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
```

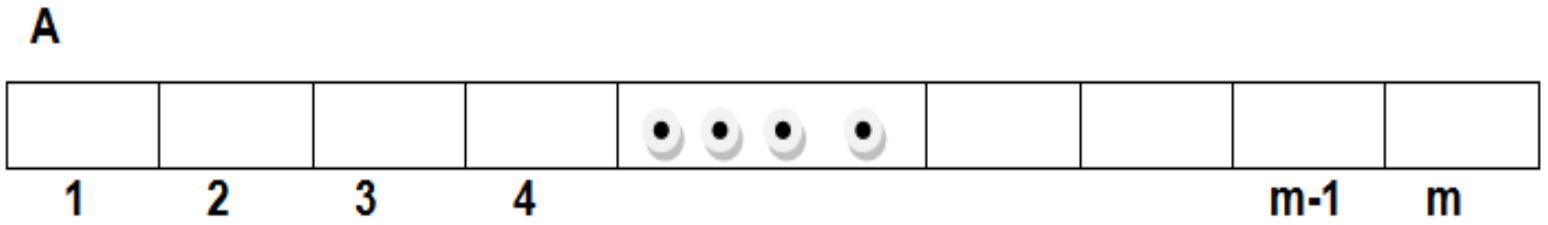
- Definición: reserva de la memoria para el array
  - Antes de usarse, un array tiene que crearse (con *new*):

```
int vector[] = new int[10];     // array de 10 enteros: vector[0]..vector[9]
```

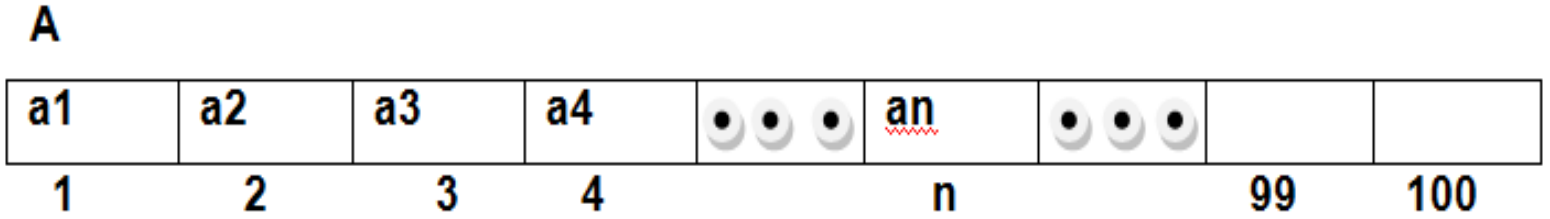
  - En este momento se especifica el tamaño del array (que no forma parte del tipo de datos)

ARREGLOS UNIDIMENSIONALES

Representación Gráfica:



A es un arreglo vació de m cajones



A es un arreglo de 100 cajones con n elementos n<=100

ARREGLOS UNIDIMENSIONALES

Propiedades

Finito, porque tiene un número limitado de cajones

Homogéneo, porque el tipo de dato o componente tiene que ser el mismo para todos los elementos.

Ordenado, porque cada cajón tiene un índice desde uno hasta el número límite m que está predeterminado y es invariable.

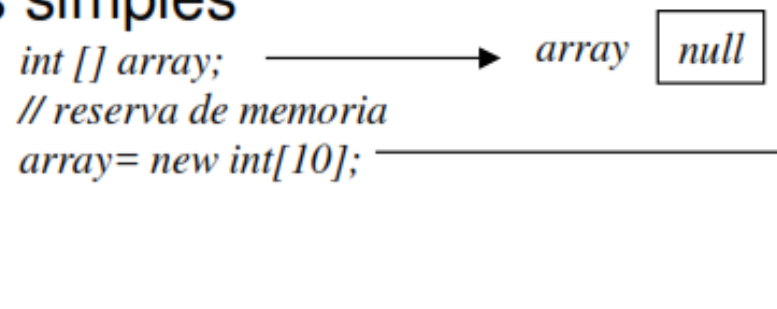
Acceso a los elementos y arrays de objetos

- 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

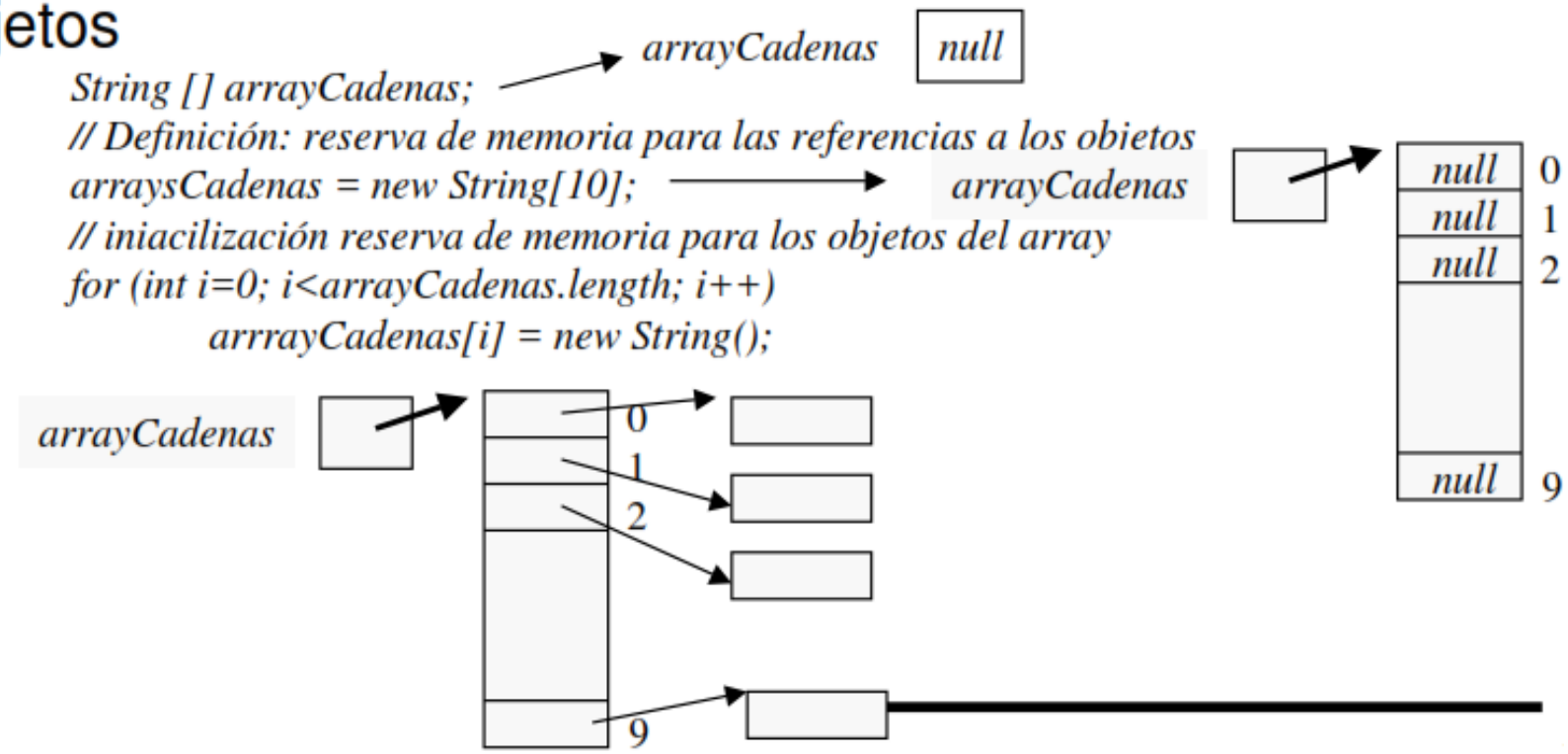
- Tipos simples

```
int [] array;
// reserva de memoria
array= new int[10];
```



- Objetos

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

EJERCICIOS

Problema 1.

Desarrollar un algoritmo que permita:

- Generar un arreglo de 15 casillas de memoria.
- Donde el valor de las celdas de memoria es directamente proporcional al cuadrado de los 15 primero números naturales.

EJERCICIOS

Problema 2.

Desarrollar un algoritmo que permita:

❖Generar un arreglo de 20 casillas de memoria y almacenar datos de manera aleatoria.

Se pide:

✓Determinar el promedio.

✓Cuantos elementos del arreglo son mayores y menores al promedio.

EJERCICIOS

Problema 3.

Desarrollar un algoritmo que permita:

❖Generar un arreglo de 35 casillas de memoria e ingresar datos de manera aleatoria.

Se pide:

✓Cuantos números de una cifra múltiplos de 3, así como indicar la casilla de memoria ubicada.

✓ Cada proceso mostrarlo de manera independiente.

EJERCICIOS

Problema 4.

Desarrollar un algoritmo que permita:

❖Generar un arreglo de 20 casillas de memoria e ingresar numero enteros de tres cifras formado por dígitos diferentes.

Se pide:

✓Generar dos nuevos arreglos, en donde se almacenaran como datos el mayor digito de cada numero y la suma de los dígitos de cada numero, al final mostrar los 3 arreglos.