Estructuras de Datos:
Strings y Arrays

Segunda Parte

¿Qué es un array?

arrays unidimensionales o vectores

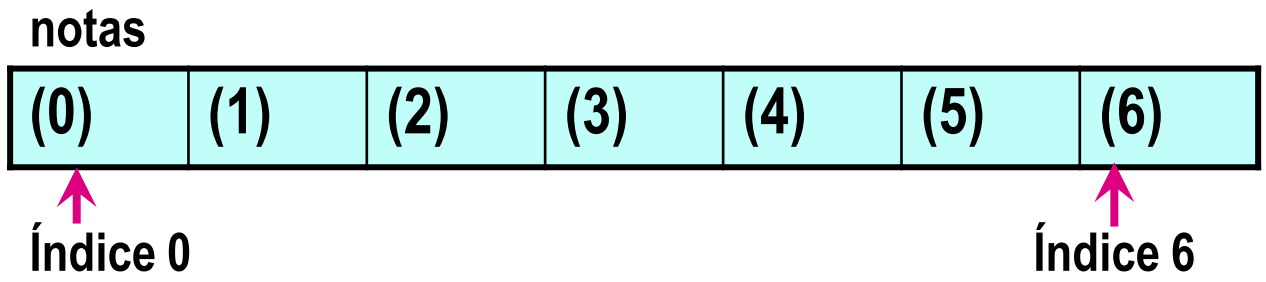
arrays multidimensionales o matrices

ARRAYS

¿Qué es un array?

- **Un array es una serie de elementos de datos**
  - Todos los elementos del array tienen **el mismo tipo** de datos
  - Se accede a los elementos individuales utilizando índices enteros
  - La primera posición del array es la 0
- **La longitud del array se establece al crearlo**
  - El número de elementos no puede ser modificado en tiempo de ejecución. Los arrays son estáticos. No podremos
    - Ni eliminar posiciones
    - Ni insertar posiciones
- **Al crear un array, si no se especifican valores, todas sus posiciones son inicializadas al valor por defecto del tipo que se trate**
- **Un array en Java es una clase especial (definida en java.util.Arrays)**
- **Un array puede contener tanto tipos primitivos como tipos complejos**
- **Si intentamos acceder a una posición fuera del array se produce un error (y se lanza una excepción. Próximo tema)**

¿Qué es un array? Ejemplo



- Para declarar y crear una matriz entera con siete elementos:

```
int[] notas =new int[7];
```

- Para acceder al tercer elemento de la matriz:

```
int total = notas[2]
```

Vector

ARRAY UNIDIMENSIONAL

Cómo declarar y crear un array unidimensional

Recuerda, como en cualquier objeto distinguimos entre

- **Declarar una variable array**
  - Creamos una variable capaz de direccionar (*o apuntar, o contener la dirección de memoria de*) un objeto array
  - El objeto aun no está creado y la variable apunta a (*o contiene*) null
- **Crear o instanciar un array**
  - Creamos físicamente el objeto. Se le asigna posiciones de memoria
  - Empleamos la palabra reservada new y se invoca al constructor
- **Podremos declarar e instanciar en la misma instrucción**

Cómo declarar y crear un array unidimensional

- **Declarar una variable array:**

```
tipo_de_dato[] nombre_del_array;
```

  - Sintaxis alternativa (para programadores de C)

    tipo_de_dato nombre_del_array[ ];

- **Crear un array y asignarlo a la variable:**

```
nombre_del_array = new tipo[num_elementos];
```

- **Declarar+Crear un array:**

```
tipo_de_dato[] nombre_del_array = new tipo[num];
```

- **Declarar+Crear+Inicializar un array:**

```
tipo_de_dato[] nombre_del_array ={v1, v2,..,vn};
```

Declarar y crear un array unidimensional. Ejemplo

- **Declarar una variable array:**

```
int[] numeros;
```

  - Sintaxis alternativa (para programadores de C)

    int numeros[ ];

- **Crear un array y asignarlo a la variable:**

```
numeros=new int[4];
```

- **Declarar+Crear un array:**

```
int[] numeros =new int[4];
```

- **Declarar+Crear+Inicializar un array:**

```
int[] numeros={2, -4, 15, -25};
```

Acceso a los elementos de un array unidimensional

- Ejemplo: int[ ] numeros = {2, -4, 15, -25};

- **Si un "array" unidimensional a tiene “n” miembros**
  - al primero se accede escribiendo numeros[0]
  - al último se accede escribiendo numeros[n-1]
- **Acceso para lectura (extraer el valor):**
  - System.out.println(numeros[3]);
- **Acceso para escritura (cargar un valor)**
  - numeros[2] = 99;

Recorrido de los elementos de un array unidimen.

- Es muy frecuente recorrer los elementos de un array
- Se puede utilizar un bucle con contador

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

for (int i = 0; i < datos.length; i++) {
   System.out.print(datos[i] + " "
}
```

- o iterar sobre los elementos

```
for (int dato: datos) {
   System.out.print(dato + " ");
}
```

Nueva Sentencia for/in

Itera para todos los elementos de cualquier tipo de colección arrays, listas.,,,

```
for (inicializacion: coleccion) {
    sentencias;
}
```

- Nota: el atributo público *length* nos proporciona el número de elementos (o longitud del array)

Recorrido de los elementos de un array unidimen.

- **Otro ejemplo:**

```
int maximo = Integer.MIN_VALUE;
for (int i = 0; i < vector.length; i++) {
    if (vector[i] > maximo)
    maximo = vector[i];
}
```

```
int maximo = Integer.MIN_VALUE;
for (int n: vector) {
    if (n > maximo)
    maximo = n;
}
```

Tamaño de los arrays. Arrays Vs. Listas

- **Los arrays son de tamaño fijo, mientras que las listas son de tamaño variable**
- **Si no sabemos el tamaño de un array al crearlo, tenemos 2 opciones:**
  1. Crearlo muy grande, de forma que nos quedan los datos en el peor caso posible

     El precio que pagamos es desperdiciar espacio
  2. Crearlo de un tamaño reducido, pero prever que si llegan más datos habrá que ampliarlo (o sea, crea un array mayor y copiar los datos)

     El precio que pagamos es tiempo de ejecución.
- **Las listas (clase List) son de tamaño variable. Las listas son una forma cómoda de aplicar la segunda opción. Lo vemos próximamente…**

Ejemplo

```
public class Matriz1 {
 public static void main(String[] args) {
 float[] notas = {5.8f,6.2f,7.1f,5.9f,3.6f,9.9f,1.2f,10.0f,4.6f,5.0f};
 String[] nombres = new String[10];
 nombres[0]="Pedro";nombres[1]="Ana";nombres[2]="Luis";
 nombres[3]="Luis";nombres[4]="Juan";Nombres[5]="Eva";
 nombres[6]="Mari";Nombres[7]="Fran";nombres[8]="Luz";
 nombres[9]="Sol";

 for (int i=0;i<nombres.length;i++)
    System.out.println(nombres[i] + " : " + notas[i]);
  System.out.println();

 int Aprobados = 0;
 String nombresAprobados = new String();
 for (int i=0;i<nombres.length;i++)
   if (notas[i]>=5.0){
      aprobados++;
      nombresAprobados = nombresAprobados+" "+nombres[i];
      }
 System.out.println("Aprobados: " + aprobados);
 System.out.println("Aprobados:" + nombresAprobados);
 }
}
```

Arrays y Métodos

- Podemos pasar un array como parámetro a un método, teniendo en cuenta **que los cambios que realicemos sobre el array en el procedimiento llamado se mantendrán** al volver el flujo de la ejecución al procedimiento llamador
- Ello es debido a que **los arrays son tipos por referencia**, y por lo tanto, las variables del array que manejamos tanto desde el procedimiento llamador, como desde el procedimiento llamado, son en realidad punteros hacia una misma zona de memoria o referencia, la que contiene el array

Arrays y Métodos. Ejemplo

Asi pues…

- Cuando se llama a un método y se le pasa un array, el método hace su copia de la referencia; pero comparte el array

```
void caso1(int[] x) {
  x[0] *= 10;
}
void test1() {

  int[] a = {1, 2, 3};
  System.out.println(Arrays.toString(a));
  caso1(a);
  System.out.println(Arrays.toString(a));
}
```

Ejecución

[1, 2, 3]
[10, 2, 3]

Ejemplo

```
public class Matriz2 {

 public static void imprimir(String[] nom) {
    for (int i=0;i<nom.length;i++)
      System.out.println(nom[i]);
  }

  public static byte entreNotas(float nota1, float nota2, float[] not) {
    byte contador = 0;
    for (int i=0;i<not.length;i++){
      if ((not[i]>=nota1)&&(not[i]<=nota2))
       contador++;
    }
    return contador;
  }

  public static void main(String[] args) {
    float[] notas = {5.8f,6.2f,7.1f,5.9f,3.6f,9.9f,1.2f,10.0f,4.6f,5.0f};
    String[] nombres = new String[10];
    nombres[0]="Pedro";nombres[1]="Ana";nombres[2]="Luis";…
    imprimir(nombres);
    System.out.println("Aprobados: ");
    System.out.println(entreNotas(5.0f,10.0f,notas));
    System.out.println("Suspensos: ");
    System.out.println(entreNotas(0.0f,4.9f,notas));
    System.out.println("Matriculas: ");
    System.out.println(entreNotas(10.0f,10.0f,notas));
  }
}
```

Copia de arrays

- **Cuando una variable de tipo array se hace igual a otro, se copia la referencia; pero se comparte el array:**

```
void copia1() {

int[] a = {1, 2, 3};
System.out.println(Arrays.toString(a));
int[] b = a;
System.out.println(Arrays.toString(b));
a[0] *= 10;
System.out.println(Arrays.toString(b));

}
```

Ejecución

[1, 2, 3]
[1, 2, 3]
[10, 2, 3]

- **Si no basta con compartir la referencia, sino que se necesita otra copia de un array, se puede usar el método clone( )**
  - Si los elementos del array son de un tipo primitivo, se copia su valor
  - Si son objetos, se copia la referencia, compartiéndose el objeto

```
void copia2() {

int[] a = {1, 2, 3};
System.out.println(Arrays.toString(a));
int[] b = a.clone();
System.out.println(Arrays.toString(b));
a[0] *= 10;
System.out.println(Arrays.toString(b));

}
```

Ejecución

[1, 2, 3]
[1, 2, 3]
[1, 2, 3]

Copia de arrays

- **Con el método clone()**
  - Si son objetos se copia la referencia, compartiéndose el objeto

```
void copia2Objetos() {

Punto[] a =
   {new Punto(1, 2), new Punto(3, 4)};
System.out.println(Arrays.toString(a));

Punto[] b = a.clone();
System.out.println(Arrays.toString(b));

a[0].multiplica(-1);
System.out.println(Arrays.toString(b));
}
```

Ejecución

```
[(1,2), (3,4)]
[(1,2), (3,4)]
[(-1,-2), (3,4)]
```

- **Por último, la copia se puede programar explícitamente**

```
tipo[] a = ...;

tipo[] b = new tipo[a.length];

for (int i = 0; i < a.length; i++)
     b[i] = a[i];
```

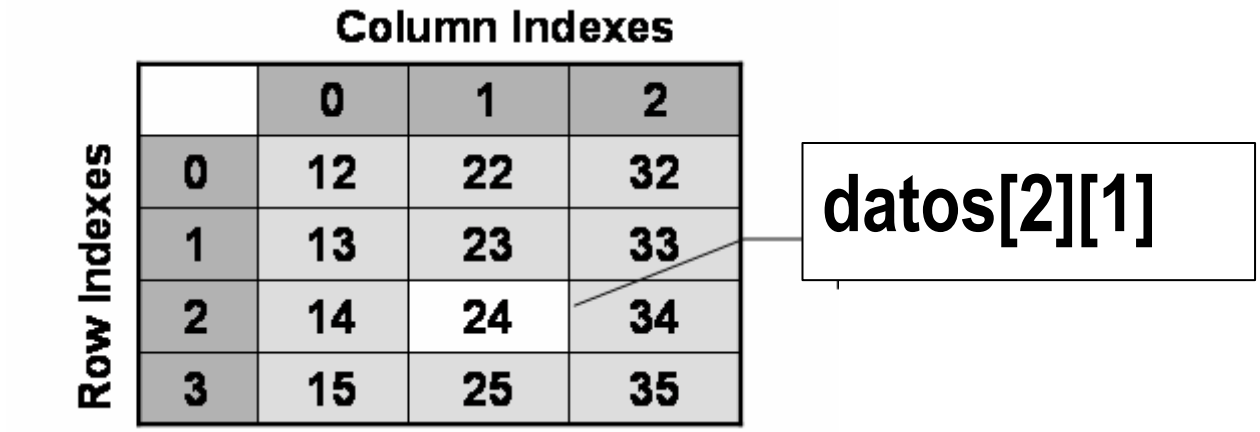
Algunos métodos de Arrays

- Propiedad pública **length**
  - Devuelve el número de elementos que contiene
- **sort( )**
  - Ordena un array
- **binarySearch( )**
  - Busca un valor en un array.
  - El array debe estar ordenado
- **equals ( )**
  - Compara arrays
- **fill( )**
  - Asigna valores a un array
- **toString ( )**
  - convierte un array en string
- **clone( ) , copyOf( ) y arrayCopy( )**
  - Copia de un array

ARRAYS MULTIDIMENSIONALES

Array bidimensional o matriz. Ejemplo

- **Podemos pensar en una matriz de dos dimensiones como si fuese una cuadrícula**



- Declarar una matriz de dos dimensiones con 4 filas y 3 columnas

  Int [ ] [ ] datos = new int [4] [3] ;

- Asignar un valor a un elemento específico de la matriz

  datos [2] [1] = 24;

Cómo declarar y crear un array bidimensional

- **Declarar una variable array:**

```
tipo[][] nombre_del_array;
```

  - Sintaxis alternativa (para programadores de C)

    tipo_de_dato nombre_del_array [ ] [ ];

- **Crear un array y asignarlo a la variable:**

```
nombre_del_array = new tipo[tamaño1][tamaño2];
```

- **Declarar+Crear un array:**

```
tipo[][] nombre = new tipo[tam1][tam2];
```

- **Declarar+Crear+Inicializar un array: (ejemplo)**

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

Array bidimensional o matriz

- **En realidad, un array bidimensional es un array de arrays**
- **Por ejemplo,**
  - Un array 4 x 3 es un array de 4 elementos, en el que cada uno de ellos es un array de 3 elementos:
    - datos[0] es un array de 3 elementos
    - …
    - datos[3] es un array de 3 elementos
- **Podemos tener arrays bidimensionales no cuadradas (cada fila tiene un número diferente de columnas)**

```
int [ ][ ] numeros = new int [4][  ];
numeros[0]=new int [7];
…
numeros[3]=new int[3];
```

Array bidimensional o matriz

- **length**

  Proporciona el número de elementos o longitud del array:

  - matriz.length // el número de filas
  - matriz[0].length // el número de columnas de la primera fila
  - matriz[1].length // el número de columnas de la segunda fila

Recorrido de los elementos de un array bidimen.

- Se puede utilizar un bucle con contador

```
double[][] matriz={{1,2,3,4},{5,6},{7,8,9,10,11,12},{13}};

for (int i=0; i < matriz.length; i++) {
    for (int j=0; j < matriz[i].length; j++) {
        System.out.print(matriz[i][j]+"\t");
        }
    System.out.println("");
}
```

- o iterar sobre los elementos

```
for (double[] fila : matriz) {
    for (double dato : fila)
        System.out.print(dato + " ");
    System.out.println();
}
```

Arrays multidimensionales. Ejemplo

```
public class MatrizUnidadApp {
    public static void main (String[] args) {
        double[][] mUnidad= new double[4][4];
        for (int i=0; i < mUnidad.length; i++) {
          for (int j=0; j < mUnidad[i].length; j++) {
               if (i == j) {
                    mUnidad[i][j]=1.0;
               }
               else {
                    mUnidad[i][j] = 0.0;
               }
          }
        }

        for (int i=0; i < mUnidad.length; i++) {
          for (int j=0; j < mUnidad[i].length; j++) {
               System.out.print(mUnidad[i][j]+"\t");
          }
          System.out.println("");
         }

   }
}
```

Arrays multidimensionales

- **Para crear una matriz multidimensional :**

```
tipo [][][]… variable =
 new tipo [elementosDim1][elementosDim2][elementosDim3]…
```

- **Total elementos = producto de todos los tamaños**