APRENDERAPROGRAMAR.COM

INTERFACE LIST DEL API JAVA. CLASES ARRAYLIST, LINKEDLIST, STACK, VECTOR. EJEMPLO CON ARRAYLIST. (CU00920C)

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

Fecha revisión: 2039

Resumen: Entrega nº20 curso "Lenguaje de programación Java Nivel Avanzado I".

Autor: Manuel Sierra y José Luis Cuenca

INTERFACE LIST

La interface List la hemos venido utilizando aunque quizás sin ser plenamente conscientes de ello. Esta interface es la encargada de agrupar una colección de elementos en forma de lista, es decir, uno detrás de otro. En una lista los elementos pueden ser accedidos por un índice que indica la posición del elemento en la colección.

LIST

Esta interfaz también conocida como "secuencia" normalmente acepta elementos repetidos o duplicados, y al igual que los arrays es lo que se llama "basada en 0". Esto quiere decir que el primer elemento no es el que está en la posición "1", sino en la posición "0".

Esta interfaz proporciona debido a su uso un iterador especial (la interfaz Iterator e Iterable las hemos podido conocer en anteriores entregas) llamada ListIterator. Este iterador permite además de los métodos definidos por cualquier iterador (recordemos que estos métodos son hasNext, next y remove) métodos para inserción de elementos y reemplazo, acceso bidireccional para recorrer la lista y un método proporcionado para obtener un iterador empezando en una posición específica de la lista.

Debido a la gran variedad y tipo de listas que puede haber con distintas características como permitir que contengan o no elementos null, o que tengan restricciones en los tipos de sus elementos, hay una gran cantidad de clases que implementan esta interfaz.

A modo de resumen vamos a mencionar las clases más utilizadas de acuerdo con nuestra experiencia.

- ArrayList.
- LinkedList.
- Stack.
- Vector.

LA CLASE ARRAYLIST

Vamos a hablar brevemente sobre todas las clases anteriores, pero vamos a comenzar por ArrayList que ha sido una de las clases en las que hemos venido trabajando más a menudo por lo que ya conocemos parte de ella.

ArrayList como su nombre indica basa la implementación de la lista en un array. Eso sí, un array dinámico en tamaño (es decir, de tamaño variable), pudiendo agrandarse el número de elementos o disminuirse. Implementa todos los métodos de la interfaz List y permite incluir elementos null.

Un beneficio de usar esta implementación de List es que las operaciones de acceso a elementos, capacidad y saber si es vacía o no se realizan de forma eficiente y rápida. Todo arraylist tiene una propiedad de capacidad, aunque cuando se añade un elemento esta capacidad puede incrementarse. Java amplía automáticamente la capacidad de un arraylist a medida que va resultando necesario.

A través del código podemos incrementar la capacidad del arraylist antes de que este llegue a llenarse usando el método ensureCapacity. Esta clase no es sincronizada lo que entre otras cosas significa que si hay varios procesos concurrentes (procesos que se ejecutan al mismo tiempo) sobre un objeto de este tipo y en dos de ellos se modifica la estructura del objeto se pueden producir errores.

EJERCICIO RESUELTO CON ARRAYLIST

Vamos a desarrollar el siguiente ejercicio: a partir de la clase Persona, vamos a crear una lista de Personas donde tengamos un conjunto de Personas con sus alturas correspondientes. A modo de estadística queremos calcular la estatura media de ese conjunto de Personas y veremos lo rápido y eficiente que esto se puede hacer con la implementación de un ArrayList.

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

```
/* Ejemplo Interfaz List 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 get y set para simplificar

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

}
```

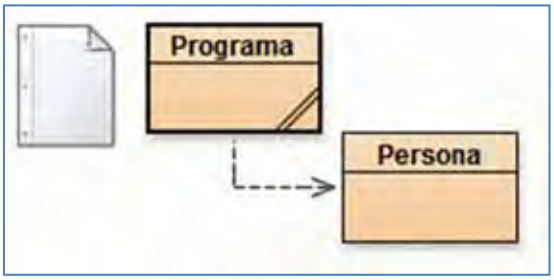
Para la clase Persona hemos creado una clase simplificada con las propiedades más sencillas posibles, con tan solo un constructor, un get y sobreescrito el método toString para la visualización.

A continuación definiremos el programa principal de la siguiente manera:

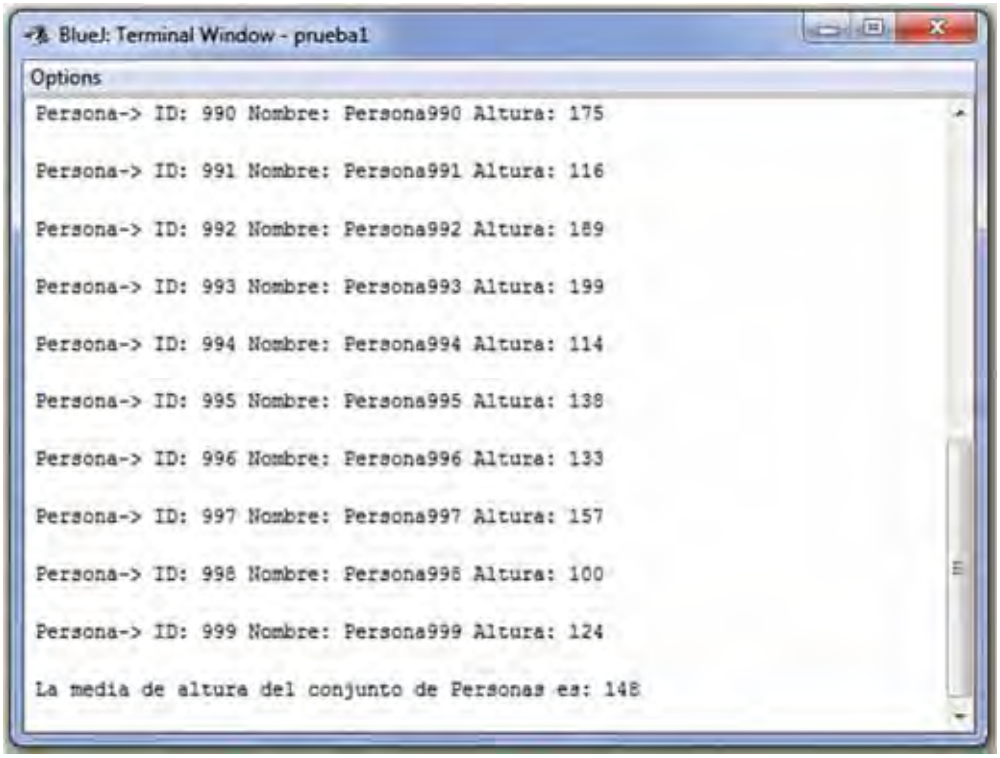
```
/* Ejemplo Interfaz List aprenderaprogramar.com */

import java.util.List;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Random;

public class Programa {

   public static void main(String arg[]) {

        List<Persona> lp = new ArrayList<Persona>(); // El tipo es List y lo implementamos con ArrayList
        Random r = new Random();
        Persona temp = null;
        int sumaaltura = 0;
        for(int i=0;i<1000;i++)
        {lp.add(new Persona(i,"Persona"+i,r.nextInt(100)+100));}
        Iterator<Persona> it = lp.iterator();

        while(it.hasNext())
        {
           temp = it.next();
           System.out.println(temp);
           sumaaltura += temp.getAltura();
        }

        System.out.println("La media de altura del conjunto de Personas es: "+sumaaltura/lp.size());
    }
}
```

En el anterior código hemos creado un ArrayList lp que es nuestro objeto con la lista de Personas. Procedemos con una carga bastante pesada como pueden ser 1.000 objetos de la clase Persona. Una vez cargadas hemos hecho un recorrido sobre esta y hemos calculado finalmente la media de altura de las Personas que se ha generado aleatoriamente entre 100 cm de altura y 200 cm.

Con el siguiente diagrama de clases en BlueJ:



Obtenemos la siguiente salida por consola:



CONCLUSIONES

Vemos cómo la implementación de ArrayList para la interface List es rápida y eficiente (aunque como comentamos no segura en sincronización) y posiblemente por ello sea la implementación más usada de List. En tan solo unos segundos o incluso menos calculamos la media de miles de personas.

EJERCICIO

Crea una clase denominada Cuadrupedo con los atributos idCuadrupedo (int) y tipo (String), donde tipo podrá tomar los valores León, Gato, Perro o Elefante.

Crea una clase con el método main donde se introduzcan 10000 cuadrúpedos en una lista de tipo estático List y tipo dinámico ArrayList. El atributo tipo debe establecerse para cada objeto de forma aleatoria. A continuación, el programa debe mostrar por consola los datos de los objetos con idCuadrupedo múltiplo de 1000 y mostrar un resumen de cuántos cuadrúpedos hay de cada tipo.

Ejemplo de ejecución:

Cuadrúpedo-> ID: 1000 Tipo: Leon
Cuadrúpedo-> ID: 2000 Tipo: Elefante
Cuadrúpedo-> ID: 3000 Tipo: Gato

Cuadrúpedo-> ID: 4000 Tipo: Gato
Cuadrúpedo-> ID: 5000 Tipo: Perro
Cuadrúpedo-> ID: 6000 Tipo: Perro
Cuadrúpedo-> ID: 7000 Tipo: Gato
Cuadrúpedo-> ID: 8000 Tipo: Gato
Cuadrúpedo-> ID: 9000 Tipo: Perro
Cuadrúpedo-> ID: 10000 Tipo: Leon

Resumen: hay 2470 Leones,2511 Gatos,2575 Perros y 2444 Elefantes

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

Próxima entrega: CU00921C

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