Cadenas y Arrays

Arrays y cadenas de caracteres (*String*)

- Tanto los arrays como las cadenas son tipos de objetos privilegiados en Java ya que existen facilidades que simplifican su manejo
- Por ejemplo se proporcionan primitivas que simplifican su creación y su inicialización
- La clase *String* se incluye en el API estándar de Java dentro del paquete básico *java.lang*

Strings

- La clase *java.lang.String* implementa cadenas
- Las cadenas son objetos constantes e inmutables
  - Su tamaño es constante
  - Cambiar una cadena significa construir una nueva

Cadenas

- Creación de cadenas
  - String saludo = "hola"; // creación de una cadena asignando un literal
  - String cadena = new String(); // creación de una cadena vacía
  - Diversos constructores
    - String(String cadena);
    - String(char array[]);
    - String(char array[],int desplazamiento,int cuenta);
- Longitud de la cadena con el método *length()*
  - System.out.println("longitud de la cadena: " + saludo.length());

Comparación de cadenas

- Con *equals()* se sabe si dos cadenas son iguales
  - Con == se comparan referencias
  - Se diferencia entre mayúsculas y minúsculas
- Con *equalsIgnoreCase()* no se diferencia entre mayúsculas y minúsculas
- Con *compareTo()* se obtiene <0, 0, ó >0 según el argumento sea menor, igual o mayor que la cadena
- Con *toLowerCase()* y con *toUpperCase()* se convierte una cadena a minúculas o mayúsculas respectivamente

Operaciones con cadenas

- Se pueden concatenar con el operador +
  - Es el único operador sobrecargado en Java
    - saludo = saludo + " que tal";
  - Cuando se concatenan cadenas se realiza copia de los operandos
  - Una cadena se puede concatenar con cualquier otro valor de otro tipo de datos
- Con charAt(indice) se obtiene el carácter situado en la posición índice de la cadena
  - La primera posición de una cadena, si existe, es la posición 0
- Tiene otros muchos métodos de utilidad => consultar en la API de Java

Cadenas y objetos

- Con *valueOf()* se obtiene una representación de cadena de una variable u objeto
  - Conversión directa de un tipo básico en cadena
  - Llamada a *toString()* para cualquier otro objeto
- Todos los objetos tienen el método *toString()* heredado de *Object*
  - Redefiniendolo se puede obtener una representación como cadena del objeto (muy útil en depuración)

Conversión de Cadenas

```
Integer(String s); //String a Integer
Integer(int i);    //primitivo a Integer
Integer.valueOf("4"); //String a Integer
int i = 4;
Integer objInt = new Integer(i);
int j = objInt.intValue(); //Integer a
  primitivo
String s = objInt.toString(); //Integer a
  String
int k = Integer.parseInt("4"); //String a
  primitivo
```


Cadenas y objetos

```
public class Punto {
   int x;     int y;
   public Punto(int x, int y) { this.x = x;
          this.y = y;}
   public String toString() {
        return "Punto[" + x + "," + y +
              "]"; }
public static void main(String  args[]) {
    Punto punto = new Punto(2,3);
    System.out.println( "visualizar datos
          del punto"+ punto );}}
```


API de String

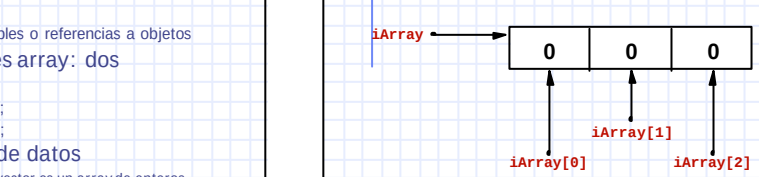
Method Summary	
char	charAt(int index) Returns the character at the specified index.
int	compareTo(Object o) Compares this String to another Object.
int	compareTo(String anotherString) Compares two strings lexicographically.
int	compareToIgnoreCase(String str) Compares two strings lexicographically, ignoring case considerat ions.
String	concat(String s) Concatenates the specified string to the end of this string.
boolean	contentEquals(StringBuffer sb) Returns true if and only if this String represents the same sequence of characters as the specified StringBuffer
static String	copyValueOf(char[] data) Returns a String that represents the character sequence in the array specified.
static String	copyValueOf(char[] data, int offset, int count) Returns a String that represents the character sequence in the array specified.
boolean	endsWith(String suffix) Tests if this string ends with the specified suffix.
boolean	equals(Object anObject) Compares this string to the specified object.
boolean	equalsIgnoreCase(String anotherString) Compares this String to another String, ignoring case considerations.

Arrays

- Colección de variables todas del mismo tipo
  - Tamaño fijo
  - Pueden ser variables simples o referencias a objetos
- Declaración de variables array: dos alternativas
  - tipoValor [] variableArray;
  - tipoValor variableArray [];
- Array de tipos básicos de datos
  - int vector[]; // vector es un array de enteros
  - int [] vector; // igual que la declaración anterior
  - int vector[10]; // ERROR: no se especifica el tamaño en la declaración

Arrays

```
int iArray[] = new int[3]; // array of 3 ints
```

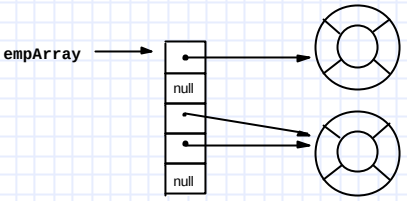


Arrays

- Definición: reserva de la memoria para el array
  - Antes de usarse, un array tiene que crearse (con *new*):
    - int vector[] = new int[10]; // array de 10 enteros: vector[0]..vector[9]
  - En este momento se especifica el tamaño del array (que no forma parte del tipo de datos)

Arrays

```
Empleado[ ]  empArray = new Empleado[5];
```

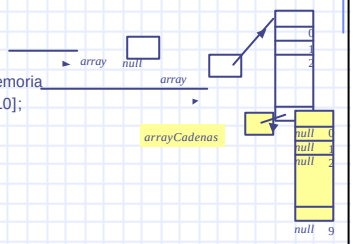


Acceso a los elementos y arrays de objetos

- Acceso a los elementos del array
  - variableArray [indice]
  - primer elemento indice 0
- Declaración de arrays de objetos
  - String S[]; // un array de cadenas -- Referencias a cadenas
  - String S,T[]; // S es una cadena y T un array de cadenas
  - String[] S,T; // Ambos, S y T, son arrays de cadenas

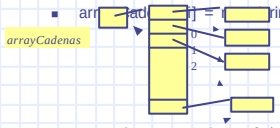
Arrays de tipos simples y de objetos

- Tipos simples
  - int [] array;
  - // reserva de memoria
  - array= new int[10];



Arrays de tipos simples y de objetos

- Objetos
  - String [] arrayCadenas;
  - // Definición: reserva de memoria para las referencias a los objetos
  - arraysCadenas = new String[10];
  - // iniacilización reserva de memoria para los objetos del array
  - for (int i=0; i<arrayCadenas.length; i++)
  - arrayCadenas[i] = new String();



Arrays

- Tamaño de un array: miembro *length*
  - A.length // correcto
  - A.length() // ERROR: no se usan paréntesis
  - for (int i = 0; i < A.length; i++) A[i]=i;
- Comprobación automática de límites del array
  - Si se intenta acceder fuera de los límites del array (entre 0 y *length*-1), se produce la excepción *IndexOutOfBoundsException*.

Arrays

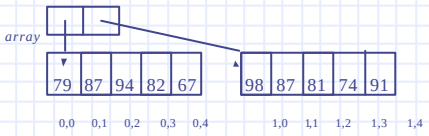
- Se puede especificar una lista de inicialización
  - int[] arrayNumeros = { 147, 323, 89 };
  - No se utiliza el operador *new* y no se especifica el tamaño
- El paso de los arrays a los métodos es por referencia
  - Se pasa la referencia al array (como con todos los objetos)
- El tamaño del array no es parte del tipo
  - Una variable de tipo String[] puede almacenar cualquier array de cadenas de cualquier tamaño

Arrays multidimensionales

- Array de arrays
  - Declaración, definición e inicialización
    - int[][] array; // creación de la referencia
    - array = new int[2]; // array con dos filas
    - for (int i = 0; i < array.length; i++)
    - array[i] = new int[5]; // cada fila puede tener distinto tamaño
  - Cada fila puede tener un tamaño diferente
    - Si todas las filas tienen el mismo tamaño
      - array = new int[2][5]; // lo mismo que las dos sentencias anteriores
  - Se permiten declaraciones parciales en las que al final pueden existir dimensiones sin especificar
    - int[][] arrayPar = new int[3][];

Arrays multidimensionales

- Inicialización mediante listas
  - int[][] array = { {79, 87, 94, 82, 67}, {98, 87, 81, 74, 91} };



Arrays multidimensionales

- Una vez que se reserva el espacio de memoria el tamaño de un array no puede cambiar
  - String[] miArray = new String[20];
  - miArray = new String[10]; // correcto se pierde la referencia al array anterior
  - miArray.length = 30; // ERROR - no permitido

Arrays

- Límites de los arra.y
  - Todos los vectores empiezan por 0.
  - El número de elementos del vector se almacena en el atributo length del propio vector.
- Un array no se puede redimensionar
- Se puede utilizar la misma variable referencia para apuntar a otro vector diferente:

```
int elements [ ] = new int [6];
elements = new int [10];
```