APRENDERAPROGRAMAR.COM

INTERFACE ITERATOR DEL API JAVA. MÉTODO REMOVE PARA BORRAR OBJETOS EN COLECCIONES. EJERCICIOS Y EJEMPLO RESUELTO. (CU00919C)

Sección: Cursos
Categoría: Lenguaje de programación Java nivel avanzado I

Fecha revisión: 2039

Resumen: Entrega nº19 curso "Lenguaje de programación Java Nivel Avanzado I".

Autor: Manuel Sierra y José Luis Cuenca

INTERFAZ ITERATOR

Aunque esta interfaz ha sido usada ya antes en este curso, en concreto cuando vimos la interfaz Iterable del paquete java.lang del api de Java, en aquella ocasión no la estudiamos con detalle. Ahora nos centramos de lleno en ella para ver y aprender todas las posibilidades que esta interface nos ofrece como programadores Java.

ITERATOR

La interface Iterator también pertenece como las anteriores al framework Collections de Java. Esta interface nos permite iterar sobre una colección de elementos. Para ello hemos de implementar sus métodos, que son los siguientes: boolean hasNext(), E next(), void remove().

Los dos primeros métodos ya los vimos en funcionamiento en la anterior entrega sobre la interfaz Iterable del paquete java.lang. Pero a continuación veremos con un ejemplo sencillo la utilidad del método remove que es un método bastante importante a la hora de crear un iterador ya que facilita la tarea de eliminar un elemento de una colección al mismo tiempo que la recorremos, cosa que no resulta sencillo de hacer cuando la colección se recorre con un bucle tradicional.

EJERCICIO RESUELTO

Vamos a desarrollar el siguiente ejercicio: a partir de la clase Persona, vamos a crear una clase que sea una colección de Personas. En este caso para no complicar el ejercicio, vamos a utilizar un ArrayList para crear nuestra clase "colección de Personas", que va a llamarse ListaPersonas.

Esta clase ListaPersonas dispondrá de iteradores que crearemos mediante una clase que implemente la interfaz Iterator y por tanto sus 3 métodos.

Después en el programa principal usaremos la clase ListaPersonas en la que vamos a recorrer la lista con el iterador y vamos a borrar ciertas personas al mismo tiempo que hacemos el recorrido.

En concreto vamos a eliminar las personas de la lista cuya estatura sea menor de un determinado valor.

Escribe ahora el siguiente código con el que vamos a trabajar:

```
/* Ejemplo Interface Iterator aprenderaprogramar.com */

public class Persona{
   private int idPersona;
   private String nombre;
   private int altura;

   public Persona(int idPersona, String nombre, int altura) {
      this.idPersona = idPersona;       this.nombre = nombre;       this.altura = altura;}

   public int getAltura(){return altura;} //Omitimos otros métodos para simplificar el ejercicio

   @Override
   public String toString() {
      return "Persona-> ID: "+idPersona+" Nombre: "+nombre+" Altura: "+altura+"\n";
   }
}
```

Para la clase Persona tendremos las propiedades más sencillas posibles con tan solo un constructor y sobreescrito el método toString para la visualización.

Definiremos la clase ListaPersonas de la siguiente manera:

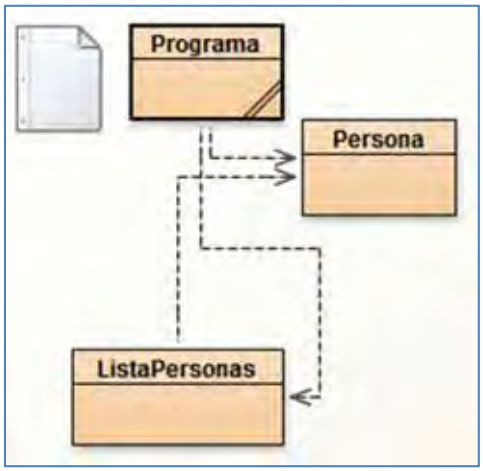
```
/* Ejemplo Interface Iterator aprenderaprogramar.com */
import java.util.ArrayList;
import java.util.Iterator;
public class ListaPersonas {
   private ArrayList<Persona> listapersona = null;  // Campo de la clase
   public ListaPersonas() {listapersona = new ArrayList<Persona>();}  // Constructor de la clase
   public ArrayList<Persona> getListaPersonas(){return listapersona;} //Omitimos otros métodos
   public Iterator<Persona> iterator() {return new MilteradorListaPersonas();} // Método de la clase
   @Override
   public String toString() {return listapersona.toString();}  // Método de la clase
   protected class MilteradorListaPersonas implements Iterator<Persona>  // Clase interna
   {
      public int posicion = 0;    public boolean sepuedeeliminar = false;  // Campos
      @Override
      public boolean hasNext() {return posicion < listapersona.size();}  // Método
      @Override
      public Persona next() {  // Método
         Persona res = listapersona.get(posicion); // Creamos un objeto Persona igual al que recorremos
         posicion ++;
         sepuedeeliminar = true;
         return res;}
      @Override
      public void remove() {
         if (sepuedeeliminar) {
            listapersona.remove(posicion-1);
            posicion--;
            sepuedeeliminar = false; }
      } // Cierre del método remove
   } // Cierre de la clase interna
}  // Cierre de la clase
```

En el anterior código podemos observar que hemos creado la clase MiIteradorListaPersonas como clase interna dentro de la clase ListaPersonas. La clase MiIteradorListaPersonas es la clase que implementa la interface Iterator sobre Persona. Así hemos tenido que implementar los métodos hasNext, next y remove pues son los que define la interface y por tanto estamos obligados a implementar. Siguiendo la documentación, no se debe permitir eliminar un elemento si previamente no se ha hecho next sobre él. Para controlar si es posible o no eliminar un elemento, hemos tenido que usar una variable booleana de control a la que hemos denominado "sepuedeeliminar".

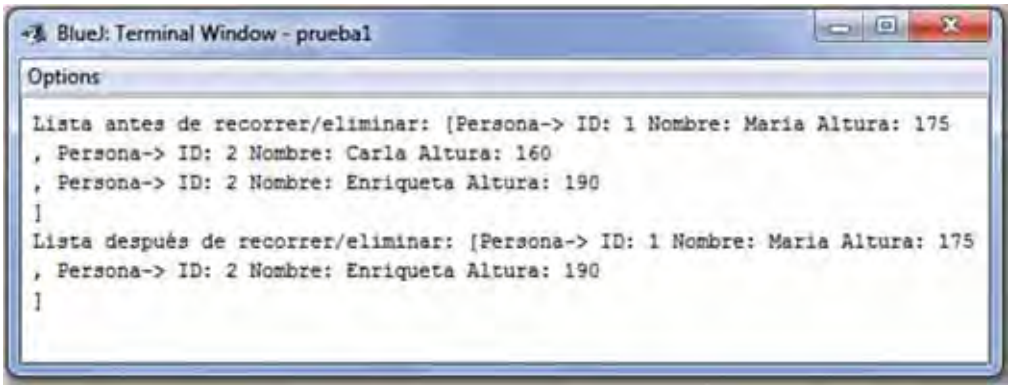
Por último el código del programa principal donde usamos las clases Persona y ListadoPersonas con su iterador. En este ejemplo vamos a eliminar de la colección de Personas a aquellas que tienen una estatura menor de 170 centímetros. Obviamente el criterio de eliminación dependerá de los objetivos del programa, en este caso tan solo tratamos de mostrar un ejemplo de uso.

```
/* Ejemplo Interfaz Iterator aprenderaprogramar.com */
import java.util.Iterator;
public class Programa {
   public static void main(String arg[]) {
        ListaPersonas lp = new ListaPersonas();
        Iterator<Persona> it;
        Persona e; // Este objeto lo usaremos para almacenar temporalmente objetos Persona
        lp.getListaPersonas().add(new Persona(1,"Maria",175));
        lp.getListaPersonas().add(new Persona(2,"Carla",160));
        lp.getListaPersonas().add(new Persona(3,"Enriqueta",190));
        System.out.println("Lista antes de recorrer/eliminar: "+lp.toString());
        it = lp.iterator();
        while (it.hasNext() ) {
           e = it.next();
           if (e.getAltura()<170) {it.remove(); } //  it.remove() elimina la persona de la colección
        }
        System.out.println("Lista después de recorrer/eliminar: " + lp.toString() );
    }
}
```

Siendo el diagrama de clases definido como muestra la siguiente imagen:



Se obtiene la siguiente salida:



Podemos observar cómo previamente al recorrido teníamos a tres personas: María, Carla y Enriqueta, respectivamente con una altura de 175, 160 y 190 centímetros.

Una vez hacemos el recorrido con el iterador sobre la colección de Personas (a la que hemos llamado ListaPersonas), vemos que se ha eliminado a Carla por tener una altura inferior a 170 centímetros, que es la condición que hemos puesto en nuestro ejemplo.

CONCLUSIONES

Vemos cómo implementando la interface Iterator no solo podemos hacer iteraciones sobre las colecciones, sino que también es muy fácil acceder a los elementos y la operación de borrado es relativamente poco costosa. Si estuviéramos realizando un recorrido con un for tradicional, el borrado sería problemático desde el momento en que alteraríamos la cantidad de elementos en la colección y se vería afectado el recorrido. Usando un iterador no ocurre esto porque el recorrido no usa a la colección en sí directamente, sino que digamos se usa una copia para realizar el recorrido. De esta forma se hacen posibles operaciones como el borrado sin causar efectos colaterales en la colección original.

EJERCICIO

Crea una clase denominada Animal con los atributos especie (String), nombre (String), peso (double) y patas (int).

Crea una clase ListaDeAnimales que permita crear una colección de animales y que permita el recorrido de la colección usando un iterador.

Crea una clase con el método main donde se introduzcan 4 animales en la lista y se muestren por pantalla usando el iterador e indicando el número que ocupan en la lista y el total de animales en la lista.

Tras esto, debe solicitarse al usuario que indique un número de la lista a eliminar. Usando el método remove() debe eliminarse ese animal de la lista y volver a mostrarse la lista indicando el nuevo número que ocupa cada animal en la lista y el total de animales en la lista.

Ejemplo de ejecución:

```
Los animales en la lista son:
Especie: pantera, Nombre: Penelope, peso: 98.55 kg, patas: 4, número: 1
Especie: loro amazónico, Nombre: Juanito, peso: 3.67 kg, patas: 2, número: 2
Especie: perro grandanés, Nombre: Artur, peso: 38.77 kg, patas: 4, número: 3
Especie: mono de tanzania, Nombre: Monk, peso: 55.32 kg, patas: 2, número: 4
¿Qué número desea eliminar? 2
Los animales en la lista son:
Especie: pantera, Nombre: Penelope, peso: 98.55 kg, patas: 4, número: 1
Especie: perro grandanés, Nombre: Artur, peso: 38.77 kg, patas: 4, número: 2
Especie: mono de tanzania, Nombre: Monk, peso: 55.32 kg, patas: 2, número: 3
```

Para comprobar si es correcta tu solución puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00920C
Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=58&Itemid=180