APRENDERAPROGRAMAR.COM

LA CLASE ARRAYLIST DEL API DE JAVA. MÉTODOS ADD, SIZE, ETC. CONCEPTO DE CLASE GENÉRICA O PARAMETRIZADA (CU00665B)

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

Fecha revisión: 2029

Resumen: Entrega nº65 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

LA CLASE ARRAYLIST DEL API DE JAVA. LISTAS REDIMENSIONABLES.

La clase ArrayList podríamos encuadrarla de la siguiente manera: Colecciones --> Listas --> ArrayList. Esta clase pertenece a la biblioteca java.util. Por tanto, para emplearla en nuestras clases o programas escribiremos como código en cabecera *import java.util.ArrayList* (o de forma más genérica *import java.util.**). **Un objeto ArrayList será una lista redimensionable** en la que tendremos disponibles los métodos más habituales para operar con listas.

Como métodos para operar con listas podemos señalar: añadir un objeto en una posición determinada, añadir un objeto al final de la lista, recuperar un objeto situado en determinada posición, etc. Los objetos de un ArrayList tienen un orden, que es el orden en el que se insertaron en la lista.

Un aspecto a tener muy presente: hablamos de colecciones de objetos. Por tanto, un ArrayList no puede ser una lista de enteros como tipo primitivo (int) pero sí de objetos Integer. La sintaxis que emplearemos con ArrayList es la siguiente:

(Declaración del objeto ArrayList) : *private ArrayList<TipoDeObjetosEnLaColección> NombreDelArrayList;*

(Creación de un objeto): NombreDeObjeto = *new ArrayList<TipodeObjetosEnLaColección>();*

(Uso del método reemplazar objeto existente): *NombreDelArrayList.set (int índice, E elemento);*

(Uso del método añadir al final): *NombreDelArrayList.add (objeto_a_añadir);*

(Uso del método obtener el número de objetos en la lista): *NombreDelArrayList.size();*

(Uso del método extraer un objeto de cierta posición): *NombreDelArrayList.get (posición);*

Si consultas la documentación de la clase, verás que la clase ArrayList tiene varios constructores. En este caso estamos utilizando el constructor sin parámetros que crea una lista ArrayList vacía con una capacidad inicial para diez objetos (la capacidad es modificable luego y se amplía automáticamente a medida que vamos añadiendo elementos). Escribe y compila este código para probar el funcionamiento de ArrayList.

```
//Ejemplo de uso ArrayList aprenderaprogramar.com
import java.util.ArrayList; //Los import deben ir siempre al principio antes de declarar la clase

//Esta clase representa una lista de nombres manejada con la clase ArrayList de Java
public class ListaNombres {
   private String nombreDeLaLista; //Establecemos un atributo nombre de la lista
   private ArrayList<String> listadenombres; //Declaramos un ArrayList que contiene objetos String

   public ListaNombres (String nombre) {    //Constructor: crea una lista de nombres vacía
      nombreDeLaLista = nombre;
      listadenombres = new ArrayList<String>(); //Creamos el objeto de tipo ArrayList
   } //Cierre del constructor
```

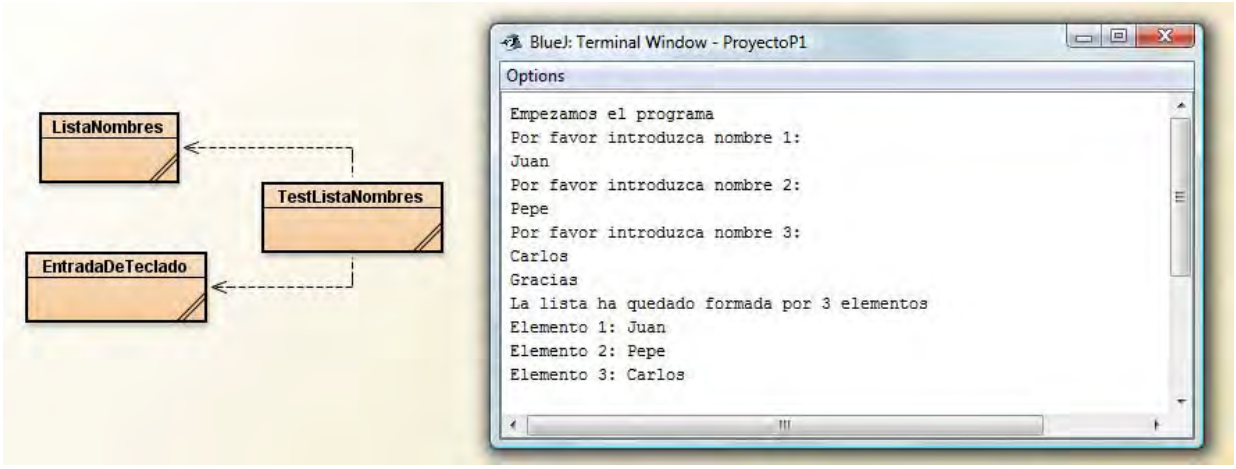
```
    public void addNombre (String valor_nombre) { listadenombres.add (valor_nombre); } //Cierre del método

    public String getNombre (int posicion) { //Método
      if (posicion >= 0 && posicion < listadenombres.size() ) {
        return listadenombres.get(posicion); }
      else { return "No existe nombre para la posición solicitada";}
    } //Cierre del método

    public int getTamaño () { return listadenombres.size(); } //Cierre del método

    public void removeNombre (int posicion) {  //Método
      if (posicion >= 0 && posicion < listadenombres.size() ) {
        listadenombres.remove(posicion); }
      else { } //else vacío. No existe nombre para la posición solicitada, no se ejecuta ninguna instrucción
    } //Cierre del método removeNombre

  } //Cierre de la clase
```

Crea un objeto y añade varios nombres en el ArrayList. Prueba el correcto funcionamiento de los métodos disponibles. Ten en cuenta que al eliminar un objeto de la colección, todos los elementos posteriores se renumeran disminuyendo su índice una posición, automáticamente. Muchas colecciones, entre ellas ArrayList, tienen una numeración implícita de cada uno de los objetos de los que consta. Esta numeración va desde cero hasta (número de elementos -1), es decir, en una colección de 18 personas los índices van desde persona(0) hasta persona(17), o en una colección que tiene 155 depósitos, los índices irán de deposito(0) a deposito(154). El número de elementos en la colección lo podemos obtener en cualquier momento utilizando el método size(). Si intentamos acceder a una colección con un índice no válido obtendremos un error del tipo “IndexOutBoundsException” (desbordamiento).

En apartados anteriores definimos una clase denominada EntradaDeTeclado que servía para recibir datos de entrada introducidos por el usuario mediante el teclado. Vamos ahora a utilizar esa clase para crear un pequeño test de la clase ListaNombres. Escribe el siguiente código:

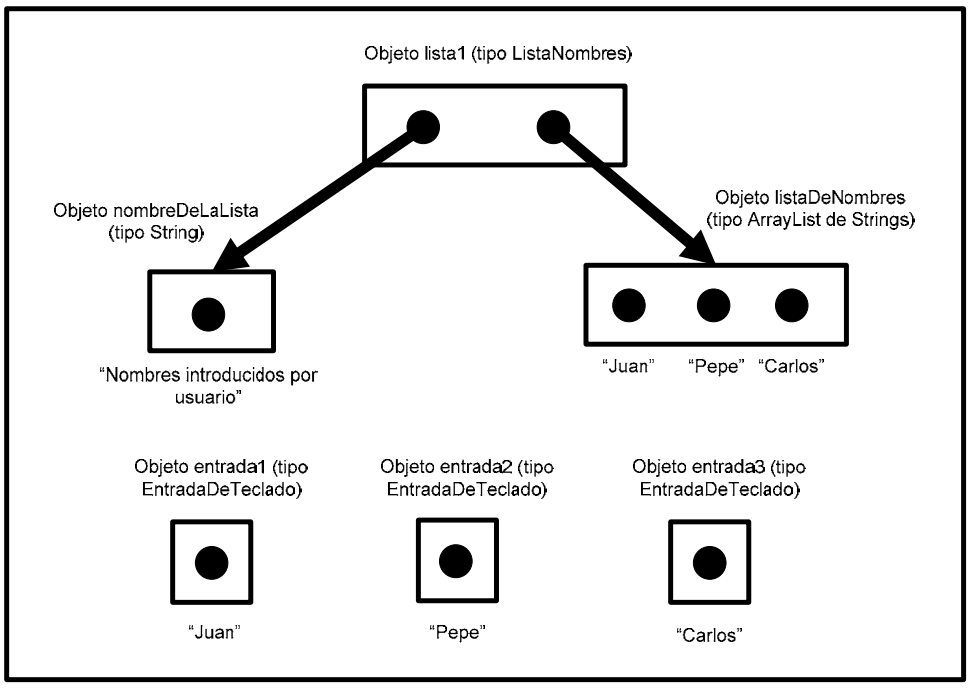
```
// Aquí el test con el método main ejemplo aprenderaprogramar.com
public class TestListaNombres {

    public static void main (String [ ] args) {

      System.out.println ("Empezamos el programa");
      System.out.println ("Por favor introduzca nombre 1:");      EntradaDeTeclado entrada1 = new EntradaDeTeclado();
      System.out.println ("Por favor introduzca nombre 2:");      EntradaDeTeclado entrada2 = new EntradaDeTeclado();
      System.out.println ("Por favor introduzca nombre 3:");      EntradaDeTeclado entrada3 = new EntradaDeTeclado();
      System.out.println ("Gracias");
      ListaNombres lista1 = new ListaNombres("Nombres introducidos por usuario");
      lista1.addNombre (entrada1.getEntrada() ); lista1.addNombre (entrada2.getEntrada() );
      lista1.addNombre (entrada3.getEntrada() );
      System.out.println ("La lista ha quedado formada por " + lista1.getTamaño() + " elementos");
      System.out.println ("Elemento 1: " + lista1.getNombre(0) );
      System.out.println ("Elemento 2: " + lista1.getNombre(1) );
      System.out.println ("Elemento 3: " + lista1.getNombre(2) );
    } //Cierre del main

  } //Cierre de la clase
```

El diagrama de clases y el resultado de ejecutar el programa son de este tipo:



La clase TestListaNombres usa tanto la clase ListaNombres como la clase EntradaDeTeclado. El código de EntradaDeTeclado ya lo teníamos escrito por haberlo usado en un programa anterior y por tanto no hemos de escribirlo de nuevo.

El siguiente diagrama muestra los objetos que intervienen en el programa anterior:



Vamos a interpretar este diagrama. El objeto lista1 contiene dos objetos: un objeto de tipo String y un objeto de tipo ArrayList. A su vez el ArrayList contiene tres objetos de tipo String. Por otro lado tenemos los objetos EntradaDeTeclado, cada uno de los cuales contiene un objeto de tipo String. ¿Por qué no salen flechas desde los objetos EntradaDeTeclado? Porque la flecha indica hacia dónde apunta una variable referenciadora de un objeto. Las variables referenciadoras de los objetos EntradaDeTeclado no apuntan (no contienen) otros objetos. Simplemente nos sirven para extraer algo. Fíjate que usamos lista1.addNombre (entrada1.getEntrada()); en vez de lista1.addNombre (entrada1);. La diferencia radica en que en el primer caso añadimos un String al ArrayList, mientras que en el segundo caso estaríamos añadiendo un objeto EntradaDeTeclado.

CONCEPTO DE CLASE GENÉRICA O CLASE PARAMETRIZADA EN JAVA

Fijémonos en la sintaxis de estas declaraciones:

DECLARACIÓN EN OTRAS CLASES	DECLARACIÓN DE TIPO ARRAYLIST
private String cadenaDeTexto; private int numeroDepositosGrupo; private Persona jefeDePersonal;	**private ArrayList<String> listaDeNombres;**

Mientras que para declarar un String o una Persona no necesitamos nada más que el nombre de la clase, para declarar un ArrayList hemos de usar además un parámetro especificado entre los símbolos < y >. Por ejemplo:

```
private ArrayList<String> listaDeFrutas; //Lista que admite solo Strings
private ArrayList<Persona> miembrosDelClub; //Lista que admite solo Personas
private ArrayList<Deposito> grupoDeDepositos; //Lista que admite solo Depósitos
```

Solo podemos añadir objetos del tipo declarado al parametrizar la clase. Este tipo de clases, que requieren un tipo como parámetro, se denominan **"clases genéricas o parametrizadas".** Dado que ArrayList es una clase que puede definir una lista de distintos tipos de objeto (como Strings, Personas, Depósitos…) decimos que la clase ArrayList define potencialmente muchos tipos puesto que con ella podemos crear listas de cualquier tipo de objetos. Un tipo ArrayList<String> es distinto a un tipo ArrayList<Deposito>, por tanto podemos tener infinitas clases basadas en la clase genérica ArrayList.

En la documentación del API de Java, que una clase está parametrizada se refleja en su documentación. Por ejemplo, si consultamos la documentación de ArrayList veremos que en cabecera aparece como Class ArrayList<E> lo que nos indica que se requiere un parámetro para crear objetos de tipo ArrayList. Si consultamos la documentación de la clase HashMap comprobaremos que en cabecera aparece como Class HashMap <K, V> lo cual nos indica que se requieren dos parámetros para crear objetos de tipo HashMap.

EJERCICIO

Crea una clase denominada ListaCantantesFamosos que al ser inicializada contenga un ArrayList con tres Strings que sean el nombre de cantantes famosos. Crea una clase test con el método main que inicialice un objeto ListaCantantesFamosos, pida dos cantantes famosos más al usuario, los añada a la lista y muestre el contenido de la lista por pantalla. Puedes comprobar si tu código es correcto consultando en los foros aprenderaprogramar.com.

Próxima entrega: CU00666B

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