

The logo for Oracle Academy. It features the word "ORACLE" in a bold, orange, sans-serif font. Below it, the word "Academy" is written in a smaller, dark gray, sans-serif font. The entire logo is centered on a light gray background, which is framed by dark gray horizontal bars at the top and bottom.

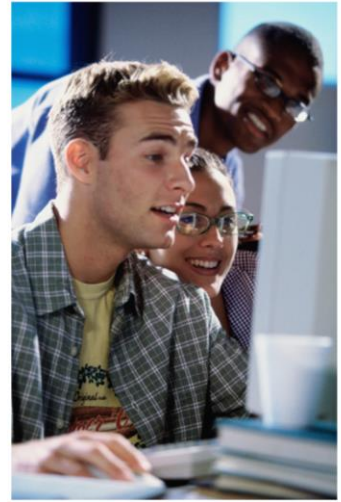
# ORACLE

## Academy

# Java Foundations

## 8-2 ArrayLists

**ORACLE**  
Academy



8-2  
ArrayLists

Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Crear una clase ArrayList
  - Manipular una clase ArrayList con sus métodos
  - Recorrer una clase ArrayList mediante iteradores y bucles for-each
  - Usar clases de envoltorio y conversión automática para agregar tipos de datos primitivos a una clase ArrayList



# Recopilación de objetos (vida real)

- En la vida real, los objetos aparecen a menudo en grupos
- Por ejemplo:
  - Estacionamientos que contienen numerosos coches
  - Bancos que contienen numerosas cuentas
  - Almacenes que contienen numerosos clientes
  - Un estudiante con numerosas calificaciones



**ORACLE**  
Academy

JFo 8-2  
ArrayLists

Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

4

# Recopilación de objetos (programación)

- Al programar, normalmente se recopilan datos (objetos)
- Se suele denominar recopilación



- En Java, la forma más sencilla de recopilación de información consiste en utilizar ArrayList
- La clase ArrayList de Java puede almacenar un grupo de muchos objetos

## Gestión de los estudiantes matriculados en una clase

- Digamos que un grupo de estudiantes está inscrito en Programación Java 101
- Desea escribir un programa Java para realizar un seguimiento de los estudiantes matriculados
- La forma más sencilla sería crear una matriz, como se ha explicado en la lección anterior



## Uso de arreglos para gestionar los estudiantes matriculados

- Puede escribir una matriz de estudiantes como esta:

```
String[] students={"Mary", "Sue", "Harry", "Rick", "Cindy", "Bob"};
```

- Imagine un escenario en el que, después de una semana, dos estudiantes (Mike y Larry) se inscriben en el curso y Sue abandona
- ¿Cree que será fácil modificar la matriz de los estudiantes para que se adapte a estos cambios?

## Limitaciones de las arreglas

- Tienen un tamaño fijo durante su creación y no se pueden ampliar o reducir después de la inicialización
- Tiene que crear métodos manuales para manipular su contenido
- Por ejemplo: Insertar o suprimir un elemento de una matriz



# Clase ArrayList

- Las arreglas no son la única forma de almacenar listas de datos relacionados
- Java ofrece una clase de utilidad especial denominada `ArrayList`
- La clase `ArrayList`:
  - Forma parte de la biblioteca de Java, como las clases `String` y `Math`
  - Se puede utilizar para almacenar una lista de objetos
  - Tiene un juego de métodos útiles para gestionar sus elementos:
    - `add()`, `get()`, `remove()`, `indexOf()` y muchos otros.

`ArrayList` soporta arreglas dinámicas que pueden crecer según sea necesario. En Java, las arreglas estándar tienen una longitud fija. Después de crear las arreglas, no se pueden ampliar o reducir, lo que significa que debe conocer, por adelantado, el número de elementos de una arreglo. Pero, a veces, es posible que no sepa exactamente el tamaño que la arreglo debe tener hasta el tiempo de ejecución.

## ¿Qué puede contener una clase ArrayList?

- Una ArrayList solo puede contener objetos, no primitivos
  - Puede contener cualquier tipo de objeto, incluido un tipo que ha creado mediante la escritura de una clase
- Por ejemplo, una ArrayList puede contener objetos del tipo:
  - String
  - Persona
  - Coche



**ORACLE**  
Academy

JFo 8-2  
ArrayLists

Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

10

El ejemplo de las diapositivas muestra la clase `ArrayList`, `names`, a la que se están agregando dos objetos `String` mediante el método `add` de la `ArrayList`. Puede eliminar un objeto de la `ArrayList` mediante el método `remove`.

# Importación y declaración de una ArrayList

- Debe importar `java.util.ArrayList` para utilizar una `ArrayList`

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

Puede especificar una capacidad inicial, pero no es obligatorio

```
public class ArrayListExample {  
    public static void main (String[] args) {  
        ArrayList<String> states = new ArrayList<>();  
    } //end method main  
} //end class ArrayListExample
```

Puede especificar cualquier tipo de objeto, llamado parámetros de tipo, que especifica que solo contiene los objetos `String`

## Trabajar con una ArrayList

- No puede acceder a elementos de una ArrayList a través de la notación de índice
- En su lugar, se utilizan una serie de métodos que están disponibles en la clase ArrayList

## Algunos métodos ArrayList

<b>add(value)</b>	Agrega el valor al final de la lista
<b>add(index, value)</b>	Inserta el valor determinado justo antes del índice especificado, trasladando los valores siguientes a la derecha
<b>clear()</b>	Elimina todos los elementos de la lista
<b>indexOf(value)</b>	Devuelve el primer índice donde se encuentra el valor especificado en la lista (-1 si no se encuentra)
<b>get(index)</b>	Devuelve el valor en el índice especificado
<b>remove(index)</b>	Elimina el valor en el índice especificado, trasladando los valores siguientes a la izquierda
<b>set(index, value)</b>	Reemplaza el valor en el índice especificado con un valor dado
<b>size()</b>	Devuelve el número de elementos en la lista
<b>toString()</b>	Devuelve una representación de cadena de la lista, como, por ejemplo, "[3, 42, -7, 15]"

# Trabajar con una ArrayList

- A continuación se muestra un ejemplo que utiliza estos métodos:

```
ArrayList<String> names;
names = new ArrayList();

names.add("Jamie");
names.add("Gustav");
names.add("Alisa");
names.add("Jose");
names.add(2, "Prashant");

String str = names.get(0);
System.out.println(str);

names.remove(0);
names.remove(names.size() - 1);
names.remove("Gustav");

System.out.println(names);
```

Declarar una ArrayList de cadenas

Instanciar ArrayList

Agregar elementos

Recuperar un valor

Eliminar elementos

Visualizar un elemento

**ORACLE**  
Academy

JFo 8-2  
ArrayLists

Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

14

Cuando declare una `ArrayList`, utilice el operador diamante (`<>`) para indicar el tipo de objeto. En la diapositiva de ejemplo, existen una serie de métodos para agregar datos a la `ArrayList`. En este ejemplo, se utiliza el método `add` para agregar varios objetos `string` a la lista: el método `add(int index, E element)` inserta un elemento en una ubicación concreta. Con el método `remove`, puede suprimir un elemento transfiriendo el índice o el elemento que se va a suprimir.

`remove(0)` elimina el primer elemento en el índice cero (en este caso, "Jamie").

`remove(names.size() - 1)` elimina el último elemento, que sería "Jose".

`remove("Gustav")` elimina un elemento que coincide con un valor específico.

Puede transferir la `ArrayList` a `System.out.println`. La salida resultante es [Prashant, Alisa].

# Beneficios de la clase ArrayList

- Cambio de tamaño dinámico:
  - Una ArrayList aumenta conforme se agregan elementos
  - Una ArrayList se reduce conforme se eliminan elementos
- Varios métodos incorporados:
  - Una ArrayList tiene varios métodos para realizar operaciones
  - Por ejemplo, agregar, recuperar o eliminar un elemento

## Ejercicio 1, parte 1

- Cree un nuevo proyecto y agréguele el archivo `ArrayListEx1.java`
- Examine `ArrayListEx1.java`
- Modifique el programa para implementar:
  - Cree una `ArrayList` de `Strings` denominada `estudiantes`
  - Agregue cuatro estudiantes a la `ArrayList`: Amy, Bob, Cindy y David
  - Imprima los elementos de la `ArrayList` y muestre su tamaño



## Ejercicio 1, parte 2

- Modifique el programa para implantar:
  - Agregue dos estudiantes más, Nick y Mike, en el índice 0 y 1,
  - Elimine el estudiante en el índice 3
  - Imprima los elementos de la ArrayList y muestre su tamaño

# Recorrido de una ArrayList

- Puede recorrer una ArrayList de las siguientes formas:
  - Con el bucle for-each
  - Con un Iterator
  - Con un ListIterator

## Recorrido de una ArrayList: bucle for-each

- En la lección anterior, ha utilizado un bucle for-each para recorrer una matriz
- Puede utilizar un bucle for-each para recorrer una ArrayList
- La variable i representa un nombre concreto a medida que realizar bucle de los nombres ArrayList

Tipo de objeto que  
está en la ArrayList  
(en este caso,  
String)

Variable

ArrayList

```
for (String i : names) {  
    System.out.println("Name is " + i);  
} //end for
```

**ORACLE**  
Academy

JFo 8-2  
ArrayLists

Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

19

# Recorrido de una ArrayList: bucle for-each

```
public class ArrayListTraversal {  
    public static void main(String[] args) {  
        ArrayList<String> names = new ArrayList<>();  
        names.add("Tom");  
        names.add("Mike");  
        names.add("Matt");  
        names.add("Nick");  
        System.out.println("");  
        for (String i : names) {  
            System.out.println("Name is " + i);  
        }  
    }  
}
```

Output:

```
Name is Tom  
Name is Mike  
Name is Matt  
Name is Nick
```

## Presentación de Iterator

- Es miembro del marco de recopilaciones
- Permite recorrer todos los elementos de la ArrayList, al obtener o eliminar elementos
- Tiene los siguientes métodos:
  - `hasNext()`, `next()`, `remove()`
- Solo se utiliza para desplazarse hacia adelante
- Debe importar `java.util.Iterator` para utilizar Iterator

## Recorrido de una ArrayList: Iterator

- A continuación se muestra un ejemplo de un recorrido de la recopilación de nombres mediante un iterator

Devuelve un  
objeto iterator

ArrayList

```
Iterator<String> iterator = names.iterator();
while (iterator.hasNext())
{
    System.out.println("Name is " + iterator.next());
} //end while
```

Asociación de una recopilación a  
un iterator

Para utilizar un `iterator` para recorrer el contenido de una `ArrayList`, siga estos pasos:

1. Llame al método `iterator()` de la recopilación.
2. Configure un bucle que realice una llamada a `hasNext()` y haga que itere el bucle siempre que `hasNext()` devuelva `true`.
3. En el bucle, obtenga cada elemento al llamar a `next()`.

# Introducción a ListIterator

- ListIterator
  - Es miembro del marco de recopilaciones
  - Permite recorrer la ArrayList en ambas direcciones
  - No contiene el método remove
- Debe importar `java.util.ListIterator` para utilizar un ListIterator

## Recorrido de una ArrayList: ListIterator

- A continuación se muestra un ejemplo de uso ListIterator para recorrer los nombres de la ArrayList hacia adelante o hacia atrás:

```
ListIterator<String> litr = names.listIterator();

System.out.println("Traversing list forwards: ");
while (litr.hasNext()) {
    System.out.println("Name is " + litr.next());
} //end while

System.out.println("Traversing list backwards: ");
while (litr.hasPrevious()) {
    System.out.println("Name is " + litr.previous());
} //end while
```



# ArrayList y primitivos

- Una ArrayList solo puede almacenar objetos, no primitivos

✗ `ArrayList<int> list = new ArrayList<int>();`

`int` no puede ser un parámetro de tipo

- Pero aún puede utilizar ArrayList con tipos primitivos mediante clases especiales denominadas clases de envoltorio

`ArrayList<Integer> list = new ArrayList<Integer>();`

Clase de envoltorio para `int`

# Clases de envoltorio

- Java ofrece clases, conocidas como clases de envoltorio, que corresponden a los tipos primitivos
- Estas clases encapsulan o envuelven los tipos primitivos en un objeto
- Los ocho tipos de clases de envoltorio se corresponden con cada tipo de dato primitivo

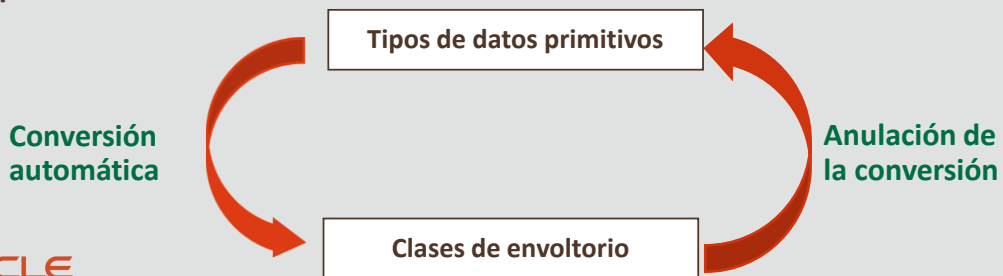
## Lista de clases de envoltorio

- Aquí se muestra la lista de tipos de datos primitivos y sus correspondientes clases de envoltorio:

Tipo primitivo	Tipo de envoltorio
<code>byte</code>	Byte
<code>short</code>	Short
<code>int</code>	Integer
<code>long</code>	Long
<code>float</code>	Float
<code>double</code>	Double
<code>char</code>	Character
<code>boolean</code>	Boolean

# Introducción a la conversión automática y anulación de la conversión

- Java tiene una función llamada conversión automática y anulación de la conversión
- Esta función realiza la conversión automática de tipos de datos primitivos en sus clases de envoltorio y viceversa
- Permite escribir un código más reducido y limpio, con lo que es más fácil de leer



## ¿Qué es la conversión automática?

- La conversión automática que el compilador Java hace entre los tipos primitivos y sus correspondientes clases de envoltorio de objetos

```
Double score = 18.58;
```




**Conversión automática del valor double primitivo**

## ¿Qué es la anulación de la conversión?

- Convertir un objeto de un tipo de envoltorio a su correspondiente valor primitivo

```
1 Double score = 18.58;  
2 double goal = score;
```



**Anulación de la conversión del objeto score de tipo double, al valor score primitivo que es de tipo double**

# Clases ArrayList y de envoltorio

- Clases de envoltorio que permiten a una ArrayList almacenar valores primitivos

```
public static void main(String args[]) {  
    ArrayList<Integer> nums = new ArrayList<>();  
    for (int i = 1; i < 50; i++) {  
        nums.add(i);  
    } //end for  
  
    for(Integer i:nums ){  
        int nos = i;  
        System.out.println(nos);  
    } //end for  
} //end method main
```

Conversion automática

Anulación de la conversión

El ejemplo de código muestra cómo se puede completar ArrayList con datos primitivos mediante la conversión automática y la anulación de la conversión.

**Conversión automática:** int se convierte en un entero y se agrega a ArrayList sin necesidad de tener que llamar a la clase de envoltorio, Integer.

**Anulación de la conversión:** el objeto Integer se convierte en el primitivo int sin llamar a cualquier otro método para realizar la conversión.

## Ejercicio 2

- Agregue el archivo `ArrayListEx2.java` al proyecto creado para el ejercicio 1
- Examine `ArrayListEx2.java`
- Realice lo siguiente:
  - Cree una `ArrayList` con una lista de números
  - Muestre el contenido de la `ArrayList` mediante `Iterador`
  - Elimine todos los números pares
  - Muestre el contenido de la `ArrayList`



# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Crear una clase ArrayList
  - Manipular una clase ArrayList con sus métodos
  - Recorrer una ArrayList mediante iteradores y bucles for-each
  - Usar clases de envoltorio y conversión automática para agregar tipos de datos primitivos a una clase ArrayList



The Oracle Academy logo is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font, with the letters "O", "R", "A", and "C" being slightly larger than "L", "E", and "A". Below "ORACLE" is the word "Academy" in a smaller, dark gray, sans-serif font. The entire logo is framed by two dark gray horizontal bars, one at the top and one at the bottom.

# ORACLE

## Academy