Programación en Java

Tema 2. El lenguaje de programación Java. Arrays y cadenas

Luis Rodríguez Baena

Universidad Pontificia de Salamanca (campus Madrid)
Facultad de Informática

Arrays y colecciones

- Hay dos formas de guardar varias referencias a objetos:
  - Arrays
    - ✓ Más eficiente.
    - ✓ Tamaño limitado.
    - ✓ Sólo puede almacenar referencias de un tipo.
  - Colecciones.
    - ✓ Menos eficiente.
    - ✓ Tamaño variable.
    - ✓ Almacena objetos sin tipo específico.
    - ✓ Clases genéricas `List`, `Set` y `Map`.

Declaración de arrays

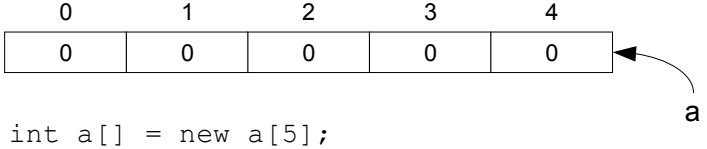
- No son un tipo de dato, sino una clase que extiende la clase `Object`.
- Declaración.
  - *`modificadores tipo[] identificador;`*
  - Los modificadores se aplican al array, no a los elementos.
- Asignar espacio de almacenamiento creando una instancia mediante `new`.

```
int[] a = new int[5];
int b[];
b = new int[3];
```

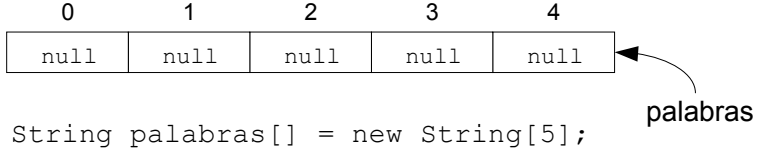
  - `length`, indica el número de elementos.
  - Los índices van de 0 hasta `length - 1`.
    - ✓ Cualquier otro valor lanza la excepción `ArrayIndexOutOfBoundException`.

Inicialización de arrays (I)

- Inicialización a los valores por omisión.
  - Un array de tipos primitivos inicializa los elementos a sus valores por omisión.
  - Un array de referencias inicializa los elementos a referencias nulas (`null`).
    - ✓ Un array de referencias no guarda el contenido del objeto, sino una referencia a la instancia.




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



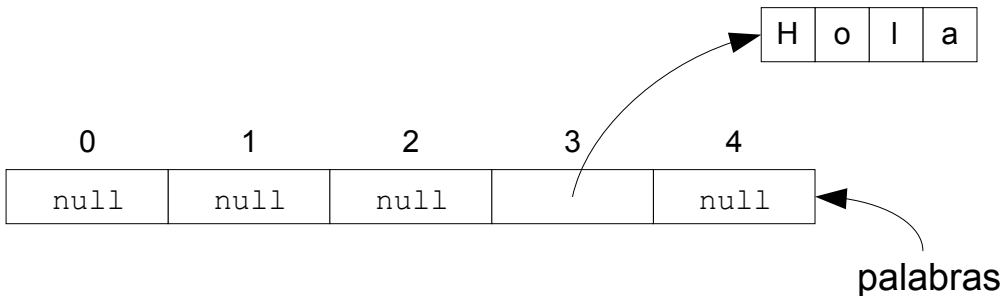

```
String palabras[] = new String[5];
```

Inicialización de arrays (II)

- Llenar el array.
  - Mediante asignaciones:

```
int[] a = new int[5];
//Carga de elementos en el array
//con números aleatorios entre 0 y 19.
  for (int i=0;i<=a.length-1;i++)
          a[i] = (int) (Math.random() * 20);
```

  - En arrays de referencia.




```
palabras[3] = "Hola";
```

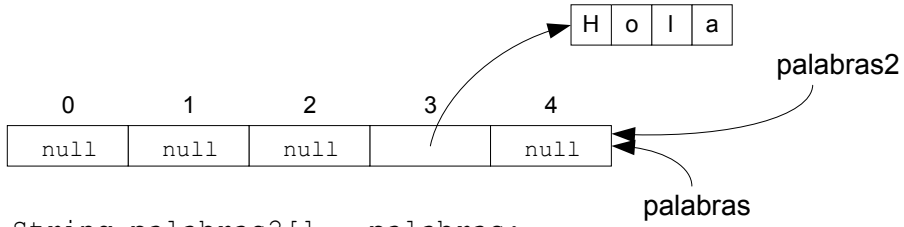
Inicialización de arrays (III)

- Inicialización de arrays en la declaración.
  - Sólo se puede hacer en la declaración.
  - No se debe poner el tamaño del array.
    - ✓ Se crea espacio de almacenamiento y se rellena en la misma declaración

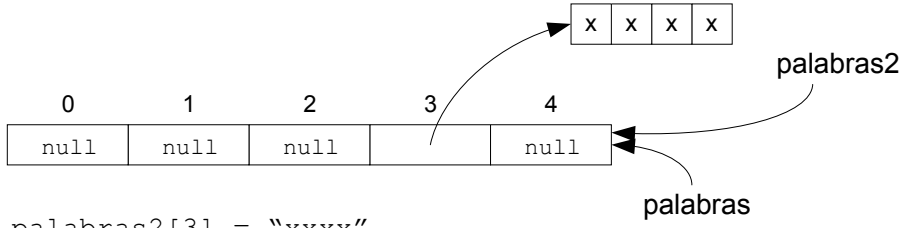
```
char b[] = {'a','b','c'};.
Integer números[] = { new Integer(1),
                      new Integer(2),
                      new Integer(3)};
```

Asignación de arrays

- Cuando se asigna un array a otro array se copia la referencia.
  - Copiar un array supone asignar los elementos a otro array.




```
String palabras2[] = palabras;
```




```
palabras2[3] = "xxxx"
```

- La clase `System` contiene un método para copiar arrays.

```
public static void arraycopy(Object origen, int posOrig,
                             Object dest, int posDest,
                             int length)
```

Arrays y métodos

- Arrays y métodos.
  - Los métodos también pueden devolver arrays.

```
static int[] cargaArray(int n){
  int[] a = new int[5];
   for (int i=0;i<=a.length-1;i++)
      a[i] = (int) (Math.random() * 20);
  return a;
}
```

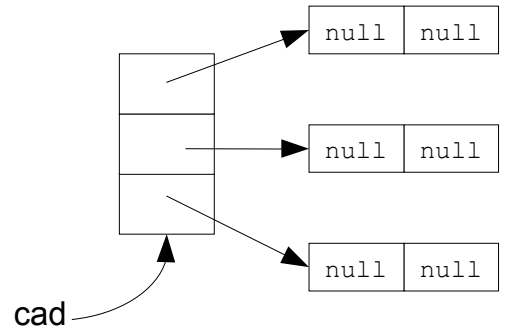
  - O recibir arrays como argumentos
    - ✓ En este caso, el array se pasa por referencia (como todos los objetos).

```
static void imprimirArray(Object[] v){
    //Muestra los elementos del array
   for (int i=0;i<=v.length-1;i++)
     System.out.println("Elemento " + i + ": " + v[i]);
}
```

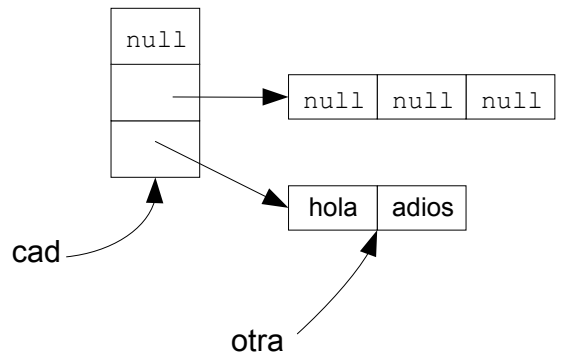
Arrays multidimensionales (I)

- Declaración.

```
String [][] cad = new String[3][2];
```




```
String cad[][] = new String[3][];
cad[1] = new String[3];
String otra[] = {"hola","adios"};
cad[2] = otra;
```



Arrays multidimensionales (II)

- El atributo `length` en arrays multidimensionales.
  - `a.length`, devolvería el tamaño de la primera dimensión.
  - `a[i].length`, devolvería el tamaño de la fila `i`.

```
for(int i=0;i <= cad.length-1;i++){
     System.out.println("Línea " + i );
     if(cad[i] == null)
        System.out.println(cad[i]);
     else
        for(int j=0;j<=cad[i].length-1;j++)
            System.out.println(cad[i][j]+" ");
};
```

El array args

- Contiene los argumentos pasados mediante la línea de órdenes.

```
class HolaMundo {
    public static void main(String args[]) {
        // Muestra "¡Hola mundo, hoy es dd-mm-aa!"
        int conta = args.length;
        System.out.print("!Hola ");
        if(conta>0){
           for(int i=0;i<=conta-1;i++)
              System.out.print(args[i] + ", ");
           System.out.println();
        }
        else
           System.out.println("anónimo,");
    }
}
```

La clase Arrays (I)

- La bibilioteca `java.utils` contiene la clase `Arrays` con funciones de utilidad para los arrays.
  - Sería necesario incorporar `import java.utils.*;` al comienzo del archivo fuente.
- Comparar arrays.

```
public static boolean equals(tipo[] a, tipo[] a2)
```

```
int v1[] = {1,2,3};
int v2[] = {1,2,3};
System.out.println(Arrays.equals(v1,v2));
```

La clase Arrays (II)

- Rellenar arrays.
  - Rellenar el array a con el valor del segundo argumento.

    `public static void fill(`*`tipo`*`[] a, `*`tipo`*` val)`
  - Rellenar el array a entre dos posiciones dadas con el valor del segundo argumento.

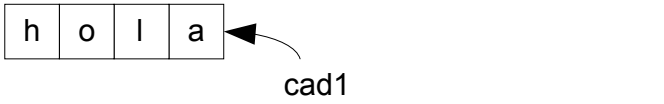
    `public static void fill(`*`tipo`*`[] a, int desde, int hasta,`
    `tipo` `val)`
- Ordenar arrays.

  `public static void sort(`*`tipo`*`[] a)`
- Buscar en arrays

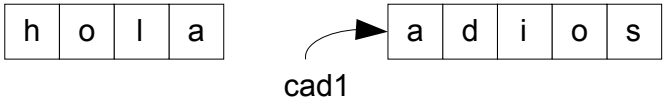
  `public static int binarySearch( `*`tipo`*`[] a, `*`tipo`*` elemento)`
  - Buscar el elemento en el array. Si lo encuentra devuelve su posición, si no devuelve un número negativo con la posición donde debería estar.

Cadenas

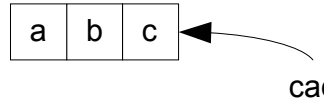
- Dos clases para almacenar cadenas.
  - `String`. Cadenas inmodificables.
    - ✓ Consume menos recursos.
  - `StringBuffer`. Cadenas modificables.



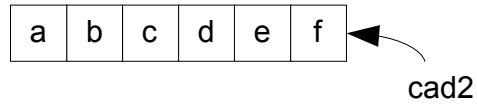

```
String cad1 = "hola";
```




```
cad1 = "adios"
```




```
StringBuffer cad2 = "abc";
```




```
cad2.append("def");
```

Constructores de String y StringBuffer

- Se puede construir un objeto de la clase `String` como si se tratara de un tipo de dato primitivo.

```
String cad1 = "hola";
```

- Constructores de `String`:
  - `String()`, crea una nueva cadena vacía.
  - `String(String valor)`. Crea una nueva cadena a con el contenido de valor.
  - `String(StringBuffer sb)`. Crea una nueva cadena a partir de un `StringBuffer`.
- Constructores de `StringBuffer`:
  - `StringBuffer()`, crea una nueva cadena vacía con una capacidad inicial de 16 caracteres.
  - `StringBuffer(int longitud)`. Crea una nueva cadena vacía con una capacidad inicial de longitud caracteres.
  - `StringBuffer(String str)`. Crea un nuevo `StringBuffer` a partir de una valor de tipo `String`.

Comparación de cadenas (I)

- El operador de igualdad sólo compara si las referencias son iguales.

```
String c1 = "hola";
String c2;
String c3 = new String(c1);
c2 = c1;
System.out.println(c1 == c2); //Devuelve true
System.out.println(c1 == c3); //Devuelve false
```

- Método equals.
  - `public boolean equals(String str)`

```
System.out.println(c1.equals(c3)); //Devuelve true
```

Comparación de cadenas (II)

- Método equalsIgnoreCase.
  - `public boolean equalsIgnoreCase(String str)`

```
c3 = "Hola";
System.out.println(c1.equalsIgnoreCase(c3)); //Devuelve true
```

- Método compareTo.
  - `public int compareTo(String str)`
  - Devuelve 0 si las cadenas son iguales; mayor que 0 si la cadena que invoca es mayor que str o menor que 0 si la cadena que invoca es menor que str.
    - ✓ El número que devuelve es la direfencia entre los códigos del primer carácter de la cadena.

```
c3 = "Hola";
System.out.println(c1.compareTo(c3)); //Devuelve 32
c3 = "Adios";
System.out.println(c3.compareTo(c1)); //Devuelve -39
```

Comparación de cadenas (III)

- Método compareToIgnoreCase.
  - `public int compareToIgnoreCase(String str)`

```
c3 = "Hola";
System.out.println(c1.compareToIgnoreCase(c3)); //Devuelve 0
c3 = "Adios";
System.out.println(c3.compareToIgnoreCase(c2)); //Devuelve -7
```

- Comparar con StringBuffer.
  - `public boolean contentEquals(StringBuffer sb)`

```
StringBuffer sb1 = new StringBuffer("hola");
System.out.println(c1.contentEquals(sb1)); //Devuelve true
```

Métodos de String (I)

- Otros métodos:
  - `public int length()`
  - `public char charAt(int i)`
    - ✓ Devuelve el carácter contenido en la posición i. El argumento debe ser mayor o igual que 0 y menor que length() -1.
  - `public int indexOf(int c)`
    - ✓ Devuelve la posición de la primera ocurrencia del carácter c o -1 si no está.
  - `public int indexOf(int c, int inicio)`
    - ✓ Devuelve la posición de la primera ocurrencia del carácter c a partir de la posición inicio o -1 si no está.

```
c1 = "cocodrilo";
System.out.println(c1.indexOf('o')); //Devuelve 1
System.out.println(c1.indexOf('o',5)); //Devuelve 8
```

Métodos de String (II)

- `public int lastIndexOf(int c)`
  - ✓ Devuelve la posición de la última ocurrencia del carácter c o -1 si no está.
- `public int indexOf(int c, int inicio)`
  - ✓ Devuelve la posición de la última ocurrencia del carácter c a partir de la posición inicio (contando desde el final) o -1 si no está.

```
c1 = "cocodrilo";
System.out.println(c1.lastIndexOf('o')); //Devuelve 8
System.out.println(c1.lastIndexOf('o',7)); //Devuelve 3
```

- `public String toUpperCase()`
- `public String toLowerCase()`
- `public String substring(int inicio)`
- `public String substring(int inicio, int fin)`
  - ✓ Devuelve una subcadena formada a partir del carácter inicio hasta el carácter fin.
- `public String[] split(String expr)`
  - ✓ Devuelve un array de cadenas formado por subcadenas separadas a partir del separador expr.

Métodos de StringBuffer (I)

- Otros métodos:
  - `public int length()`
  - `public char charAt(int i)`
  - `public int indexOf(int c)`
  - `public int indexOf(int c, int inicio)`
  - `public int lastIndexOf(int c)`
  - `public int indexOf(int c, int inicio)`
  - `public String toUpperCase()`
  - `public String toLowerCase()`
  - `public String substring(int inicio)`
  - `public String substring(int inicio, int fin)`
  - `public String[] split(String expr)`

Métodos de StringBuffer (II)

- Gestión de la capacidad de la cadena.
  - `public int capacity()`
    - ✓ Devuelve la capacidad del buffer (no su longitud).
  - `public void ensureCapacity(int minimo)`
    - ✓ Asegura una capacidad mínima al buffer.
  - `public void setLength(int nuevaLong)`
    - ✓ Modifica la longitud de la cadena (no su capacidad).

Métodos de StringBuffer (III)

- Modificación de caracteres
  - `public void setCharAt(int pos, char car)`
    - ✓ Sustituye el carácter de la posición `pos` por el carácter `car`.
  - `public StringBuffer replace(int inicio, int fin, String str)`
    - ✓ Reemplaza los caracteres comprendidos entre `inicio` y `fin` por `str`.
    - ✓ La clase `String` tiene además el método `replace(char car1,char car2)` que sustituye todas las apariciones de `car1` por `car2`.
  - `public StringBuffer append(`*`tipo`* `valor)`
    - ✓ Añade una representación de `valor` al final del `StringBuffer`.
  - `public StringBuffer insert(int pos,` *`tipo`* `valor)`
    - ✓ Inserta una representación de `valor` en la posición `pos`.
  - `public StringBuffer delete(int inicio, int fin)`
    - ✓ Borra los caracteres situados entre la posición `inicio` y `fin`.

Conversiones de cadenas de texto

Tipo	A String	De String
`boolean`	`String.valueOf(boolean)`	`new Boolean(String).booleanValue()`
`byte`	`String.valueOf(int)`	`Byte.parseByte(String, int base)` `Byte.parseByte(String)`
`short`	`String.valueOf(int)`	`Short.parseShort(String, int base)` `Short.parseShort(String)`
`int`	`String.valueOf(int)`	`Integer.parseInt(String, int base)` `Integer.parseInt(String)`
`long`	`String.valueOf(long)`	`Long.parseLong(String, int base)` `Long.parseLong(String)`
`float`	`String.valueOf(float)`	`Float.parseFloat(String)`
`double`	`String.valueOf(double)`	`Double.parseDouble(String)`

Conversiones a/de arrays de char

- Constructores
  - `String(char[] caracts, int inicio, int longitud)`
  - `String(char[] caracts)`
- Métodos que devuelven cadenas a partir de arrays de char.
  - `public static String copyValueOf(char[]caracts)`
  - `public static String copyValueOf(char[]caracts int inicio, int longitud)`
    - ✓ Devuelve una cadena formada por los caracteres de `caracts`.
- Métodos que devuelven arrays de char a partir de una cadena
  - `public char[] toCharArray()`
    - ✓ Devuelve un array de char a partir de la cadena que invoca.
  - `public void getChars(int ini, int fin, char[] dest,int destInicio)`
    - ✓ Copia caracteres desde la posición ini a la posición fin al array dest a partir de la posición destInicio.

Conversiones a/de arrays de byte

- Constructores
  - `String(byte[] bytes, int inicio, int longitud)`
  - `String(byte[] bytes)`
- Métodos que devuelven arrays de byte a partir de una cadena
  - `public byte[] getBytes()`