DIFICULTADES CON LOS ARRAYS

- Conocer a priori el tamaño
- El tamaño no se puede modificar una vez creado.
- Problemas para insertar objetos en posiciones intermedias
- No son realmente objetos
- ...

COLECCIONES

Java provee todo un API de colecciones con decenas de interfaces y clases.

Beneficios

- Menos esfuerzo de programación.
- Aumento de la calidad y velocidad.
- Interoperabilidad
- Curva de aprendizaje pequeña
- Reusabilidad.

ARRAYLIST

- De todas las colecciones de Java es quizá la más usada.
- Estructura de datos secuencia

Operaciones

- Acceso posicional
- Búsqueda
- Iteración
- Tomar un fragmento

CONSTRUCCIÓN DE UN ARRAYLIST

ArrayList()
- Construye un *arraylist* con capacidad para 10 elementos.

ArrayList(Collection c)
- Construye un *arraylist* a partir de otra colección.

ArrayList(int initialCapacity)
- Construye un *arraylist* indicando la capacidad inicial.

El primer constructor sin parámetros con una capacidad 10 elementos, se puede modificar con la llamada a un ArrayList especificando su tamaño inicial y existe una interfaz llamada Collection y podemos construir un Arraylist a partir de otra colección. En las primeras versiones de java un ArrayList podía tener cualquier objeto (en el mismo array). Ahora:

CONSTRUCCIÓN DE UN ARRAYLIST

A partir de Java 1.5
- Inclusión de los genéricos
- Permiten parametrizar el tipo

```
List<String> cars = new ArrayList<String>();
```

A partir de Java 1.7
- Operador *diamond*
- Nos ahorra indicar dos veces el tipo

```
List<String> cars = new ArrayList<>();
```

Con esta última forma, con el operador diamante, estamos creando un ArrayList de String.

ALGUNOS MÉTODOS DE ARRAYLIST

Nombre	Uso
add	Añade un elemento al final la lista
addAll	Añade todos los elementos de la colección pasada como argumento
clear	Elimina todos los elementos de la lista.
contains	Comprueba si un elemento está o no en la lista
get	Devuelve el elemento de la posición especificada de la lista
isEmpty	Verifica si la lista está vacía
remove	Elimina un elemento de la lista
size	Devuelve el número de elementos de la lista
toArray	Devuelve la lista como un array

Vamos a ver un ejemplo, con una agenda de personas. Creamos la clase persona o usamos la que ya tenemos, con nombre, apellidos y teléfono. Para hacer el ejemplo sencillo, lo vamos a hacer todo lo demás en el main. Un menú que nos va a permitir trabajar con los métodos vistos en la tabla. Creamos un array estático, para usarlo en todos los métodos, esto no es necesario, pero en este caso así lo usamos en todos los métodos que vamos a tener en la misma clase donde tenemos main.

Dentro de main, inicializamos el array.

```
//Declaramos estas dos referencias como estáticas, para poder usarlas en todos los métodos
static Scanner sc;
static ArrayList<Persona> listaPersonas;

public static void main(String[] args) {

    //Inicializamos la lista y la lectura por teclado
    listaPersonas = new ArrayList<>();
    sc = new Scanner(System.in);
    int opcion;

    do {
```

En un do While, tenemos un menú que llama a cada método del menú:

```
switch (opcion) {
case 1:
    listarPersonas();
    break;
case 2:
    aniadirPersona();
    break;
case 3:
    eliminarPersona();
    break;
case 4:
    eliminarTodas();
    break;
```

Por ejemplo, para añadir una persona, un elemento nuevo al arrayList:

```
/*
 * MÉTODO QUE RECOGE LOS DATOS DEL USUARIO
 * PARA AÑADIR UNA NUEVA PERSONA, Y LA INSERTA EN LA LISTA
 */
public static void aniadirPersona() {
    System.out.println("\n\nAÑADIR NUEVO CONTACTO");
    System.out.print("Introduzca el nombre: ");
    String nombre = sc.nextLine();
    System.out.print("Introduzca los apellidos: ");
    String apellidos = sc.nextLine();
    System.out.print("Introduzca su número de teléfono: ");
    String telefono = sc.nextLine();

    listaPersonas.add(new Persona(nombre, apellidos, telefono));

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

Para recorrer el array, una vez que he insertado varias personas:

```
 */
public static void listarPersonas() {
    if (listaPersonas.isEmpty()) {
        System.out.println("La agenda no tiene contactos\n");
    } else {
        for (int i = 0; i < listaPersonas.size(); i++) {
            Persona p = listaPersonas.get(i);
            System.out.printf("%d %s %s (%s) %n", i, p.getNombre(), p.getApellidos(), p.getTelefo
        }
        System.out.println("");
    }
}
```

En este caso está con un for, pero puedo hacerlo con un foreach. Pero si quiero indicar la posición de los distintos elementos en el array, lo hago con el for.

Para eliminar una persona, pidiendo la posición:

```
/*
 * MÉTODO QUE ELIMINA UNA PERSONA DE LA AGENDA
 * EN FUNCIÓN DE UNA POSICIÓN RECOGIDA DEL TECLADO
 */
public static void eliminarPersona() {
    System.out.println("\n\nELIMINAR CONTACTO");
    System.out.print("Introduzca la posición del contacto: ");
    int posicion = Integer.parseInt(sc.nextLine());
    if (posicion < 0 || posicion >= listaPersonas.size()) {
        System.out.println("Posición erronea");
    } else {
        System.out.print("¿Está usted seguro de querer eliminar el contacto? (S/N): ");
        String siono = sc.nextLine();
        if (siono.equalsIgnoreCase("S")) {
            listaPersonas.remove(posicion);
        }
    }
    System.out.println("");
```

Si quiero vaciar la lista:

```
    }

    /*
     * MÉTODO QUE ELIMINA TODOS LOS CONTACTOS DE LA AGENDA
     * PREVIA CONFIRMACIÓN DE LA OPERACIÓN
     */
    public static void eliminarTodas() {
        System.out.println("\n\nELIMINAR CONTACTO");
        System.out.print("¿Está usted seguro de querer eliminar el contacto? (S/N): ");
        String siono = sc.nextLine();
        if (siono.equalsIgnoreCase("S")) {
            listaPersonas.clear();
        }
        System.out.println("");

    }
```