosos y añade manualmente dos objetos de tipo CantanteFamoso a la la lista. Usando iterator muestra los nombres de cada cantante y su disco con más ventas por pantalla. Se debe pedir al usuario un nombre y disco con más ventas de otro cantante famoso, y una vez introducidos los datos mostrar la lista actualizada usando iterator. Una vez mostrada la lista actualizada, se debe dar opción a elegir entre volver a introducir los datos de otro cantante o salir del programa (se podrán introducir tantos datos de cantantes como se desee. Para ello usa un bucle while que dé opción a elegir al usuario). Puedes comprobar si tu código es correcto consultando en los foros aprenderaprogramar.com.

Ejemplo de lo que podría ser la ejecución del programa:

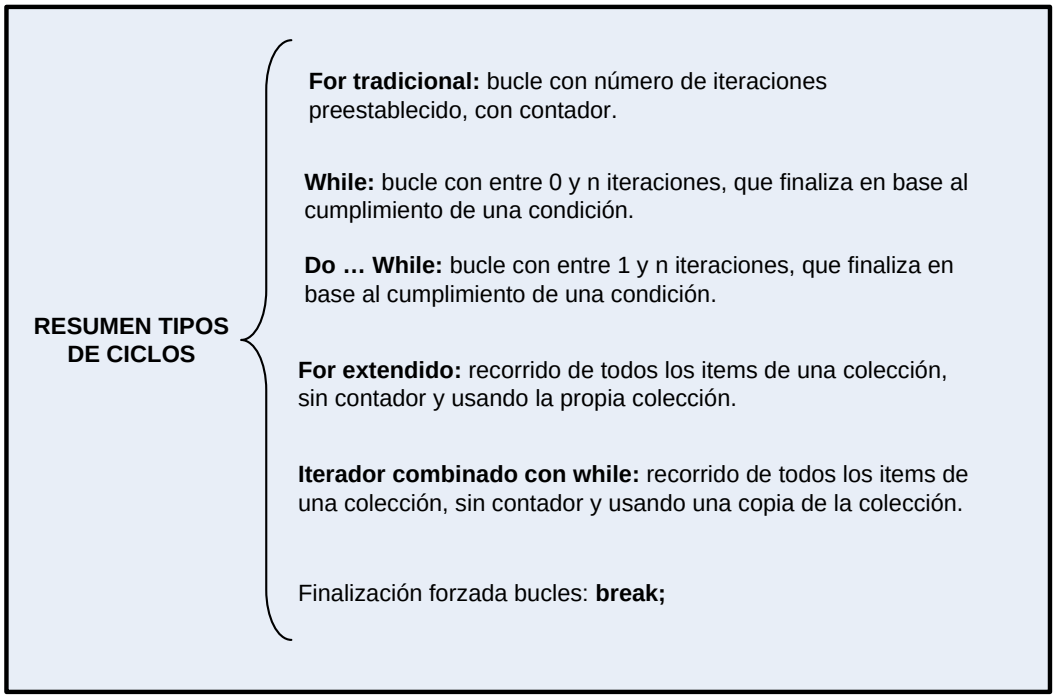
La lista inicial contiene los siguientes datos:
Cantante: Madonna. Disco con más ventas: All I want is you.
Cantante: Jorge Negrete Disco con más ventas: Jalisco.

Por favor introduzca los datos de otro cantante.
Nombre: Michael Jackson
Disco con más ventas: Thriller
La lista actualizada contiene los siguientes datos:
Cantante: Madonna. Disco con más ventas: All I want is you.
Cantante: Jorge Negrete Disco con más ventas: Jalisco.
Cantante: Michael Jackson Disco con más ventas: Thriller.
¿Desea introducir los datos de otro cantante (s/n)?
s
Por favor introduzca los datos de otro cantante.
Nombre: Luis Miguel
Disco con más ventas: Mi jardín oculto
La lista actualizada contiene los siguientes datos:
Cantante: Madonna. Disco con más ventas: All I want is you.
Cantante: Jorge Negrete Disco con más ventas: Jalisco.
Cantante: Michael Jackson Disco con más ventas: Thriller.
Cantante: Luis Miguel Disco con más ventas: Mi jardín oculto.

¿Desea introducir los datos de otro cantante (s/n)?
n

TIPOS DE BUCLES O CICLOS EN JAVA (RESUMEN)

Hasta ahora hemos visto distintas maneras de recorrer una colección o recorrer un bucle. El siguiente esquema es un resumen de ello.



Próxima entrega: CU00668B

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188