Tema 5: Vectores

Introducción a la Programación
Grado en Ingeniería Informática, EPI Gijón

{jdiez,oluaces,juanjo}@uniovi.es

Departamento de Informática - Universidad de Oviedo en Gijón

Objetivos

- Saber emplear los vectores como estructura para la representación de una colección de datos.
- Entender la representación y utilización de vectores multidimensionales (matrices).
- Saber utilizar la clase String para el manejo de cadenas de caracteres.
- Ser capaz de diseñar programas simples que manejen vectores, cadenas de caracteres y matrices.

Contenidos

Las variables simples NO son suficientes

Necesitamos estructuras de datos más complejas

- Tanto una variable simple como un objeto de una clase nos permiten representar **un elemento de información**
- ¿Qué pasa si un programa necesita representar cientos de esos elementos de información?¿Declaramos tantas variables simples u objetos como datos tengamos?
- Los programas necesitan representar **colecciones** grandes de datos, lo que invalida usar una variable individual por cada dato.
- Son necesarias estructuras para organizar esas colecciones de datos. El objetivo último es facilitar el tratamiento del conjunto de datos dentro de los programas.

Ejemplos:

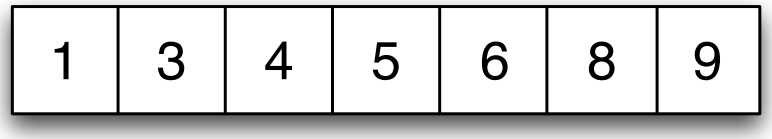
1. Almacenar la información sobre los clientes de una empresa.
2. Cotización del cambio Euro/Dólar todos los días de un año.

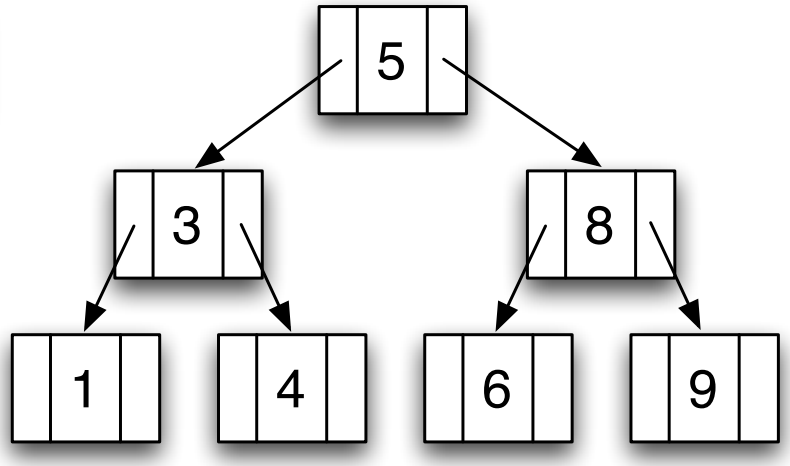
Estructuras de datos

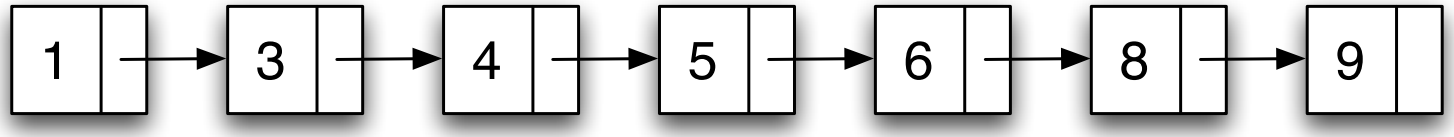
Una formas de organizar grupos de datos

Estructura de datos

Forma de organizar un conjunto de datos en la memoria con el objetivo de facilitar su tratamiento.







Estructuras de datos

Operaciones

Operaciones típicas:

1. Leer/Modificar: obtener o cambiar el valor de uno de los elementos de la estructura.
2. Insertar/Borrar: añadir un nuevo elemento o eliminar de la estructura un elemento ya existente.
3. Buscar: encontrar un determinado elemento.
4. Ordenar: todos los elementos de acuerdo a un criterio.
5. Mezclar: dadas dos estructuras, combinar sus elementos ordenadamente para crear una nueva.

Cada estructura ofrece ventajas y desventajas en relación a la simplicidad y eficiencia para la realización de esas operaciones.

Programa: Temperaturas

Enunciado

Enunciado

Realizar un programa que permita al usuario introducir las temperaturas registradas durante las 24 horas del día, desde las 0 horas hasta las 23 horas, y calcule diversos parámetros como la temperatura máxima y media de ese día.

¿Qué es nuevo en este programa?

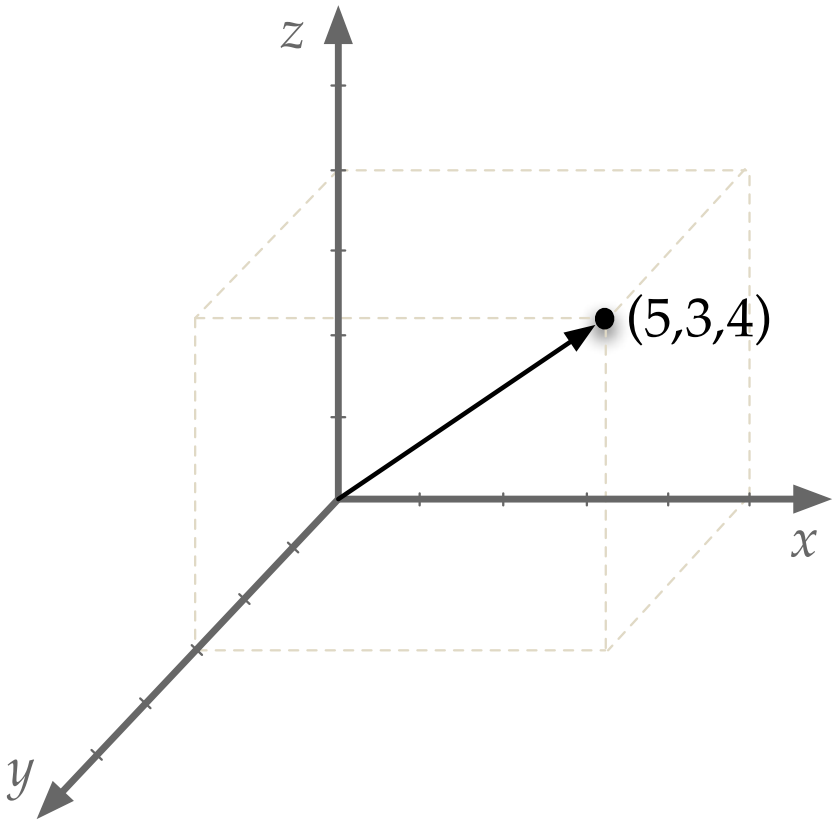
- Necesitamos representar 24 valores distintos, la temperatura de cada hora.
- ¿Sería lógico usar 24 variables diferentes?
- Obviamente NO, sería muy complicado hacer ciertas operaciones con todas ellas, por ejemplo calcular el máximo.
- Necesitamos guardar esos datos en una estructura más compleja que las variables simples, de manera que nos facilite realizar **tratamientos secuenciales** de todos esos datos.

Idea geométrica de vector

El valor de un punto en un espacio de varias dimensiones

Vector (Matemáticas)

Un vector de dimension n es una tupla de n números reales.

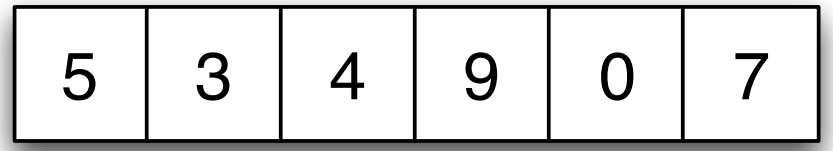


Vector (Programación)

Varios elementos de un mismo tipo

Vector (Programación)

Un vector de tamaño n es una tupla de n variables (llamadas elementos o componentes) del mismo tipo.



- Los vectores en Programación pueden guardar elementos de cualquier tipo de dato.
- Los elementos tienen que ser **todos** del mismo tipo.
- Además, los elementos tienen que estar relacionados entre sí. Es decir, no se trata de agrupar variables simples sin relación.

Ejemplo de temperaturas diarias: usaremos un vector que guardará las temperaturas de las 24 horas de un *mismo* día.

Definición y creación de un vector

Es un tipo referenciado

Declaración y creación de un vector

`tipo[ ] nombre` [= `new tipo` [*tamaño*]];

- El tipo de un vector es `tipo[ ]`, los `[ ]` representan vector.
- Los vectores son **objetos**: una cosa es la creación de la variable y otra la creación del objeto vector usando `new`.
- Al crear el objeto vector usando `new` se debe indicar el `tipo` de las componentes y su tamaño o número de elementos. El tamaño debe ser una expresión `int` entre corchetes.
- Las componentes en un vector de un tipo básico se inicializan a 0 (`false` para `boolean`) y si son objetos de una clase a `null`.

```
1    int[ ] v = new int [6];          //Vector de 6 enteros
2    double[ ] t;                     //Declaración vector de reales
3    t= new double [24];              //Creación vector de 24 reales
4    Fecha[ ] f= new Fecha [365];     //Vector de 365 objetos Fecha
```

Inicialización de un vector

Otra forma de declarar y reservar espacio para los vectores

Declaración, creación e inicialización de un vector

`tipo[ ] nombre` [= { *lista de valores* }];

- La *lista de valores* será una lista de expresiones del mismo tipo del vector separadas por comas y encerradas entre llaves.
- Automáticamente se reserva espacio en memoria para tantos elementos como valores contenga la *lista de valores* y el vector se referirá a ese espacio.

```
1    int[ ] v = { 5, 3, 4, 9, 0 ,7};              //Vector 6 enteros
2    char[ ] vocales= {'a','e','i','o','u'};      //Vector 5 vocales
3    Fecha[ ] f= {new Fecha(), new Fecha()};      //Vector 2 Fechas
```

Importante

El tamaño del vector será el mismo que el tamaño de la lista de valores indicada

Otra forma de declarar un vector

Una sintaxis diferente

Otra forma de declarar un vector

`tipo nombre[ ]` [= `new tipo` [*tamaño*]];
`tipo nombre[ ]` [= { *lista de valores* }];

- Los corchetes se pueden situar después del nombre del vector.
- Es útil para declarar a la vez variables simples y vectores.

```
int a=5, b[ ] = new int [4], c=7;
```

- Si se quieren declarar varios vectores habría que poner los corchetes después de cada nombre.

```
double d[ ]= new double[10], e[ ]= { 3.3 , 4.5 , 5.7};
```

Importante

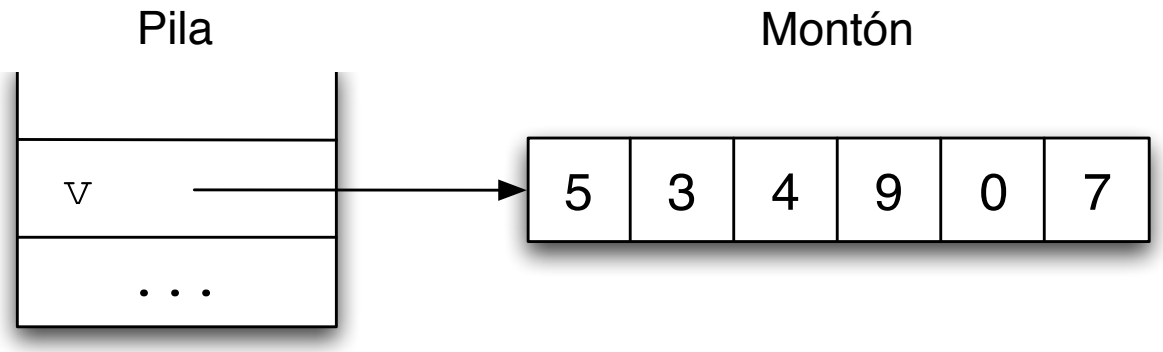
Es preferible declarar cada vector en un declaración y usar la sintaxis `tipo[ ]`

Representación en memoria

Es un tipo referenciado, las componentes se guardan en el Montón

Los vectores son objetos (tipos referenciados), eso implica dos cosas de cara a su representación:

1. Por un lado está la variable que representa todo el vector y que se guarda en la Pila. Se reserva cuando declaramos la variable.
2. Por otro lado está el objeto vector con sus componentes y se guarda en el Montón. Se reserva al hacer new o al inicializar el vector con una lista de valores.



Las componentes están contiguas en memoria

Se aceleran los accesos a los elementos del vector

Importante

Las componentes de un vector se guardan en posiciones consecutivas de la memoria

- Es clave que los elementos estén consecutivos porque es la manera de agilizar los accesos a esos datos.
- Dado que todas las componentes ocupan el mismo espacio en memoria (son del mismo tipo), y además sabemos donde empieza el vector, se puede determinar la posición exacta de cada dato en la memoria.

Importante

El espacio necesario para un objeto vector solo podrá reservarse si hay suficiente memoria consecutiva libre

El operador corchete [ ]

Acceso a los elementos de un vector

Para acceder a las componentes individuales de un vector se usa el operador corchete ([ ]). Tiene dos operandos:

1. el nombre del vector, y
2. la posición de la componente a la que queremos acceder.

Operador [ ]

`vector`[*índice*]

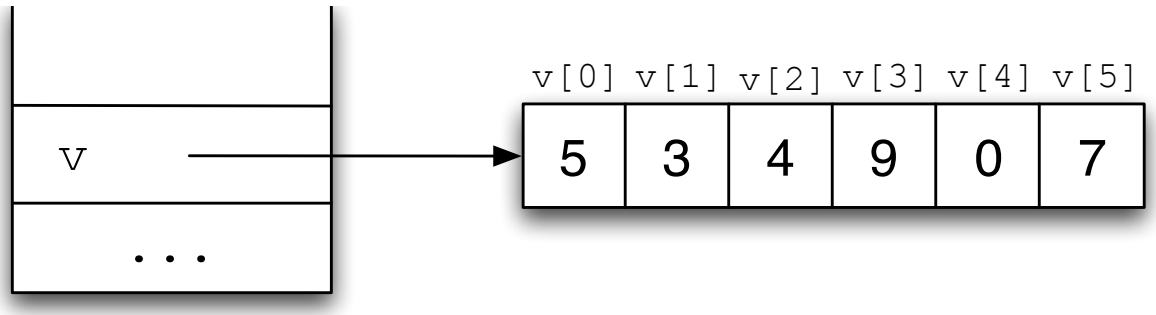
- El *índice* debe ser una expresión de tipo `int`, con un valor no negativo.
- Si el *índice* está fuera del rango del vector se produce un error en tiempo de ejecución.
- Es un **left-value**, es decir, se puede colocar a la izquierda en una asignación.

La primera componente es la CERO

Hace más eficiente el acceso a las componentes

En un vector de n elementos:

- La primera componentes tiene índice 0, y
- por tanto, la última tendrá índice $n-1$.
- Objetivo: hacer los accesos a las componentes más rápidos.



La posición de cada componente del vector se calcula usando:

$$dir.\ elemento\ i\text{-}ésimo = principio\ vector + i * n^{o}\ bytes\ tipo$$

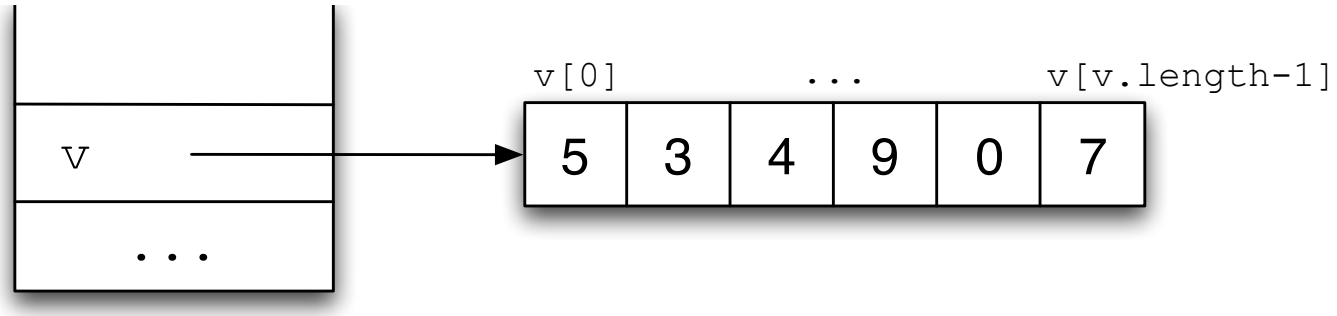
Ejemplo: v es un vector de `int` situado en la dirección $0x3000$

$$\mathsf{v}[0] \rightarrow 0x3000 + 0 * 4 = 0x3000$$
$$\mathsf{v}[1] \rightarrow 0x3000 + 1 * 4 = 0x3004$$
$$\mathsf{v}[2] \rightarrow 0x3000 + 2 * 4 = 0x3008 \quad \ldots$$

El atributo `length`

Los vectores no cambian de tamaño

- El atributo `length` guarda el tamaño del vector.
- Es una constante pública (`public` y `final`), es decir, es accesible desde fuera de la clase y no cambia de valor.
- Se le da valor cuando se crea el vector.



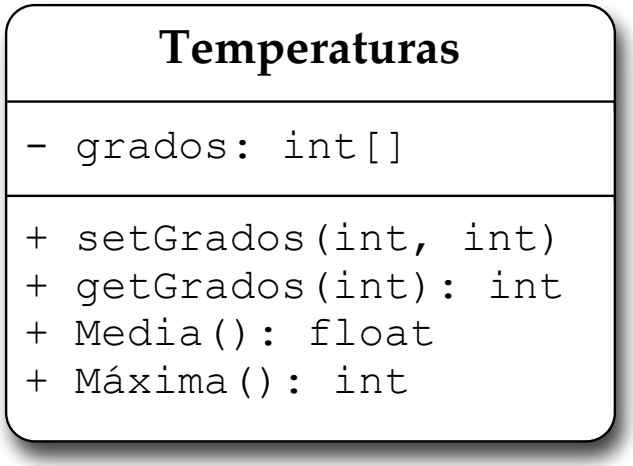
Importante

Los vectores nunca cambian de tamaño, por eso el atributo `length` es constante

Programa: Clase Temperaturas

Análisis

- Atributos:
  1. Un vector con temperaturas (enteros)
- Métodos:
  1. Calcular la temperatura `máxima()` y `media()` (enunciado).
  2. Métodos `set()` y `get()` para poder cambiar las componentes del vector.



Clase Temperaturas

Atributos y métodos set() y get()

```
4  public class Temperaturas {
5      /**Valor de la temperatura de cada hora del día*/
6      private int[ ] grados = new int[24];

8      /**Fija la temperatura de una hora del día
9       * @param h indica la hora que se va a fijar
10      * @param t indica el valor de la temperatura*/
11     public void setGrados(int h, int t) {
12         if ( h>=0 && h<=23)  grados[h]=t;
13     }

15     /**Devuelve el valor de la temperatura de una hora
16      * @param h indica la hora que se quiere obtener
17      * @return el valor de temperatura de esa hora */
18     public int getGrados(int h) {
19         return grados[h];
20     }
       ...
```

Clase Temperaturas

Métodos máxima() y media()

```
22      /**Calcula la temperatura media del día
23       * @return la temperatura media*/
24      public double media() {
25          int suma=0;
26          for (int hora=0; hora<grados.length; hora++)
27              suma+=grados[hora];
28          return (double)suma/24;
29      }

31      /**Calcula la temperatura máxima del día
32       * @return la temperatura máxima*/
33      public int máxima() {
34          int max=grados[0];
35          for (int hora=1; hora<grados.length; hora++)
36              if (grados[hora]>max)   max=grados[hora];
37          return max;
38      }
49  }
```

Ejemplo 1 - Clase Temperaturas

Programa principal

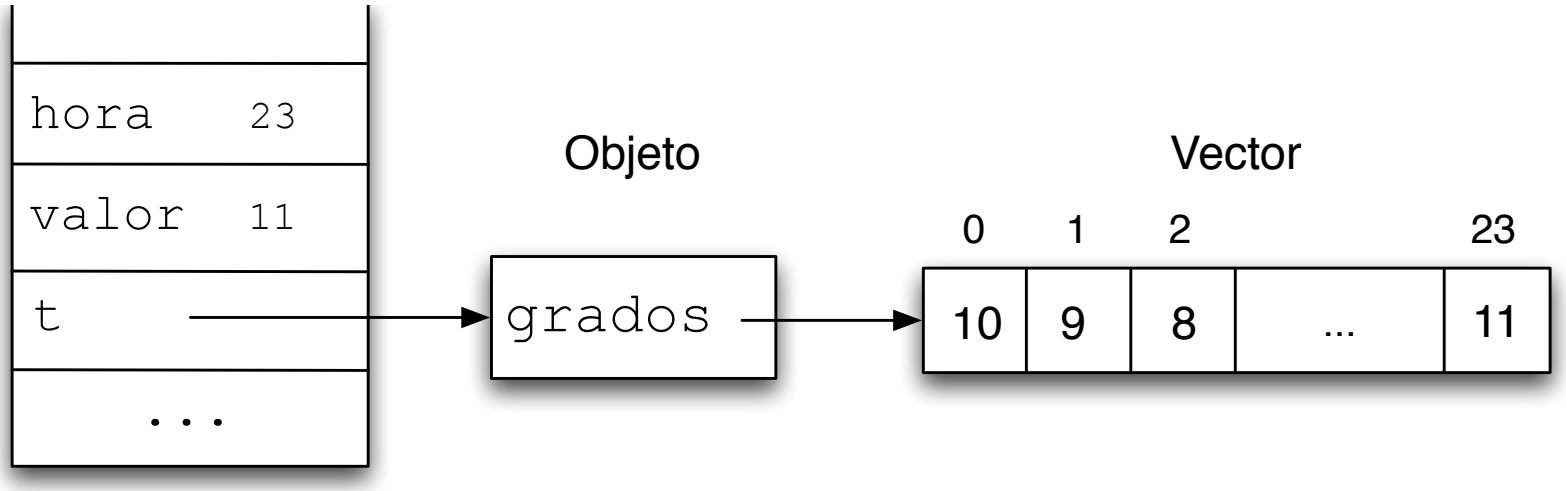
```
1  public class Ejemplo1Temperaturas {
2
3      public static void main(String[ ] args) {
4          //Objeto Scanner asociado con el teclado
5          Scanner teclado= new Scanner(System.in);
6          //Objeto de la clase Temperaturas
7          Temperaturas t = new Temperaturas();
8          //Leemos las temperaturas
9          System.out.print("Introduce las temperaturas: ");
10         int valor;
11         for (int hora=0;hora<24;hora++) {
12             valor=teclado.nextInt();
13             t.setGrados(hora, valor);
14         }
15         //Mostramos el máximo y la media en la pantalla
16         System.out.printf("La máxima ha sido %d\n",t.máxima());
17         System.out.printf("La media ha sido %f\n",t.media());
18
23     }
24 }
```

Representación de un objeto Temperaturas

Las cosas se complican

Importante

Tanto los vectores como los objetos son elementos referenciados



Características de los vectores

Acceso en tiempo constante

1. Los elementos de un vector se sitúan contiguos en memoria.
2. Las componentes deben ser todas del mismo tipo.
3. **Tiempo de acceso constante** a cualquier elemento, no importa su posición. Se tarda lo mismo en acceder a cualquier componente.
4. Uso: se emplean para guardar colecciones de datos de un mismo tipo a los que necesitemos acceder en un tiempo constante.

Importante

Con los vectores es muy importante garantizar que se accede a una posición existente

Método: pintaGráfico() Temperaturas

Enunciado

Enunciado

Añadir un método a la clase `Temperaturas` *que pinte un gráfico de barras horizontal (usando asteriscos) con la temperatura de cada hora.*

```
0  : **************
1  : ***********
2  : **********
...
23 : ****************
```

Secuencia: 0,. . . ,23, recorriendo `grados[0]`,. . . ,`grados[23]` (enumeración)

1. *primer-elemento*: `hora`=0
2. *sgte-elemento*: `hora`=`hora`+1
3. *fin-secuencia*: `hora`>=`grados.length`

Clase Temperaturas

Método pintaGráfico()

```
 4 public class Temperaturas {
       ...
40     /**Pinta un gráfico de barras con la temperatura de
           cada hora*/
41     public void pintaGráfico() {
42         for (int hora=0; hora<grados.length; hora++) {
43             System.out.printf("%d:",hora);
44             for (int i=1;i<=grados[hora];i++)
45                 System.out.print('*');
46             System.out.println();
47         }
48     }
49 }
```

Programa: Búsqueda lineal en un vector

Enunciado

Enunciado

Realizar un programa que lea de teclado primero un vector de enteros y luego un valor entero y determine si dicho valor aparece o no en el vector.

- Secuencia: `v[0]`, . . . , `v[v.length-1]` (enumeración)
- Propiedad: `número` == `v[i]`

Algoritmo 1 Búsqueda lineal en un vector

Leer número y v de teclado
i=0
mientras (i<v.length) Y (número != v[i]) **hacer**
    i=i+1
fin mientras
si (i<v.length) **entonces** Imprimir *SÍ está*
sino Imprimir *NO está*
fin si

Búsqueda lineal en un vector

Programa principal

```
 5 public class BúsquedaLineal1 {

 7     public static void main(String[ ] args) {
 8         //Objeto Scanner asociado con el teclado
 9         Scanner teclado= new Scanner(System.in);
10         //Leemos el vector de enteros de teclado
11         int[ ] v = leeVector(teclado);
12         //Leemos el número
13         System.out.print("\nIntroduce un número: ");
14         int número = teclado.nextInt();
15         //Hacemos una búsqueda asociativa, buscando v[i]==número
16         int i=0;
17         while ( ( i < v.length ) && ( v[i] != número ) )
18             i++;
19         if ( i < v.length )
20             System.out.printf("\n%d SÍ está en el vector",número);
21         else
22             System.out.printf("\n%d NO está en el vector",número);
23     }
       ...
```

Método para leer un vector de teclado

Devolviendo un vector

```
25      /** Lee un vector del objeto Scanner pasado
26       * @param t objeto Scanner del que leer los datos
27       * @return el vector de enteros leído    */
28      public static int[ ] leeVector(Scanner t) {
29          System.out.print("Introduce el tamaño del vector: ");
30          int tamaño=t.nextInt();
31          //Reservamos memoria para el vector
32          int[ ] vec = new int[tamaño];
33          //Leemos las componentes
34          System.out.print("Introduce las componentes: ");
35          for (int i=0; i<vec.length; i++)
36              vec[i]=t.nextInt();
37          return vec;
38      }
39  }
```

Búsqueda lineal (con un método)

Devolviendo un valor booleano

```
 5 public class BúsquedaLineal2 {

 7     public static void main(String[ ] args) {
           ...
15         //Comprobamos si está en el vector
16         if ( búsquedaLineal(v,número) )
17             System.out.printf("\n%d SÍ está en el vector",número);
18         else
19             System.out.printf("\n%d NO está en el vector",número);
20     }

48     public static boolean búsquedaLineal(int[ ] vec, int n) {
49         //Hacemos una búsqueda asociativa, buscando vec[i]==n
50         int i=0;
51         while ( ( i < vec.length ) && ( vec[i] != n ) )
52             i++;
53         return ( i < vec.length );
54     }
55 }
```

Problema: Temperaturas de un mes

Enunciado

Enunciado

Realizar un programa que lea todas las temperaturas horarias de los días de un mes y calcule e imprima la temperatura máxima de ese mes.

- En lugar de crear un vector de elementos de un tipo básico, vamos a utilizar un vector cuyo tipo será una clase.
- Pero hay que tener en cuenta que crear y reservar espacio para el vector (usando new) no implica que se creen sus componentes.
- Es decir:
  1. se debe crear el vector, y
  2. cada uno de los objetos que contiene.

Temperaturas de un mes

Fichero

```
1  import java.io.File;
2  import java.io.FileNotFoundException;
3  import java.util.Scanner;

5  /** Temperaturas de un mes, ejemplo de un vector de objetos
6   *  @author los profesores de IP  */
7  public class TemperaturasUnMes {

9      public static void main(String[] args) throws
           FileNotFoundException {
10         //Objeto File para acceder a un fichero
11         File f = new File ( "temperaturas.txt" );
12         //Objeto Scanner asociado con el objeto f
13         Scanner fichero= new Scanner(f);
14         //Leemos el número de días para el que tiene el fichero
15         int días = fichero.nextInt();
           ...
```

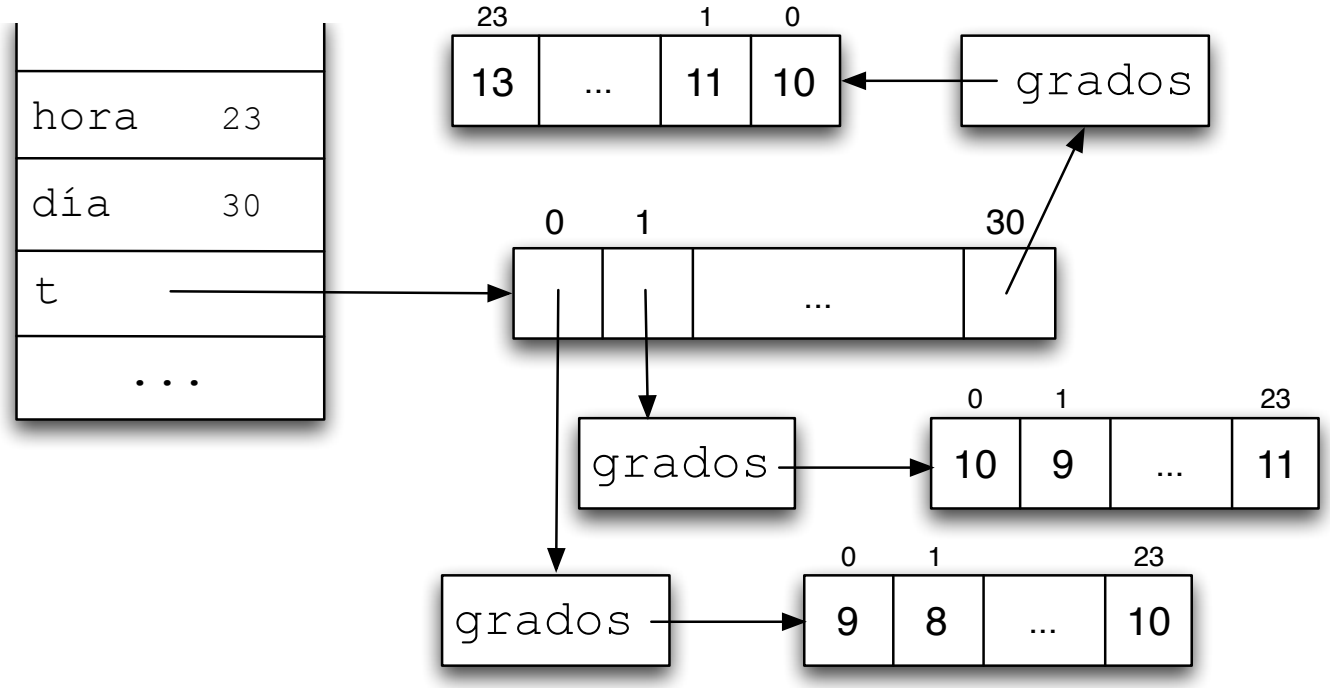
Temperaturas de un mes

Vector de objetos

```
16          //Vector con tantos objetos Temperaturas como días
17          Temperaturas[ ] t = new Temperaturas[días];
18          //Leemos las temperaturas de cada día
19          for (int día=1; día <= días; día++) {
20              t[día-1]=new Temperaturas(); //creamos el objeto!
21              for (int hora=0;hora<24;hora++)
22                  t[día-1].setGrados( hora, fichero.nextInt() );
23          }
24          //Calculamos la temperatura máxima del mes
25          int máxima=t[0].máxima(); //temp máxima primer día mes
26          //Recorremos el resto de días, hasta t[t.length-1]
27          for (int día=1; día<t.length; día++) {
28              int m = t[día].máxima(); //para no llamarlo 2 veces
29              if ( máxima < m )   máxima = m;
30          }
31          //Mostramos la temperatura máxima en la pantalla
32          System.out.printf("La temperatura máxima ha sido %d\n",
                máxima);
33      }
34  }
```

Vectores de objetos

Un vector puede contener objetos de una clase



Importante

Un vector de objetos implica que se debe hacer un new para el vector y un new por cada objeto que contenga el vector

Cadenas de caracteres

Secuencias de caracteres

Cadena de caracteres

Es una secuencia de caracteres, donde cada uno de los caracteres pertenece al alfabeto que define el esquema de codificación empleado.

- Permiten guardar la información de carácter textual.
- El ejemplo más habitual es guardar el nombre y el apellido de una persona, o la dirección y ciudad en la que vive.
- Las constantes cadena de caracteres se representan como secuencias de caracteres entre **comillas dobles**.
- Ejemplos: `"Hola, Mundo"` o `"Introduce un valor: "`

Importante

No confundir las constantes cadena, entre comillas dobles, con las constantes `char`*, entre comillas simples*

Cadenas de caracteres

Vectores de caracteres y algo más

Importante

En Java las cadenas no se suelen representar con `char[ ]`

- Con las cadenas se deben poder hacer operaciones de entrada y salida de manera estándar.
- Suelen utilizarse para representar datos, pero no se hacen operaciones con ellas.
- Requieren realizar otros tipos de operaciones, como convertir un cadena en mayúsculas o comparar si dos cadenas son iguales.
- Por todo ello, se suelen manipular con clases que incluye Java, por ejemplo:
  1. `String`: cadenas inmutables (no cambian).
  2. `StringBuffer`: cadenas mutables (sí cambian).

Clase String: declaración, uso y métodos

Hay muchos más aspectos, consulta la documentación de la clase

```
1    String s1 = "Hola, ";              //s1 creado e inic. sin new
2    String s2 = new String("Mundo"); //Ahora con new
3    s1 += s2;                          //s1 contendrá "Hola, Mundo"
4    System.out.print(s2+" Java" );     //Imprime "Mundo Java"
5    String s3 = teclado.next();        //Lee una cadena con Scanner
```

Algunos métodos importantes:

- `int length()`: devuelve la longitud.
- `char charAt(int i)`: devuelve el caracter de la posición `i`.
- `boolean equals(String s)`: devuelve cierto si el objeto contiene la misma cadena que `s`.

Importante

Los objetos `String` son inmutables, es decir, pueden devolver otros objetos `String` distintos pero no se puede cambiar ningún carácter de ellos mismos

Problema: Palíndromo

Enunciado

Enunciado

Realizar un programa que lea de teclado una cadena `c` *y diga si es un palíndromo o no.*

Palíndromo (R.A.E.): Palabra o frase que se lee igual de izquierda a derecha, que de derecha a izquierda
Ejemplos: anilina; dábale arroz a la zorra el abad.

Buscamos un carácter que no sea igual a su simétrico.

Secuencia (opción 1)

- Primer elto.: `i=0`, `j=c.length()-1`
- Sgte. elemento: `i++`, `j--`
- Fin secuencia: `i>=j`

Buscar: `c.charAt(i)!=c.charAt(j)`

Secuencia (opción 2)

- Primer elemento: `i=0`
- Sgte. elemento: `i++`
- Fin secuencia: `i>=c.length()/2`

`c.charAt(i)!=c.charAt(c.length()-1-i)`

Palíndromo

Dos métodos

```
26      public static boolean esPalíndromoV1(String c) {
27          //Dos índices. Buscamos: c.charAt(i)!=c.charAt(j)
28          int i=0;
29          int j=c.length()-1;
30          while ( (i<j) && (c.charAt(i)==c.charAt(j)) ) {
31              i++;
32              j--;
33          }
34          return ( i >= j );
35      }

        ...
40      public static boolean esPalíndromoV2(String c) {
41          //Un índice. Buscamos: c.charAt(i)!=c.charAt(c.length()-1-i)
42          int i=0;
43          while ( (i < c.length()/2) &&
44                  (c.charAt(i)==c.charAt(c.length()-1-i)) )
45              i++;
46          if ( i < c.length()/2 ) return false;
47          else return true;
48      }
```

Problema: Cadena en mayúsculas

Enunciado

Enunciado

Realizar un programa que lea de teclado una palabra en inglés y la convierta a mayúsculas.

- El programa necesita tratar todos los caracteres de una cadena y modificar los que están en minúsculas.
- Al ser los objetos `String` inmutables no podemos usar un `String` si queremos modificar la cadena.
- Necesitamos otra forma de hacerlo. Hay varias alternativas:
  1. Usar un vector de elementos `char`.
  2. Usar la clase `StringBuffer` que permite trabajar con cadenas modificables.

Cadena en mayúsculas

Programa

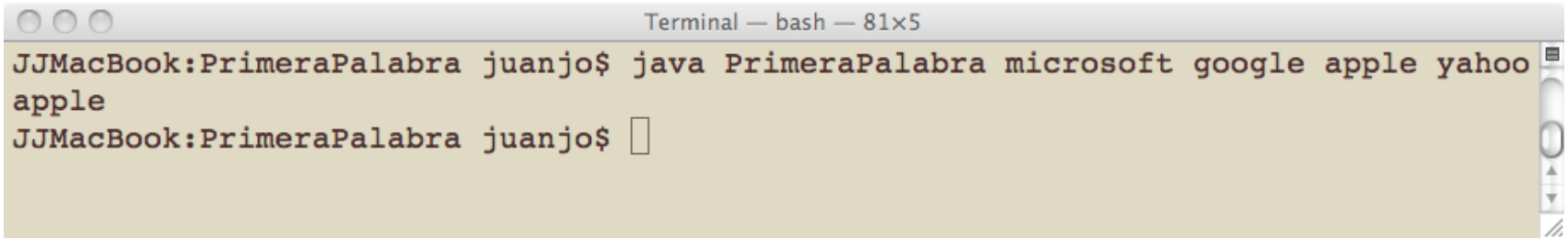
```
5  public class Mayúsculas {

7      public static void main(String[ ] args) {
8          //Objeto Scanner asociado con el teclado
9          Scanner teclado= new Scanner(System.in);
10         System.out.print("Cadena: ");
11         //Leemos la cadena de caracteres
12         String cadena = teclado.next();
13         char[ ] mayúsculas = cadena.toCharArray();
14         //La pasamos a mayúsculas
15         for (int i=0; i<mayúsculas.length; i++) {
16             if (mayúsculas[i] >= 'a' && mayúsculas[i] <= 'z')
17                 //hay que restarle 32: 'A' es 65 y 'a' es 97
18                 mayúsculas[i] = (char) (mayúsculas[i] - 32);
19         }
20         //Se imprime la cadena
21         System.out.println(mayúsculas);
22     }
23 }
```

El parámetro args

Contiene todo los argumentos con los que se ejecuta un programa.

Enunciado

Hacer un programa que imprima por pantalla el parámetro que vaya antes alfabéticamente



El programa recibirá las 4 cadenas en un vector de argumentos:

Programa: Primera Palabra

Muestra en pantalla el parámetro que va antes alfabéticamente

```
1  /** Recibe varias cadenas como parámetros e imprime la primera
       alfabéticamente
2   *  @author los profesores de IP  */
3  public class PrimeraPalabra {
4      public static void main(String[ ] args) {
5          if ( args.length > 0 ) {
6              int primera=0;
7              for (int i=1; i<args.length; i++)
8                  if (args[primera].compareTo(args[i]) > 0)
9                      primera = i;
10             System.out.println(args[primera]);
11         }
12     }
13 }
```

Matrices

Tablas bidimensionales de componentes

Matriz

Es una tabla rectangular o cuadrada de elementos ordenados en filas y columnas. Al elemento que se encuentra en la fila i-ésima y la columna j-ésima se le llama elemento (i,j)-ésimo de la matriz.

			j			
	1	4	7	0	5	2
	5	6	4	1	8	7
i	4	3	7	9	0	2
	6	1	3	5	6	8

Importante

Como en los vectores, la numeración de las filas y las columnas de una matriz empieza en cero

Definición, creación, inicialización de una matriz

Igual que los vectores, pero con una dimensión más

Declaración y creación de una matriz

`tipo[ ][ ] nombre` [= `new tipo` [*filas*][*columnas*]];

Ejemplos:

```
1    int[ ][ ] m = new int [4][6];      //Matriz 4 filas y 6 col.
2    double[ ][ ] t;                    //Declaración de la matriz
3    t = new double [31][24];           //Creación: 31x24 reales
4    Fecha p[ ][ ] = new Fecha [3][2]; //Matriz 3x2 objetos Fecha
```

Importante

Al crear la matriz hay que indicar ambas dimensiones, el número de filas y de columnas

Inicialización de una matriz

Otra forma de declarar, reservar espacio e inicializar una matriz

Declaración, creación e inicialización de una matriz

`tipo[ ][ ] nombre` = { { *lista valores fila 0* } ,
{ *lista valores fila 1* } ,
...
{ *lista valores fila n* } };

Ejemplo:

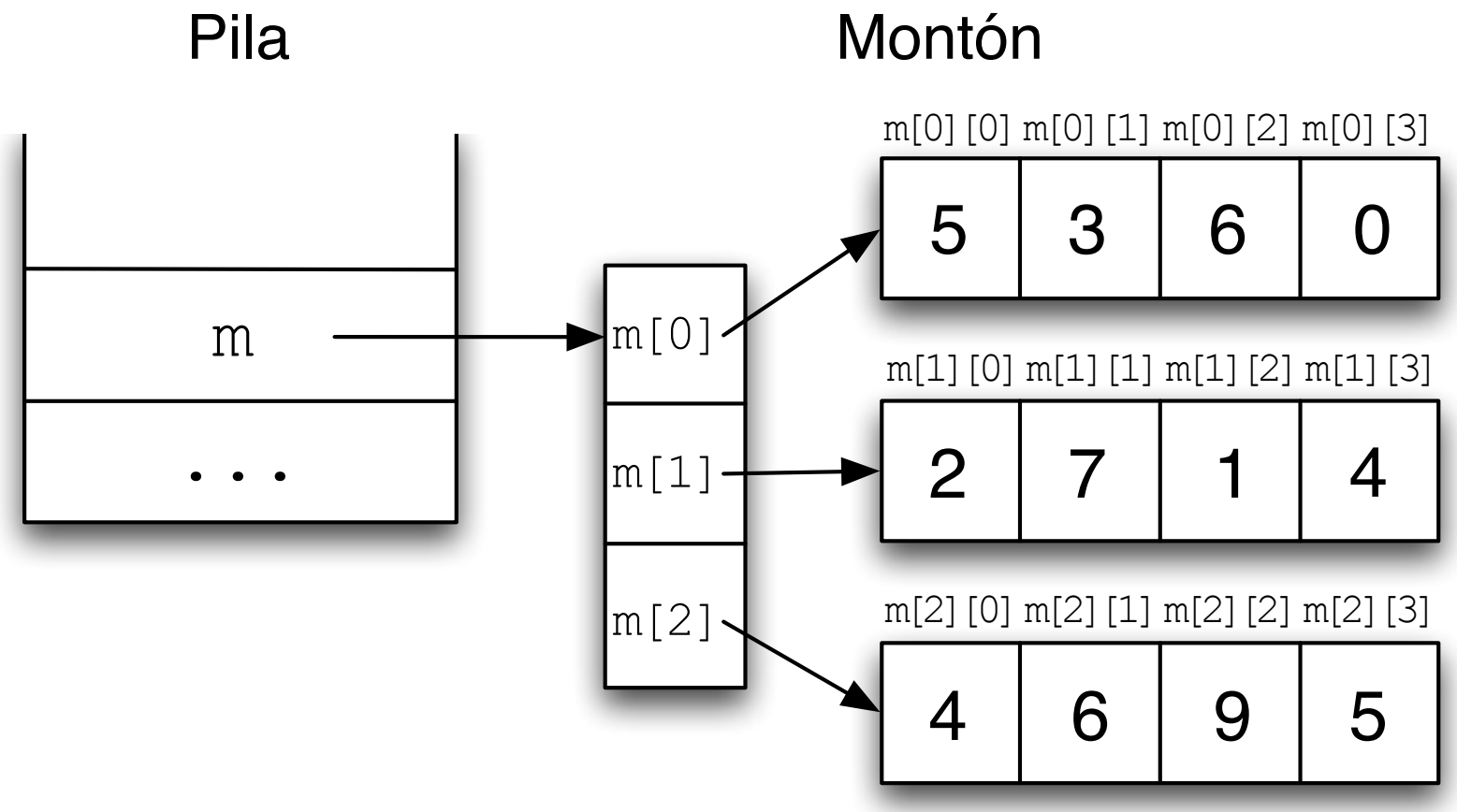
```
1     //Matriz de 4 filas y 6 columnas, inicializada
2     int[ ][ ] m = {  { 1, 4, 7, 0, 5, 2 },
3                      { 5, 6, 4, 1, 8, 7 },
4                      { 4, 3, 7, 9, 0, 2 },
5                      { 6, 1, 3, 5, 6, 8 }  };
```

Importante

Los valores de los elementos de cada fila van separados por comas

Representación en memoria de una matriz

Es un vector de vectores



Importante

Una matriz es un vector de filas, donde cada fila es a su vez un vector

Uso: el operador [ ] y `length`

Acceso a las componentes y tamaño

Operador [ ] con matrices

componente: `matriz[`*fila*`][`*columna*`]`
vector fila: `matriz[`*fila*`]`

Hay que recordar que:

- Los índices de la fila y la columna deben ser de tipo `int`,
- se empieza a contar en 0, y
- hay que evitar salirse de la matriz controlando sus dimensiones.

Para conocer las dimensiones:

- Nº de filas: `matriz.length`
- Nº de columnas: `matriz[`*fila*`].length`

Programa: Suma de dos matrices

Enunciado

Enunciado

Dadas dos matrices de enteros no vacías de la misma dimensión, calcular otra matriz que sea la suma de ambas e imprimirla en pantalla.

1	4	7	0
5	6	2	1
9	3	4	8

+

3	2	9	1
8	4	5	7
2	1	3	6

=

4	6	16	1
13	10	7	8
11	4	7	14

Algoritmo 2 Calcular la suma de dos matrices (alto nivel)

Leer `matriz1` y `matriz2` de teclado
Crear la matriz `suma`
Calcular las componentes de la matriz `suma` (bucles anidados)
Imprimir la matriz `suma`

Suma de matrices

Programa principal

```
 5 public class SumaMatrices {

 7     public static void main(String[ ] args) {
 8         //Objeto Scanner asociado con el teclado
 9         Scanner teclado= new Scanner(System.in);
10         //Leemos dos matrices de enteros de teclado
11         int[ ][ ] matriz1 = leeMatriz(teclado);
12         int[ ][ ] matriz2 = leeMatriz(teclado);
13         //Creamos la matriz suma
14         int[ ][ ] suma = new int[matriz1.length][matriz1[0].length];
15         //Calculamos la suma
16         for (int i=0; i<suma.length; i++)
17             for (int j=0; j<suma[i].length; j++)
18                 suma[i][j]=matriz1[i][j]+matriz2[i][j];
19         //Imprimimos la matriz con la suma
20         imprimeMatriz(suma);
21     }
       ...
```

Leer e imprimir una matriz

Recorridos con bucles anidados

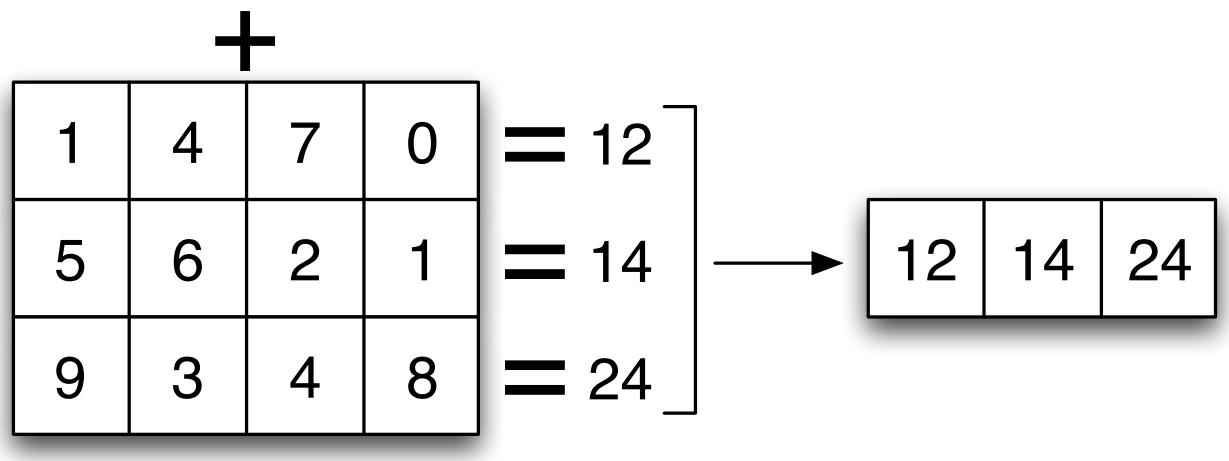
```
26      public static int[ ][ ] leeMatriz(Scanner t) {
27          System.out.print("Nº de filas y columnas: ");
28          int filas=t.nextInt();
29          int columnas=t.nextInt();
30          //Reservamos memoria para la matriz
31          int[ ][ ] m = new int[filas][columnas];
32          //Leemos las componentes, recorriendo la matriz
33          System.out.print("Componentes: ");
34          for (int i=0; i<m.length; i++)
35              for (int j=0; j<m[i].length; j++)
36                  m[i][j]=t.nextInt();
37          return m;
38      }

42      public static void imprimeMatriz(int[ ][ ] m) {
43          //Imprimimos las componentes
44          for (int i=0; i<m.length; i++) {
45              for (int j=0; j<m[i].length; j++)
46                  System.out.printf("%d ",m[i][j]);
47              System.out.println();
48          }
49      }
50  }
```

Programa: Suma de cada fila de una matriz

Enunciado

Enunciado

Dada una matriz no vacía de números reales leída de teclado, calcular e imprimir en pantalla un vector donde cada componente represente la suma de los elementos de la fila correspondiente en la matriz.



Lo interesante del programa es que nos permite tratar individualmente cada fila de una matriz.

Suma de cada fila de una matriz

Programa principal

```
 5 public class SumaFilas {

 7     public static void main(String[ ] args) {
 8         //Objeto Scanner asociado con el teclado
 9         Scanner teclado= new Scanner(System.in);
10         //Leemos la matriz de reales de teclado
11         float[ ][ ] matriz = leeMatriz(teclado);
12         float[ ] sumas= sumaPorFilas(matriz);
13         //Imprimimos la suma de cada fila
14         for (int i=0; i<sumas.length; i++)
15             System.out.printf("\nFila %d: suma %f",i,
                   sumas[i]);
16     }
       ...
```

Suma de cada fila de una matriz

Tratando cada fila como un vector

```
38      public static float[ ] sumaPorFilas(float[ ][ ] m) {
39          //Calculamos su suma, lo hacemos sobre un vector
40          float[ ] s = new float[m.length];
41          for (int i=0; i<m.length; i++)
42              s[i]=sumaVector(m[i]);
43          return s;
44      }
```

```
38      public static float sumaVector(float[ ] v) {
39          float suma = 0;
40          for (int i=0; i<v.length; i++)
41              suma+=v[i];
42          return suma;
43      }
44  }
```

Vectores multidimensionales

Se pueden crear vectores de más de dos dimensiones

```
1    int[ ][ ][ ] m = new int [3][4][6];
```