

# ESTRUCTURAS DE DATOS EN JAVA

# INDICE

- Tipos abstractos de datos (TAD).
- Almacenamiento de la información.
- Estructuras estáticas de datos.
- Estructuras dinámicas de datos

# TIPOS ABSTRACTOS DE DATOS

- En toda aplicación de software, podemos destacar dos aspectos importantes: la interfaz del usuario y el funcionamiento interno de la aplicación.
  - La interfaz es importante a corto plazo, ya que su función es ser atractiva para el usuario.
  - Por el contrario, a largo plazo, necesitamos que las operaciones y estructuras encargadas de almacenar la información, perduren en el tiempo.
- Con el fin de modelar el funcionamiento interno, usamos objetos abstractos que incluyan operaciones y estructuras en entes autónomos, autosuficientes y cerrados. Estos entes se conocen como **tipos abstractos de datos (TAD)**.

# ALMACENAMIENTO DE LA INFORMACIÓN

- En Java existen diferentes formas de almacenar la información:
  - ✓ **Variables:** Referencian pequeñas áreas de memoria, donde se almacenan algunos datos. Según el tipo de dato que almacenen: primitivas o de referencia.
  - ✓ **Estructuras:** Permiten almacenar distintos tipos de datos en múltiples dimensiones, tamaños, formas y comportamientos diferentes:
    - Según su tamaño: Estáticas o dinámicas.
    - Según la relación entre sus elementos: Lineales o no.

# ESTRUCTURAS ESTÁTICAS

## ➤ Enumerados

- Son un tipo especial de tipo de dato que nos permite asignarle un valor predefinido constante a una variable.
- Estas variables tienen que tener un valor predefinido.
- Al ser valores constantes, los valores son escritos en mayúsculas.

## ➤ Definición:

modificadorAcceso enum NombreVariable {VALOR1,...VALORN}

## ➤ Ejemplo:

```
public enum semaforo {ROJO,AMARILLO,VERDE};
```

# ESTRUCTURAS ESTÁTICAS

## ➤ Enumerados. Ejemplo:

```
public enum semaforo {ROJO,AMARILLO,VERDE};  
public static void main(String[] args)  
{  
    semaforo s = semaforo.ROJO;  
    System.out.println(s.ROJO);  
    System.out.println(s);  
    for (semaforo s2: semaforo.values())  
        System.out.println(s2);  
}
```

# ESTRUCTURAS ESTÁTICAS

## ➤ **Enumerados. Ejercicios:**

### ➤ Crea y muestra los valores de:

- 1) Un enumerado de los días de la semana.
- 2) Un enumerado de los meses del año.
- 3) Un enumerado de tipos de vehículos de tierra que conozcas.
- 4) Un enumerado de la principales marcas de coches.
- 5) Un enumerado de los principales miembros sanguíneos de una familia.

# ESTRUCTURAS ESTÁTICAS

- Tenemos que indicar su tamaño y/o sus valores en tiempo de compilación.
- **Arrays**
  - Podemos definirlo como un conjunto de variables del mismo tipo que se acceden mediante un nombre común.
  - Cada una de estas variables tiene una determinada posición dentro del array, a la cuál denominados **índice**.
  - La posición de los elementos dentro del array se empieza por el número 0.
- **Matrices**
  - Son arrays de dos dimensiones.
  - El primer array representa las filas y el segundo las columnas.



# Definición

## ➤ Arrays

tipoDato nombreVariable [] = {valor1,...valorN};

tipoDato nombreVariable [] = new tipoDato[tamaño];

## ➤ Matrices

tipoDato nombreVariable [][] = {{vf11,...,vf1N},{vf21,...,vf2N},...};

tipoDato nombreVariable [][] = new tipoDato[Nfiles][Ncols];

## ➤ Ejemplo:

int vector[] = {1,2,3,4};

int matriz[][] = {{1,2},{3,4},{5,6}};

# Definición y uso de arrays

```
int m = 5;  
int [] a = new int[5];
```

0	0	0	0	0
a[0]	a[1]	a[2]	a[3]	a[4]

```
a[1] = 2;
```

0	2	0	0	0
a[0]	a[1]	a[2]	a[3]	a[4]

```
a[2] = a[1];
```

0	2	2	0	0
a[0]	a[1]	a[2]	a[3]	a[4]

```
a[0] = a[1] + a[2] + 2;
```

6	2	2	0	0
a[0]	a[1]	a[2]	a[3]	a[4]

```
a[0]++;
```

7	2	2	0	0
a[0]	a[1]	a[2]	a[3]	a[4]

```
int m = 5;  
a[3] = m + 10;
```

7	2	2	15	0
a[0]	a[1]	a[2]	a[3]	a[4]

Acceso:

- Válido: posición desde 0 a tamaño -1.
- Inválido: Fuera de rango (out of bounds), menor que 0 o mayor/igual que tamaño.

## Acceso a los elementos de los arrays

Lo realizaremos mediante una estructura de control tipo for, generalmente.

Ejemplo:

```
int ej1 [] = {2, 8, 3, 9, 4, 5, 7, 1, 6, 0};  
for (int i = 0; i < 10; ++i)  
{  
    System.out.print(ej1[i] + " ");  
}
```

## Definición y uso de arrays

Una matriz es un array de dos dimensiones, por lo que necesitaremos dos índices (n filas y m columnas para acceder a sus elementos.

Mis característica que con los arrays: monotipo, acceso,...

```
//int a [][] = new int[FILAS][COLUMNAS];  
int a [][] = new int[3][5];
```

	0	1	2	3	4
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15

## Acceso a los elementos de una matriz

Al igual que en los arrays, generalmente, lo realizaremos mediante una estructura de control tipo for anidada.

### Ejemplo:

```
//tanto para los arrays como las matrices, podemos acceder al atributo length para obtener la longitud de esa dimensión.  
for (i = 0; i < a.length; i++)// estructura de control para acceder a las filas  
{  
    for (j = 0; j < a[i].length; j++) // estructura de control para acceder a las columnas  
    {  
        System.out.print(a[i][j] + " ");  
    }  
    System.out.println();  
}
```