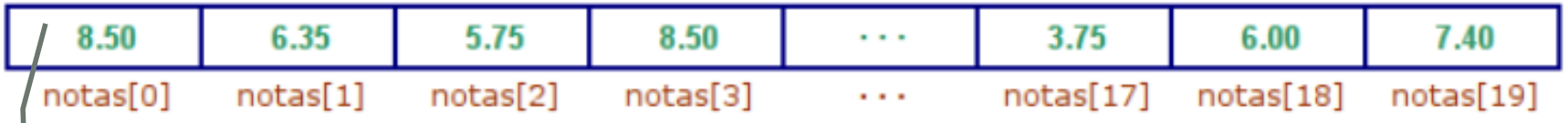
DAM
PROGRAMACIÓN

TIPOS ENUMERADOS.
ARRAYS

Arrays

- Un array es una variable que almacena una lista de valores del mismo tipo.
- El array almacena sus valores en posiciones contiguas y el acceso a ellos se realiza mediante índices.
- Es necesario determinar el número de elementos que va a tener un array en el momento de su declaración, ya que necesita reservar el espacio de memoria para todos sus elementos.
- El tamaño del array no puede ser modificado en tiempo de ejecución.
- Por ejemplo, supongamos que tenemos que almacenar las notas de 20 alumnos. Podemos representar gráficamente el array de notas de la siguiente forma:



Array **notas** de **20** posiciones que permite almacenar valores de tipo **double**

Array: declaración

- **Sintaxis de la declaración:**

```
tipo[ ] nombreVble = new tipo [tamaño];
                o
tipo nombreVble[ ] = new tipo [tamaño];
```

  - **tipo** un tipo de Java (double, int, String, etc.)
  - tamaño es el número de elementos que es capaz de almacenar el array.
  - **Ejemplo**: La declaración del array **notas** visto anteriormente.

```
double[ ] notas = new double [20];
```

- También se puede hacer la declaración el dos pasos:

```
tipo[ ] nombreVble;
nombreVble = new tipo [tamaño];
```

- **Se puede pedir el tamaño del array en tiempo de ejecución**, pero **nunca se puede cambiar** si ya se ha determinado su tamaño anteriormente.

```
Scanner sc = new Scanner(System.in);
System.out.print("Introduce el número de elementos del array: ");
int tamano = sc.nextInt();
int[] miArray = new int[tamano];
```

Array: inicialización

- Los arrays pueden inicializar su contenido en el momento de la declaración dando valor a sus elementos entre llaves.

```
double[] notas = {6.7, 7.5, 5.3, 8.75, 3.6, 6.5};
```

  - En esta declaración se declara un array de 6 elementos con un valor ***double*** establecido en cada posición.
- Otros ejemplos:

```
//creación de un array de 4 elementos booleanos
boolean[] aprobados = {true,false,true,false};

//creación de un array de 7 elementos de tipo String
String[] dias = {"Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"};
```

Array: acceso

- **Sintaxis del acceso a un array:**

```
nombreVble[ i ]
```

  - El índice i debe estar dentro del rango de *0..tamaño*.
  - **Ejemplo**: Para ver la nota del elemento tercero del ejemplo:

```
System.out.println ("La nota del tercer alumno es "+notas[2]);
```

Cada elemento del array puede estar implicado en las mismas operaciones que permite el tipo de elemento.

```
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]++;
```

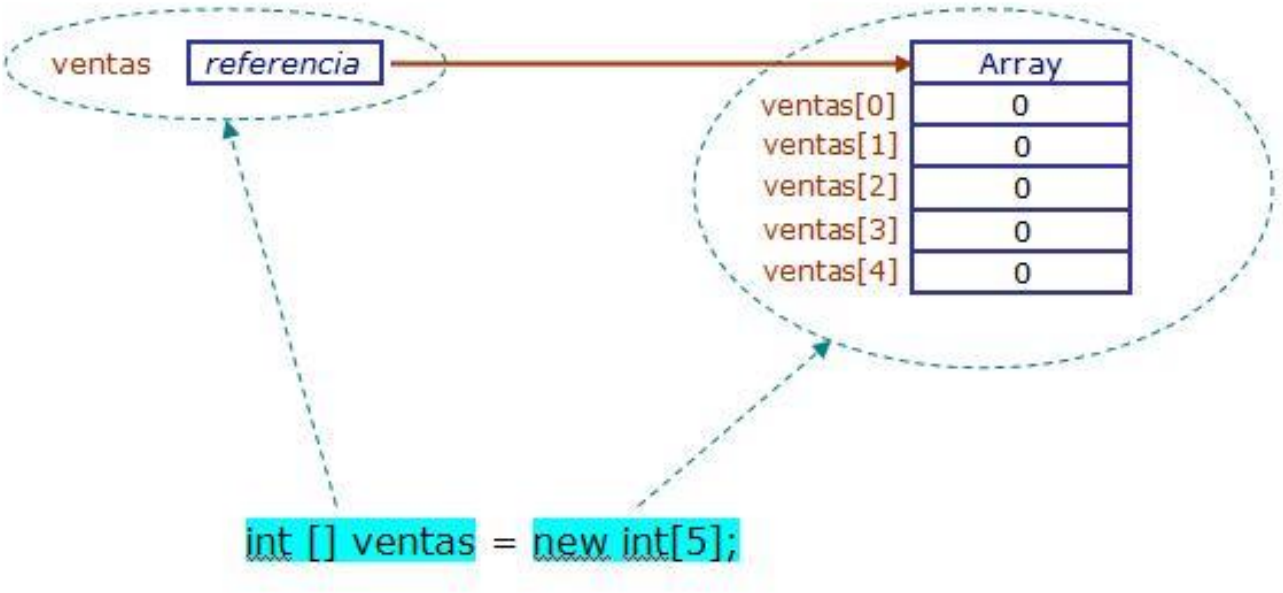
7	2	2	0	0
a[0]	a[1]	a[2]	a[3]	a[4]

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

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

Array: objeto

- Un array es un objeto, por lo tanto cuando se crea se utiliza una referencia que apunta al contenido en sí.



- Los array reciben una inicialización automática de cada uno de sus elementos, así:
  - Los tipos numéricos se inicializan a 0
  - Los tipos carácter se inicializan a '\u0000'
  - Los tipos boolean a false.
  - Los tipos String y objetos a null

Recorrer un array

- Al intentar acceder a un elemento del array que está fuera de sus límites, se produce la excepción **ArrayIndexOutOfBounds**.
- Por ello hay que tener mucho control sobre el índice del array.
- El tamaño de un array lo proporciona el siguiente método:

nombreVbleArray.length()

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

//Usando el for clásico
for (int i = 0; i <miArray.length; i++) {
    System.out.println(miArray[i]);
}

//Usando el foreach
for (int elemento : miArray) {
    System.out.println(elemento);
}
```

Recorrido de un array con un for

Recorrido de un array con un foreach. Permite mostrar sus elementos pero no realizar modificaciones ni acceder al índice.

Arrays multidimensionalse

- Las matrices multidimensionales son **arrays de arrays**, donde cada elemento del array contiene la referencia de otro array.
- Se crea una matriz multidimensional al agregar un conjunto de corchetes ([]) por dimensión.
- **Ejemplos:**

```
int[ ][ ] intArray = new int[tamano1][tamaño2]; //un array 2D vacío
int[ ][ ] intArray = { {2,7,9}, {3,6,1} }; //un array 2D inicializado
```

```
// declarar e inicializar array 2D
int arr[ ][ ] = {{2, 7, 9}, {3, 6, 1, 5}};
int indi=2, indj=3;

// imprimir array 2D
for (int i = 0; i < indi; i++) {
    for (int j = 0; j < indj; j++) {
        System.out.print(arr[i][j]+"-");
    }
    System.out.println();
}
```

```
2-7-9-
3-6-1-
```

Declaración, inicialización y recorrido de un array de tres dimensione

Para acceder a un elemento de un array multidimensional es necesario poner un índice por dimensión

```java
static void mediaNegativosPositivos(){
    Scanner miScanner = new Scanner(System.in);
    int i;
    int pos = 0, neg = 0; //contadores de los números positivos y negativos
    int[] numeros = new int[10]; //array que contendrá los números leídos por
teclado
    double sumaPos = 0, sumaNeg = 0; //acumuladores para las sumas de
positivos y negativos

    //lectura de datos y llenar el array
    System.out.println("Lectura de los elementos del array: ");
    for (i = 0; i < 10; i++) {
        System.out.print("numeros[" + i + "]= ");
        numeros[i]=miScanner.nextInt();
    }

    //recorrer el array para sumar por separado los números positivos
    // y los negativos
    for (i = 0; i < 10; i++) {
        if (numeros[i] > 0){ //sumar positivos
            sumaPos += numeros[i];
            pos++;
        } else if (numeros[i] < 0){ //sumar negativos
            sumaNeg += numeros[i];
            neg++;
        }
    }

    //Calcular y mostrar las medias
    if (pos != 0) {
        System.out.println("Media de los valores positivos: " + sumaPos / pos);
    } else {
        System.out.println("No ha introducido numeros positivos");
    }
    if (neg != 0) {
        System.out.println("Media de los valores negativos: " + sumaNeg / neg);
    } else {
        System.out.println("No ha introducido numeros negativos");
    }
}
```

Ejemplo

Método que lea por teclado 10 números enteros y los guarde en un array.
A continuación calcula y muestra la media de los valores positivos y la de los valores negativos.

Práctica 1

- Crea una clase con el método main donde declares una variable **de tipo array** de Strings que contenga los doce meses del año, en minúsculas y declarados en una sola línea. A continuación declara una variable mesSecreto de tipo String, y hazla igual a un elemento del array (por ejemplo mesSecreto = mes[9]. El programa debe pedir al usuario que adivine el mes secreto y si acierta mostrar un mensaje y si no pedir que vuelva a intentar adivinar el mes secreto. Puedes comprobar si tu código es correcto consultando en los foros aprenderaprogramar.com.
- Un ejemplo de ejecución del programa podría ser este:
  - Adivine el mes secreto. Introduzca el nombre del mes en minúsculas: febrero
  - No ha acertado. Intente adivinarlo introduciendo otro mes: agosto
  - No ha acertado. Intente adivinarlo introduciendo otro mes: octubre
  - ¡Ha acertado!

Práctica 2

- En España cada persona está identificada con un Documento Nacional de Identidad (DNI) en el que figura un número y una letra, por ejemplo 56999545W
- La letra que sigue al número se calcula siguiendo la metodología que vamos a indicar. Crea un programa que calcule la letra de un DNI a partir del número de DNI que introduzca el usuario. Es decir, se debe pedir el DNI sin la letra por teclado y el programa nos devolverá el DNI completo (con la letra).
- Para calcular la letra, se debe tomar el resto de dividir nuestro número de DNI entre 23. El resultado debe estar por tanto entre 0 y 22.
- Crea un método obtenerLetra(int numeroDNI) donde según el resultado de la anterior fórmula busque en un array de caracteres la posición que corresponda a la letra. Esta es la tabla de caracteres:
- Posición 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
- Letra T R W A G M Y F P D X B N J Z S Q V H L C K E
- Por ejemplo, si introducimos el DNI 20267079, el resto de dividirlo por 23 sería 8, luego la letra sería la P, que es la que ocupa esa posición en la matriz de caracteres.

Tipos enumerados

- Los tipos de datos enumerados son una forma de declarar una variable con un conjunto restringido de valores.
- Se declaran con la palabra reservada **enum**, seguida del

enum ClaseEnum **{valor1, valor2,…., valorN }**

- Un tipo **enum** es una ***clase especial*** que contiene un grupo de valores constantes y estáticos.
- Los valores son nombres significativos para el programador que no tienen ni tipo ni contenido.
- Para acceder a cada elemento de una variable declarada como **enum** se utiliza la siguiente sintaxis:

ClaseEnum**.valor1**

```
public class Principal_Enumerado {
    public enum Dias {LUNES, MARTES, MIÉRCOLES, JUEVES, VIERNES, SABADO, DOMINGO};

    public static void main(String[] args) {
        //Almacenar el valor de un enum en una variable
        Dias actual=Dias.LUNES;
        Dias festivo=Dias.DOMINGO;
        //Mostrar el valor de una vble enum
        System.out.println("Hoy es "+actual);
        System.out.println("El festivo es "+festivo);
    }
}
```

```
Hoy es Lunes
El festivo es Domingo
```

Tipos enumerados: comparación

- Las variables de tipo **enum** se pueden comparar de dos formas:
  - **==** → Es el método recomendado.
  - **equals()** → Usando este método se puede producir una excepción (***NullPointerException***) en el caso de que alguna de las variables no tenga un valor asignado.

```
static void comparar(){
    //Podríamos haber obtenido los valores de las variables de otros procesos
    Dias citaMedico=Dias.MARTES;
    Dias diaTrabajo=Dias. MARTES;

    //Puedo comparar con equals()
    //Si alguna de las vbles no tiene valor asignado se compararía con null y saltaría Exception
    if (diaTrabajo.equals(citaMedico)){
        System.out.println("Son iguales");
    }
    //Puedo comparar con ==
    if (diaTrabajo==citaMedico){
        System.out.println("Son iguales");
    }
}
```

Tipos enumerados: métodos

- **toString()**→ devuelve el literal del *enum*
  - **String *diaS*=Dias.LUNES.toString();** → ***diaS*** contiene **"*LUNES*"**
- **valueOf(String** literal**)**→ devuelve el *enum* correspondiente al literal que se pasa por parámetro.
  - **Dia *elDia*=Dias.valueOf("MARTES");** → ***elDia*** es = **Dia.MARTES**
- Podemos sobrescribir el método **toString()** para un tipo *enum* como se haría en una clase "*normal*".

```
public enum Precio {
   BARATO,
   MEDIO,
   CARO;

   public String toString() {
      return "soy un poco " + super.toString();
   }
}
```

Al escribir Precio.CARO, se mostrará "soy un poco CARO"

Ejemplo

```
public class Principal_Enun_ToString {

    public enum Color {
        ROJO,
        VERDE,
        AZUL;

        public String toString() {
            return "El color " + super.toString();
        }
    }

    public static void main(String[] args) {
        Color miColor=Color.AZUL;
        System.out.println(miColor);
    }

}
```

Redefinición del método toString() para un enumerado

Tipos enumerados: constructores y atributos

- **enum** se define como una clase y permite la implementación de constructores y atributos.
- **Por ejemplo**:

```
enum Precio {
    CARO(100, 1000),
    MEDIO(10, 100),
    BARATO(0, 10);

    private int precioMinimo;
    private int precioMaximo;

    private Precio(int minimo, int maximo) {
        precioMinimo = minimo;
        precioMaximo = maximo;
    }

    public int getPrecioMaximo() {
        return precioMaximo;
    }
    public int getPrecioMinimo() {
        return precioMinimo;
    }
}
```

Definición de la clase *enum*

Uso de la clase enum

```
public static void main(String[] args) {
        Precio precio = Precio.CARO;
        System.out.println("un precio " + precio + " esta entre "
                + precio.getPrecioMinimo() + " y "
                + precio.getPrecioMaximo());
    }
```

Ejemplo

```
static void usarSwitch() {
    Scanner miScanner = new Scanner(System.in);
    Dias dia;
    byte diaB=0;
    String diaS;
    boolean correcto;

    //Control de introdución de datos.  Pedir el dato hasta que se introduzca un nº del rango requerido
    do {
        correcto=true;
        System.out.println("Introduce el dia de la semana (0:Domingo..6: Sábado) ");
        diaS = miScanner.nextLine();
        try {
            diaB = Byte.valueOf(diaS);
            if (diaB < 0 || diaB > 6) {//Si el dato no está en el rango correcto
                correcto = false;
            }
        } catch (Exception e) {//Si el dato no es del tipo Byt
            correcto = false;
        }
        if (!correcto) {
            System.out.println("Valor no correcto. Vuelva a intentarlo");
        }
    } while (!correcto);

    //Dar valor al enum a partir del índice del array  que se correcponde con el nº del día introducid
    dia = Dias.values()[diaB];
    switch (dia) {//Según el valor del enum mostrar información por pantalla
        case DOMINGO:
            System.out.println("Festivo");
            break;
        case SABADO:
            System.out.println("Festivo");
            break;
        default:
            System.out.println("Dia");
            break;
    }
}
```

Pedir el día de la semana y mostrar por pantalla si es un día festivo o no.