Introducción a Arrays (Arreglos)

¡Comencemos!

¿Qué aprenderás?

- Reconocer los arrays en el lenguaje de programación para sus distintas implementaciones y uso.
- Construir programas utilizando arrays para manejar y tratar la información en volúmenes de datos.

Introducción

En este capítulo conoceremos e implementaremos los arreglos, sus diferentes formas de declaración, inicialización y llenado.

Conocer los arreglos es necesario para adentrarnos en el mundo de las base de datos que, a nivel de lógica, es lo más cercano a manejar grandes volúmenes de información. Para lograr este manejo de información aprenderemos distintos métodos que nos ayudarán a implementar los arreglos de manera oportuna y efectiva consiguiendo un mejor trato de este tipo de datos.

¡Vamos con todo!

¿Para qué sirven los Arrays?

Los arrays se utilizan mucho dentro de la programación. Java trata un arreglo como una variable normal, esto quiere decir que se declara, inicializa y utiliza. Los corchetes [ ] le indican al compilador que esa variable es un arreglo de un tipo de dato en específico. Nos permite resolver diversos tipos de problemas.

Ejemplo:

```
int [] a = {2,4,5,6};
```

```
String[ ] nombre = {"Juan", "Pedro"};   //Array de 2 elementos
```

Algunos posibles usos son:

- En una aplicación web podemos traer los datos de la base de datos en un array y luego mostrar los datos en la página.
- Obtener los datos desde una API (Aplication Programming Interface), los datos devueltos de una API pueden venir como una colección y podemos guardarlos en una base de datos o archivos.
- Traer información guardada en uno o más archivos.

Tipos de arreglos

En Java existen dos clasificaciones para los arreglos:

- **De tipo de datos primitivos (int, char ,double, boolean…):** Son arreglos de tamaño estático, al cual se le define el tamaño al inicializarlo, y pueden almacenar tipos de datos primitivos como objetos. Se almacena en la memoria stack.
- **De tipo objeto:** Java a ese tipo de variables las trabaja como objeto y no como variable de dato primitivo, estas variables se almacenan en la memoria HEAP. En tiempo de ejecución de nuestro programa pueden ir cambiando su tamaño (agregando o quitando elementos). Estos son algunos tipos de Arrays: Arraylist, LinkedList, HashSet, List, entre otros.

Creando un Array

Para crear un array utilizaremos la siguiente sintaxis dependiendo de cada caso.

Estáticos

Para definir una variable como array, debemos especificar el tipo de dato, los corchetes y el nombre de la variable.

Al momento de definir el array, los corchetes pueden estar antes o después del nombre de la variable.

```
int[] a;
int b[];
```

Para definir el tamaño del array, ahora debemos decirle de qué tamaño será dentro de los corchetes.

```
a = new int [4]; //array llamado a, de tipo enteros, de tamaño 4
```

Para introducir un valor podemos escribir:

```
a[0] = 4; // en la posición 0 agrega el valor 4
```

O bien, al momento de declarar un arreglo, asignarle los valores iniciales:

```
int[] ba = {2,4,5,6};//arreglo llamado ba, de tipo enteros, tamaño 4
```

Dinámicos

Para el caso de crear un arreglo de tipo dinámico, usaremos `ArrayList`, agregando previamente la librería `import java.util.ArrayList;`

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

Índices

Cada elemento del arreglo tiene una posición determinada, a la cual se le denomina índice. El índice nos permite acceder al elemento que está dentro del arreglo.

Por ejemplo, tenemos:

```
System.out.printf("%d\n", a[0]) // 1
```

Si queremos acceder al primer elemento, debemos acceder a la posición 0.

```
System.out.printf("%d\n", a[0]) // 1
```

Recorrido

Para recorrer un array se debe ocupar sentencias de bucle (for, for each, While), esto se recorre desde 0 hasta `n-1` , donde `n` es el tamaño del arreglo.

Con la propiedad `length` de un arreglo de corchetes podemos obtener el tamaño:

```
int i;
int[] a = {1,2,3,4,5};
int n = a.length;
for(i=0;i<n;i++){
 System.out.printf("%d\n",a[i]);
}
```

Salida

```
1
2
3
4
5
```

Tipos de errores con los índices en un Arrays

Los errores más comunes cuando se trabaja con arrays son los de tipo índices, esto quiere decir que se intenta agregar más elementos a un arrays definido y se accede a un índice que no existe; como por ejemplo un índice negativo.

```
int[] a = {1,2,3,4,5}; System.out.printf("%d\n", a[5]);
```

Pero al ejecutar el código obtenemos el error que nos dice que el índice 5 está fuera del límite del arreglo.

```
------------------------------------------------------------------------
---
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for
length 5 at .(#33:1)
```

¿Y si colocamos índices negativos?

```
System.out.printf("%d\n", a[-1]);
```

Ocurre el mismo error:

```
------------------------------------------------------------------------
---
 java.lang.ArrayIndexOutOfBoundsException: Index -1 out of bounds for
length 5 at .(#34:1)
```

Ejercicio guiado: Sumar dentro de un arreglo

Construir un programa el cual nos permita sumar los valores que estén entre 1 y 10 dentro de un arreglo.

Paso 1: Creamos un método llamado Suma que retorna un número entero de la suma.

```
// Paso 1
public static int suma() {

}
```

Paso 2: Dentro del método suma, declaramos una variable local llamada suma que parte en cero.

```
public static int suma() {
    //Paso 2
    int suma = 0;
}
```

Paso 3: Inicializamos el arreglo con valores aleatorios.

```
public static int suma() {
    int suma = 0;
    //Paso 3
    int[] arreglo = { 1, 5, 11, 33, 4, 6, 7, 44, 6, 1, -1 };
}
```

Paso 4: Recorremos el arreglo con un ciclo `for`.

```
public static int suma() {
    int suma = 0;
    int[] arreglo = { 1, 5, 11, 33, 4, 6, 7, 44, 6, 1, -1 };
    //Paso 4
    for (int x = 0; x < arreglo.length; x++) {
    }
}
```

Paso 5: Dentro del ciclo, realizamos condición `if` donde preguntamos por los valores que están en el intervalo solicitado.

```
public static int suma() {
    int suma = 0;
    int[] arreglo = { 1, 5, 11, 33, 4, 6, 7, 44, 6, 1, -1 };
    for (int x = 0; x < arreglo.length; x++) {
        //Paso 5
        if (arreglo[x] >= 1 && arreglo[x] <= 10) {

        }
    }
}
```

Paso 6: Si la condición se cumple, sumará todos los valores.

```
public static int suma() {
    int suma = 0;
    int[] arreglo = { 1, 5, 11, 33, 4, 6, 7, 44, 6, 1, -1 };
    for (int x = 0; x < arreglo.length; x++) {
        if (arreglo[x] >= 1 && arreglo[x] <= 10) {
           //Paso 6
           suma = suma + arreglo[x];
        }
    }
}
```

Paso 7: Este método lo llamamos dentro del método `main`.

```
public static void main(String[] args) {
    //Paso 7
    System.out.println("La suma es: " + suma());
}
```

Finalmente el ejercicio quedaría de la siguiente manera:

```
public static void main(String[] args) {
            System.out.println("La suma es: " + suma());
      }

      public static int suma() {
            int suma = 0;
```

```
        int[] arreglo = { 1, 5, 11, 33, 4, 6, 7, 44, 6, 1, -1 };
        for (int x = 0; x < arreglo.length; x++) {
            if (arreglo[x] >= 1 && arreglo[x] <= 10) {
                suma = suma + arreglo[x];
            }
        }
        return suma;
    }
```