PROGRAMACIÓN EN JAVA

Materia PJA-001 material creado por el Ing. Jim Requena

Fundamentos de Java I (tercera parte - arreglos)

Materia PJA-001 material creado por el Ing. Jim Requena

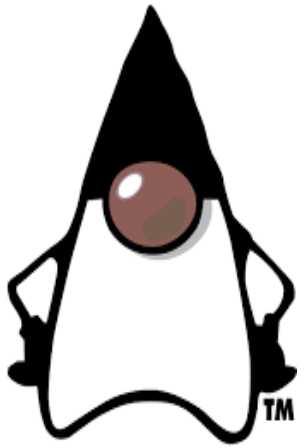
Objetivos de la sesión

- Comprender qué son los arrays en Java
- Aprender a declarar y inicializar arrays
- Practicar el acceso y modificación de elementos en arrays
- Trabajar con arrays multidimensionales
- Utilizar los métodos de la clase Arrays

¿Qué es un Array?

- Estructura de datos que almacena múltiples valores del mismo tipo
- Los elementos se almacenan en posiciones contiguas de memoria
- El índice del primer elemento es 0
- Tiene un tamaño fijo una vez creado

Declaración de Arrays



Sintaxis :

```
tipo[] nombreArray;
// o
nombreArray[];
```

Ejemplos :

```
int[] numeros;

String[] nombres;
```

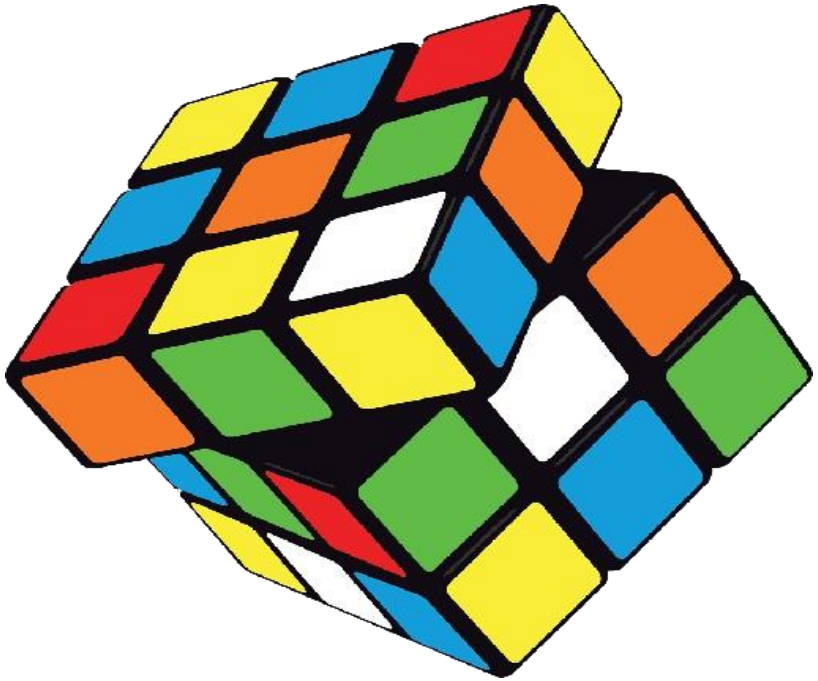
Inicialización de Arrays

1. Inicialización con tamaño:

```
int[] numeros = new int[5];
```

2. Inicialización con valores:

```
int[] numeros = {1, 2, 3, 4, 5};
```



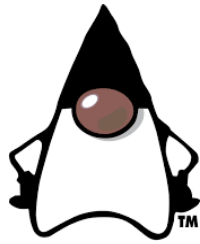
3. Inicialización combinada:

```
String[] dias = new String[]{"Lun", "Mar", "Mié", "Jue", "Vie"};
```

Acceso y modificación de elementos

- Se accede a los elementos mediante su índice
- Los índices van de 0 a (longitud - 1)

```
int[] numeros = {10, 20, 30, 40, 50};

System.out.println(numeros[0]); // Imprime 10
numeros[2] = 35; // Modifica el tercer elemento

System.out.println(numeros[2]); // Imprime 35
```



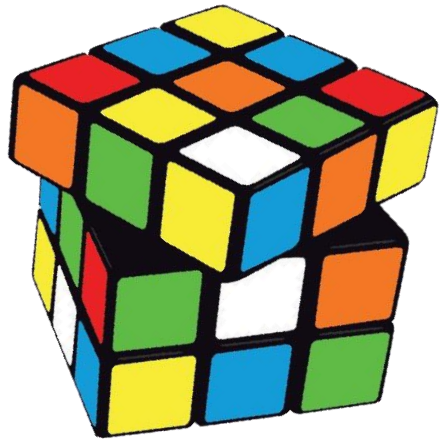
Recorriendo Arrays

Usando un bucle for:

```
int[] numeros = {1, 2, 3, 4, 5};
for (int i = 0; i < numeros.length; i++) {
        System.out.println(numeros[i]);
}
```

Usando for-each:

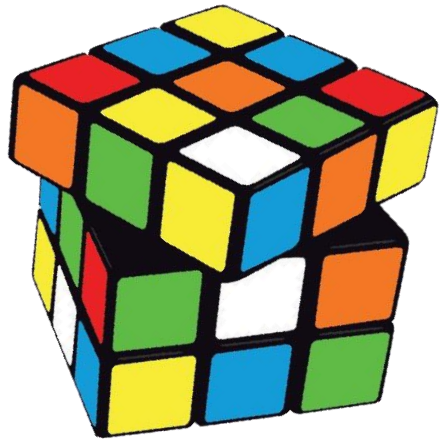
```
for (int numero : numeros) {
        System.out.println(numero);
 }
```



Arrays Multidimensionales

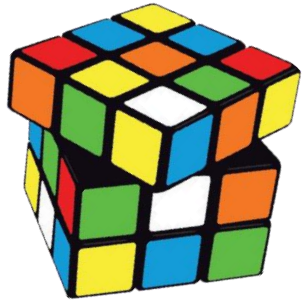
Son arrays de arrays. Ejemplo de array bidimensional:

```
int[][] matriz = {
                    {1, 2, 3},
                    {4, 5, 6},
                    {7, 8, 9}
};
System.out.println(matriz[1][2]); // Imprime 6
```



Clase Arrays

Java proporciona la clase `java.util.Arrays` con métodos útiles:



```java
import java.util.Arrays;

int[] numeros = {5, 2, 8, 1, 9};
Arrays.sort(numeros); // Ordena el array
System.out.println(Arrays.toString(numeros)); // Imprime [1, 2, 5, 8, 9]

int indice = Arrays.binarySearch(numeros, 5); // Busca el valor 5
System.out.println("El 5 está en el índice: " + indice);
```

Ejercicio práctico

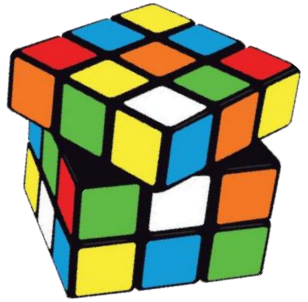
Crear un programa que:

1. Pida al usuario un número
2. Use un bucle para imprimir la tabla de multiplicar de ese número (del 1 al 10)
3. Pregunte al usuario si quiere ver otra tabla (S/N)
4. Si el usuario responde 'S', repita el proceso

Solución al ejercicio práctico tablas

```java
import java.util.Scanner;

public class TablaMultiplicar {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		String respuesta;
		do {
			// 1. Pedir al usuario un número
			System.out.print("Ingrese un número para ver su tabla de multiplicar: ");

			int numero = scanner.nextInt();

			// 2. Imprimir la tabla de multiplicar del 1 al 10
				System.out.println("Tabla de multiplicar del " + numero + ":");
			for (int i = 1; i <= 10; i++) {
				System.out.println(numero + " x " + i + " = " + (numero * i));
			}

			// 3. Preguntar al usuario si quiere ver otra tabla
			System.out.print("¿Quiere ver otra tabla? (S/N): ");
			respuesta = scanner.next();

		} while (respuesta.equalsIgnoreCase("S"));
			// 4. Repetir si el usuario responde 'S'
			System.out.println("¡Gracias por usar el programa!");
			scanner.close();
	}
}
```

Primer Ejercicio práctico



Crear un programa que:

1. Pida al usuario que ingrese 5 nombres

2. Almacene estos nombres en un array

3. Ordene el array alfabéticamente

4. Imprima los nombres ordenados

5. Pida al usuario que busque un nombre y diga si está en la lista

Solución del ejercicio

```java
import java.util.Arrays;
import java.util.Scanner;

public class OrdenarBuscarNombres {
		public static void main(String[] args) {
			Scanner scanner = new Scanner(System.in);
			String[] nombres = new String[5];
			// Ingresar nombres
			for (int i = 0; i < 5; i++) {
				System.out.print("Ingrese el nombre " + ( +1) + ": ");
				nombres[i] = scanner.nextLine();
			}
			// Ordenar e imprimir
			Arrays.sort(nombres);
			System.out.println("Nombres ordenados: " + Arrays.toString(nombres));
			// Buscar nombre
			System.out.print("Ingrese un nombre a buscar: ");
			String buscar = scanner.nextLine();
			 int indice = Arrays.binarySearch(nombres, buscar);

			 if (indice >= 0) {
				System.out.println(buscar + " está en la posición " + (indice + 1));
			} else {
				System.out.println(buscar + " no está en la lista");
			}
		}
}
```

Segundo Ejercicio práctico modificado

Modificar el programa del ejercicio práctico para:

1. Permitir al usuario elegir cuántos nombres ingresar
   - Se pide cuántos nombres se va ingresar
   - Luego debe usar un bucle para solicitar nombre por nombre
2. Cree un menú Debe brindar la opción para eliminar un nombre de la lista:
   - Para esto debe agregar una tercera opción de menú
   - Use el método remove() de ArrayList para eliminar el nombre que ingreso por línea de comando.
3. Implemente un nuevo método para encontrar al nombre más largo:
   - Para esto debe agregar una cuarta opción de menú
   - Debe nombrar al método como: “encontrarNombreMasLargo() para esto use streams para encontrar el nombre más largo

Cambios adicionales:

-Ya no use un array fijo en su lugar use un ArrayList para tener mayor flexibilidad.

-Implemente un menú con un bucle while para permitir múltiples operaciones.

-Debe usar String::compareTo para ordenar los nombres, que es más eficiente que el Array.sort() para Strings, que vimos en clases

-Por último no olvide usar indexOf() en lugar de binarySearch(), ya que la lista no siempre estará ordenada

Tarea

1. Modifica el programa para que permita al usuario elegir cuántos nombres quiere ingresar
2. Agrega una opción para eliminar un nombre de la lista
3. Implementa una función para encontrar el nombre más largo en la lista
4. Sube tu solución al repositorio del curso

Recursos adicionales

Recursos adicionalesDocumentación oficial de Java sobre Arrays:
docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

Métodos de la clase Arrays:
docs.oracle.com/javase/8/docs/api/java/util/Arrays.html

¡Gracias!

Explicación tarea ejercicio práctico modificado

Explicación de las modificaciones:

1. Vamos a permitir al usuario elegir cuántos nombres ingresar
    - Al inicio del programa, se pide al usuario que especifique cuántos nombres desea ingresar.
    - Se usa un bucle for para solicitar esa cantidad exacta de nombres.
2. Opción para eliminar un nombre de la lista:
    - Se agregó una nueva opción en el menú (opción 3).
    - Utiliza el método remove() de ArrayList para eliminar el nombre especificado.
3. Función para encontrar el nombre más largo:
    - Se implementó el método encontrarNombreMasLargo() que usa streams para encontrar el nombre más largo.
    - Se agregó como opción 4 en el menú.

Cambios adicionales:

- Se cambió de usar un array fijo a un ArrayList para mayor flexibilidad.
- Se implementó un menú con un bucle while para permitir múltiples operaciones.
- Se usa String::compareTo para ordenar los nombres, que es más eficiente que Arrays.sort() para Strings.
- La búsqueda ahora usa indexOf() en lugar de binarySearch(), ya que la lista no siempre estará ordenada.

Explicación del error no manejado de `InputMismatchException`

Este error ocurre cuando el tipo de entrada que el programa espera no coincide con lo que el usuario ingresa. En este caso, probablemente sucedió cuando el programa esperaba un número entero y recibió algo diferente.

Vamos a modificar el código para manejar este error y hacer el programa más robusto:

Explicación de los cambios:

1. Manejo de excepciones:
   - Se creó un método leerEntero() que maneja la excepción NumberFormatException que ocurre cuando se ingresa algo que no es un número.
   - Este método se usa en lugar de scanner.nextInt() para leer enteros de forma segura.

2. Uso de scanner.nextLine():
   - Se cambió todo uso de scanner.nextInt() por scanner.nextLine() seguido de Integer.parseInt().
   - Esto evita problemas con el buffer del scanner y hace la lectura de entrada más consistente.

3. Estructura del código:
   - Se dividió el código en métodos más pequeños para mejorar la legibilidad y mantenibilidad.
   - Cada opción del menú ahora tiene su propio método.

4. Validaciones adicionales:
   - Se agregó una comprobación en encontrarNombreMasLargo() para manejar el caso de una lista vacía.

5. Uso de variables de clase:
   - scanner y nombres se convirtieron en variables de clase para evitar pasarlas como parámetros repetidamente.

Solución error InputMismatchException parte 1

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

public class OrdenarBuscarNombres {
            private static Scanner scanner = new Scanner(System.in);
            private static ArrayList<String> nombres = new ArrayList<>();

            public static void main(String[] args) {
                        ingresarNombres();
                        mostrarMenu();
            }
private static void ingresarNombres() {
            int cantidadNombres = leerEntero("¿Cuántos nombres desea ingresar? ");


            for (int i = 0; i < cantidadNombres; i++) {
                        System.out.print("Ingrese el nombre " + (i + 1) + ": ");
                        nombres.add(scanner.nextLine());
            }
         }

private static void mostrarMenu() {
        while (true) {
            System.out.println("\nOpciones:");
            System.out.println("1. Ordenar y mostrar nombres");
            System.out.println("2. Buscar un nombre");
            System.out.println("3. Eliminar un nombre");
            System.out.println("4. Encontrar el nombre más largo");
            System.out.println("5. Salir");

int opcion = leerEntero("Elija una opción: ");
```

Solución error InputMismatchException parte 2

```
switch (opcion) {
        case 1: ordenarYMostrar(); break;
        case 2: buscarNombre(); break;
        case 3: eliminarNombre(); break;
        case 4: encontrarNombreMasLargo(); break;
        case 5:
            System.out.println("Gracias por usar el programa.");
            return;
        default:
            System.out.println("Opción no válida. Por favor, intente de nuevo.");
    }
  }
 }

 private static int leerEntero(String mensaje) {
    while (true) {
            try {
                System.out.print(mensaje);
                return Integer.parseInt(scanner.nextLine());
            } catch (NumberFormatException e) {
                System.out.println("Por favor, ingrese un número válido.");
            }
    }
  }

  private static void ordenarYMostrar() {
            nombres.sort(String::compareTo);
            System.out.println("Nombres ordenados: " + nombres);
  }
```

Solución error InputMismatchException parte 3

```
private static void buscarNombre() {
        System.out.print("Ingrese un nombre a buscar: ");
        String buscar = scanner.nextLine();
        int indice = nombres.indexOf(buscar);
        if (indice >= 0) {
                System.out.println(buscar + " está en la posición " + (indice + 1));
        } else {
                System.out.println(buscar + " no está en la lista");
        }
}

private static void eliminarNombre() {
        System.out.print("Ingrese el nombre a eliminar: ");
        String eliminar = scanner.nextLine();
        if (nombres.remove(eliminar)) {
                System.out.println(eliminar + " ha sido eliminado de la lista");
        } else {
                System.out.println(eliminar + " no se encontró en la lista");
        }
}

private static void encontrarNombreMasLargo() {
        if (nombres.isEmpty()) {
                System.out.println("La lista está vacía");
                return;
        }
        String nombreMasLargo = nombres.stream()
                .max((a, b) -> Integer.compare(a.length(), b.length()))
                .orElse("");
        System.out.println("El nombre más largo es: " + nombreMasLargo);
   }
 }
```