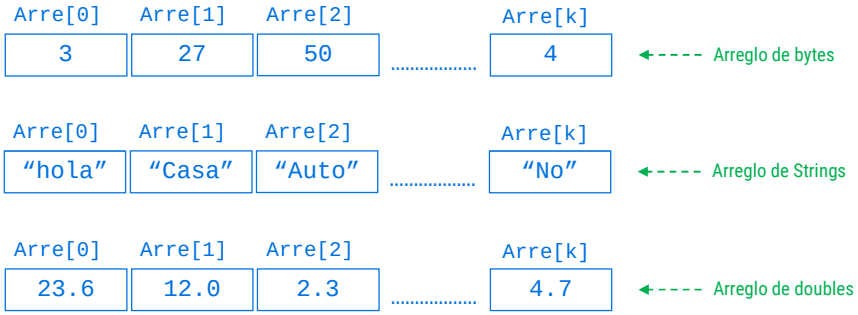
Arreglos y matrices en Java

Prof. Mg. Rafael Mellado S.
EII147 – Introducción a las tecnologías de información
Escuela de Ingeniería Industrial
Pontificia Universidad Católica de Valparaíso

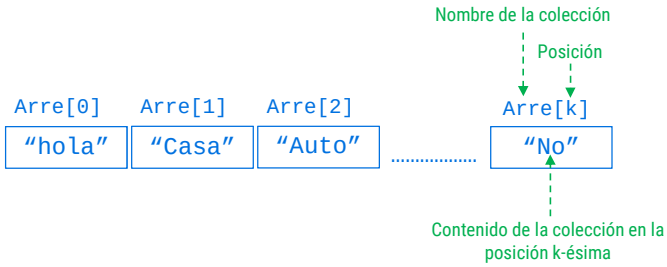
Arreglos en Java

- Los arreglos son colecciones ordenadas de datos del mismo tipo.
- Ejemplos:



Arreglos en Java

- Cada arreglo se reconoce por un identificador y cada dato se almacena en una posición **indexada**.
- Un arreglo de largo N, tiene posiciones indexadas mediante enteros desde 0 a N-1.



104

Creación de arreglos

- Se debe definir una variable que identifique al arreglo, indicando la naturaleza de los datos que se almacenarán:

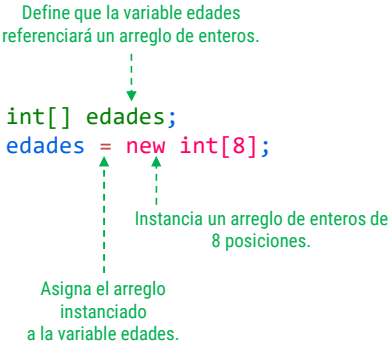
```
tipo[ ]  variable;
tipo variable[ ];
```

- Luego se debe instanciar el arreglo indicando el largo que tendrá, y asignarlo a la variable:

```
variable = new tipo[ entero ];
```

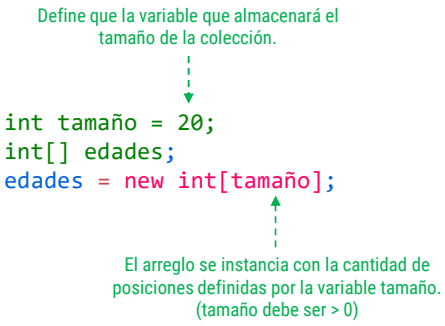
105

Creación de arreglos



106

Creación de arreglos



107

Creación de arreglos

- Forma abreviada de creación de arreglos:
  - La definición de variable, instanciación del arreglo y su asignación a la variable puede realizarse en una sola instrucción.

```
tipo[ ]  variable = new tipo[ entero ];
```

  - Ejemplo:

```
double[] nota = new double[4];
```

108

Creación de arreglos

- Asignación y creación literal de arreglos:
  - También es posible instanciar arreglos escribiéndolos como literales en el código fuente:

```
int[] nota = { 23, 14, 55, 18 };
```

Instancia un arreglo de enteros de tamaño 4

nota[0]	nota[1]	nota[2]	nota[3]
23	14	55	18

109

Recorrido de arreglos

- Se puede utilizar la propiedad **length** del arreglo para controlar procesos iterativos sobre el mismo:

```
...
int i;
long[] números;
números = new long[ 20 ];
...
i=0;
while( i < números.length ) {
   System.out.println( números[ i ] );
   i++;
}
...
```

110

Recorrido de arreglos

- Cuando se recorre arreglos se presenta un error típico:
  - Tratar de acceder una posición inexistente del arreglo, por ejemplo, la posición 10 de un arreglo de largo 10.
  - Cuando lo anterior ocurre, se genera en tiempo de ejecución una excepción denominada: **ArrayIndexOutOfBoundsException**.
  - Se debe recordar que los arreglos de tamaño n son recorridos desde la posición **0** a la posición **n-1**.

111

Consideraciones

- Una vez instanciado un arreglo, no puede modificarse su largo.
- **length** es una propiedad o atributo del arreglo que contiene el largo del mismo.

```
double[] nota;
nota = new double[4];
System.out.println( "El largo es " + nota.length );
```

Las propiedades o atributos se consultan sin paréntesis al final, a diferencia de los métodos.

112

Consideraciones

- Sobre el arreglo de parámetros declarado al inicio del método main es necesario destacar:
  - Es instanciado por Java al momento de ejecutarse la aplicación.
  - El arreglo se instancia con un largo igual a la cantidad de parámetros traspasados en la línea de comandos.
  - El arreglo de parámetros debe ser declarado como arreglo de Strings.
  - En este curso no se hará uso de dicho arreglo.

```
public static void main(String arg[ ]) throws   IOException
```

113

Consideraciones

- Los arreglos se instancian. La instanciación ocurre de tres formas:
  - cuando se utiliza el operador **new**.
  - cuando el arreglo es declarado literalmente.
  - al ejecutar la aplicación, en el caso del arreglo de parámetros del método main.
- Los arreglos son referenciados desde una variable.
- Todas las posiciones del arreglo son del mismo tipo.
- El atributo **length** permite acceder al largo del arreglo.
- Y tratar de acceder una posición inexistente del arreglo genera una excepción **ArrayIndexOutOfBoundsException**.

114

Matrices

- Un arreglo puede tener más de una dimensión.
- Los arreglos de más de una dimensión se denominan matrices.
- El caso más común es la matriz bidimensional:

12	-3	4	55
4	700	-8	0
1	0	-2	14

Matrices

- Java permite crear matrices de la siguiente forma:

```
tipo[ ] [ ] variable = new tipo[ entero1 ] [ entero2 ];
```

  - Por ejemplo:

```
int[ ][ ] utilidad;
utilidad = new double[10][15];
```

  - También pueden ser creadas de forma literal (al igual que los arreglos):

```
double[ ][ ] uti = { {-1, 7, 15}, {3, 0, 2}, {4, -3, 12} };
```

uti[1][2] contiene un 2.

116

Matrices

- Todas las posiciones de una matriz son de un mismo tipo.
- La matrices se identifican por su nombre, y se caracterizan además por sus dimensiones.
- Las dimensiones de una matriz no se pueden modificar.

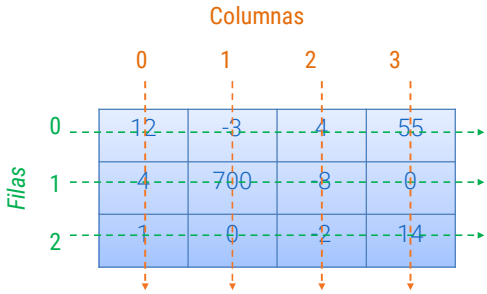
mediciones

12	-3	4	55
4	700	-8	0
1	0	-2	14

Matriz *mediciones* de datos enteros, de dimensiones 3 x 4

Matrices

- En el caso de las matrices de dos dimensiones, llamaremos a la primera dimensión “fila”, y a la segunda, “columna”.
- Representaremos visualmente en nuestros ejemplos las filas “hacia abajo” y las columnas “hacia el lado”.
- La primera fila será la fila 0 y la primera columna, la columna 0.
- Ejemplo: Matriz de 3 filas y 4 columnas



118

Acceso a posiciones

- Las posiciones de una matriz se acceden independientemente (una a una).
- Se utiliza un subíndice para especificar cada dimensión.
- En una matriz de dos dimensiones, el primer subíndice representa la fila, y el segundo, la columna.

mediciones

12	-3	4	55
4	700	-8	0
1	0	-2	14

mediciones[0][0] contiene un 12
mediciones[0][3] contiene un 55
mediciones[2][0] contiene un 1
mediciones[1][2] contiene un −8
mediciones[2][1] contiene un 0
etc...

119

Particularidades

- En rigor Java no provee arreglos multidimensionales. Java los implementa mediante arreglos de arreglos.
- Esto permite crear, por ejemplo, matrices bidimensionales con cantidades de posiciones distintas por fila:

```
int[][] nota = { {-1,7,15}, {3, 2}, {4,-3,12}, {3}};
```

120

Manejo de Strings

Prof. Mg. Rafael Mellado S.
EII147 – Introducción a las tecnologías de información
Escuela de Ingeniería Industrial
Pontificia Universidad Católica de Valparaíso

String en Java

- String es una clase, no un tipo primitivo.
- Las variables de tipo String, en cualquier instante, pueden contener:
  - Un String (ej. “Hola”, “11/11/2011”, “123”, “”, etc.)
  - El valor null (cuando la variable no contiene un String).
- Observaciones:
  - Cuando se instancia un arreglo de String, sus posiciones se inicializan automáticamente en null.
  - El valor null puede ser asignado a cualquier variable String.

Valor null

– Para saber si una variable de tipo String contiene el valor null:

```
String x;

//Aquí x puede o no recibir un valor
...

if( x == null )
    System.out.println("No contiene String");
else
    System.out.println("Sí contiene String");
```

123

Método equals

– Para saber si una variable de tipo String (no **null**) contiene un determinado texto, debemos usar el método **equals** sobre la variable:

```
String x;

//Aquí x recibe un valor
...

if( x.equals("Esternocleidomastoideo") )
    System.out.println("Igual");
else
    System.out.println("Distinto");
```

124

Método equals

– Para establecer si dos variables String (no null) contienen el mismo valor:

```
String x, y;

//Aquí x e y reciben valores
...

if( x.equals( y ) )
    System.out.println(“Igual”);
else
    System.out.println(“Distinto”);
```

125

Método equals

– Para establecer si dos variables String (no null) contienen valores distintos, se niega el resultado del método equals:

```
String x, y;

//Aquí x e y reciben valores
...

if( ! x.equals( y ) )
    System.out.println(“Distintos”);
else
    System.out.println(“Iguales”);
```

126

Método equalsIgnoreCase

- El método equals distingue entre mayúsculas y minúsculas.
- Si no deseamos hacer distinción entre mayúsculas y minúsculas debemos utilizar el método **equalsIgnoreCase**.

```
String x = "Esternocleidomastoideo";
String y = "EsternocleidoMastoideo";

if( x.equals( y ) )
    System.out.println("Iguales");
else
    System.out.println("Distintas"); -----► Distintas

if( x.equalsIgnoreCase( y ) )
    System.out.println("Iguales"); -----► Iguales
else
    System.out.println("Distintas");
```

127

Consideración **muy importante**

- Para aplicar cualquiera de los métodos anteriores sobre una variable, es necesario que ésta no sea **null**. Para verificar esta condición:

```
String x, y;

//Aquí x e y  pueden o no recibir un valor
...

if( x != null )
    if( x.equals( y ) )
       System.out.println("Igual");
...
// o en forma alternativa (en este orden):
if( x != null && x.equals( y ) )
     System.out.println("Igual");
```

128

Método substring

- Entrega una porción de un String, delimitada por subíndices de posición de sus caracteres. Ejemplo:

```
String nombre = "Nogatongamegalosomanjarchafafrinilofo";
String parte= nombre.substring( 2, 5 ); -----► Gato
System.out.println( parte );
```

- Nota: el primer caracter ocupa la posición 0.

129

Método trim

- Entrega un String con el mismo contenido, pero con el espacio inicial y final eliminado.

```
String queja = "    odio el espacio que me rodea     " ;
System.out.println( "(" + queja + ")" ); -----► (    odio el espacio que me rodea    )
```

```
String solución = queja.trim();
System.out.println( "(" + solución + ")" ); -----► (odio el espacio que me rodea)
```

130

Método replaceAll

- Entrega un String que reemplaza las porciones que coinciden con regex, por el String de reemplazo.

```
String dato = "LA CASA DEL CASADO" ;
String otroDato = dato.replaceAll( "AS", "OL" );
System.out.println( otroDato ); -----► LA COLA DEL COLADO
```

131

Método split

- Entrega un arreglo de Strings que contiene en cada posición una porción del String original, considerando como divisor a regex.

```
String familia = "Juan;Ana;Camila" ;
String personas = familia.split(";");
System.out.println( personas[0] ); -----► Juan
System.out.println( personas[2] ); -----► Camila
```

132

Secuencias de escape

- El backslash ("\") se utiliza dentro de de String literales para iniciar secuencias de escape.
- Secuencias de escape importantes:
  - "\n": cambio de línea
  - "\t": tabulación
  - "\'" : comilla simple
  - "\"": comilla dobe
  - "\\": backslash

133