


Open Opened 2 weeks ago by  **Rubén Montero**

Alternativa fácil a arrays

Resumen

- Hablaremos de `ArrayList`
- Veremos algunas de sus ventajas
- Implementaremos un `NewTravelStops`, esta vez usando un `ArrayList`

Descripción

Un [array](#) es tipo de dato que existe en prácticamente todos los lenguajes de programación. El lenguaje [C](#) también permite el uso de *arrays* y **no** es un lenguaje orientado a objetos (POO).

Sin embargo, existe una manera más *orientada a objetos* de trabajar con *arrays*: `ArrayList`.

¿Qué es `ArrayList`?

Se trata de una clase **que ya viene con las librerías esenciales de Java**. Podemos **usarlo** libremente.

Por ejemplo, crearíamos una nueva instancia así:

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

Un segundo... ¿qué es eso de `<String>`?

Bien visto.

Es una sintaxis **especial** que se usa en ciertas ocasiones (no sólo con `ArrayList`) para indicar el tipo específico de algún dato genérico.

¿Y eso qué significa?

En el caso de `ArrayList`, se usa para indicar de **qué tipo** son los elementos *dentro* del `ArrayList`.

¿Y esto de dónde surge?

Originalmente `ArrayList` se usaba como una clase normal. Así:

```
ArrayList miArrayList = new ArrayList();
```

Pero algunas operaciones, como **añadir** un elemento, admitían *cualquier objeto*:

```
Person person = new Person("Isaac Asimov");
int number = 78;

// Y ahora...
ArrayList miArrayList = new ArrayList();
miArrayList.add(person);
miArrayList.add(number); // Mal! Contiene elementos de distinto tipo!
```

...y esto se traducía en **errores inesperados** en tiempo de ejecución.

Para prevenir estos problemas en **tiempo de compilación** (donde son más fáciles de detectar), y, además, favorecer la reutilización de código, se introdujo [genericidad](#) en [Java 1.5](#).

```
Person person = new Person("Isaac Asimov");
int number = 78;

// Y ahora...
ArrayList<String> miArrayList = new ArrayList<>();
miArrayList.add(person);
miArrayList.add(number); // Ahora falla en tiempo de compilación (IntelliJ IDEA lo subraya rojo)
```

Excelente

Sí.

Un momento

¿Con `ArrayList` puedo *añadir* elementos a un *array*?

Efectivamente.

Hasta ahora hemos **inicializado**, **consultado** y **actualizado** un *array*.

¿Te has preguntado cómo sería **añadir** un elemento?

Fácil.

¡No se puede!

Un *array* tiene un tamaño *fijo*. No se puede *modificar*. La solución más parecida sería crear un **nuevo array** con **1** posición más, copiar **todo** el contenido y luego insertar al final el nuevo elemento.

Algo así:

```
String[] diasDeSemana = new String[] { "Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo" };

// Vamos a añadir un nuevo día a la semana
String nuevosDiasDeSemana = new String[8];
for (int i = 0; i < diasDeSemana.length; i++) {
    nuevosDiasDeSemana[i] = diasDeSemana[i];
}
nuevosDiasDeSemana[7] = "Redomingo";
```

Qué tedioso

En efecto.

Usando `ArrayList`, lo conseguimos con una línea:

```
ArrayList<String> diasDeSemana = new ArrayList<>(List.of("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"));

// Vamos a añadir un nuevo día a la semana
diasDeSemana.add("Redomingo");
```

Entonces, ¿cómo uso un `ArrayList` ?

En ejemplos anteriores hemos visto cómo inicializarlo, vacío o con valores¹:

```
ArrayList<String> diasVacio = new ArrayList<>();
ArrayList<String> diasDeSemana = new ArrayList<>(List.of("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"));
```

...y cómo añadir un valor **al final**:

```
ArrayList<String> finDeSemana = new ArrayList<>();
finDeSemana.add("Sábado");
finDeSemana.add("Domingo");
```

También, al igual que con un *array* normal, podemos:

- **Consultar** (recuperar) un valor con `.get()`, pasando el *índice/posición*:

```
// Equivale a String primerDia = dias[0];
String primerDia = dias.get(0);
```

- **Actualizar** un valor con `.set()`, pasando el *índice/posición* y el nuevo valor:

```
// Equivale a dias[0] = "Lunes";
dias.set(0, "Lunes");
```

La tarea

Se pide que añadas una nueva clase `NewTravelStops`.

En dicha clase, **de momento sólo se pide** que:

- Tenga un atributo privado tipo `ArrayList<String>`
- En el método constructor (que **no** recibirá parámetros), se inicialice con los valores:

```
"Kyoto"  
"Singapur"  
"Hanoi"
```

- Tenga un método `public void printStop(int index)` que imprima con `System.out.println` el valor de la *parada* en la posición correspondiente
- Tenga un método `public void changeStop(int index, String newValue)` que *actualice* la *parada* en la posición correspondiente, con el nuevo valor indicado
- Tenga un método `public void addStop(String newValue)` que **añada** al final del `ArrayList`, la nueva parada

Por último

Una vez verifiques que el *test* funciona correctamente, la tarea ha sido completada.

Haz `commit` y `push` para subir los cambios al repositorio.

1. <https://www.geeksforgeeks.org/initialize-an-arraylist-in-java/> 



Rubén Montero @ruben.montero changed milestone to [%Sprint 1](#) 2 weeks ago



Ania Blanco @ania.blanco mentioned in commit [27c69b11](#) 2 weeks ago