Ministerio de Producción
Presidencia de la Nación

Ministerio de Educación y Deportes

Subsecretaría de Servicios Tecnológicos y Productivos

Programa 111 mil
VOS PODÉS SER UNO.

Arreglos en JAVA

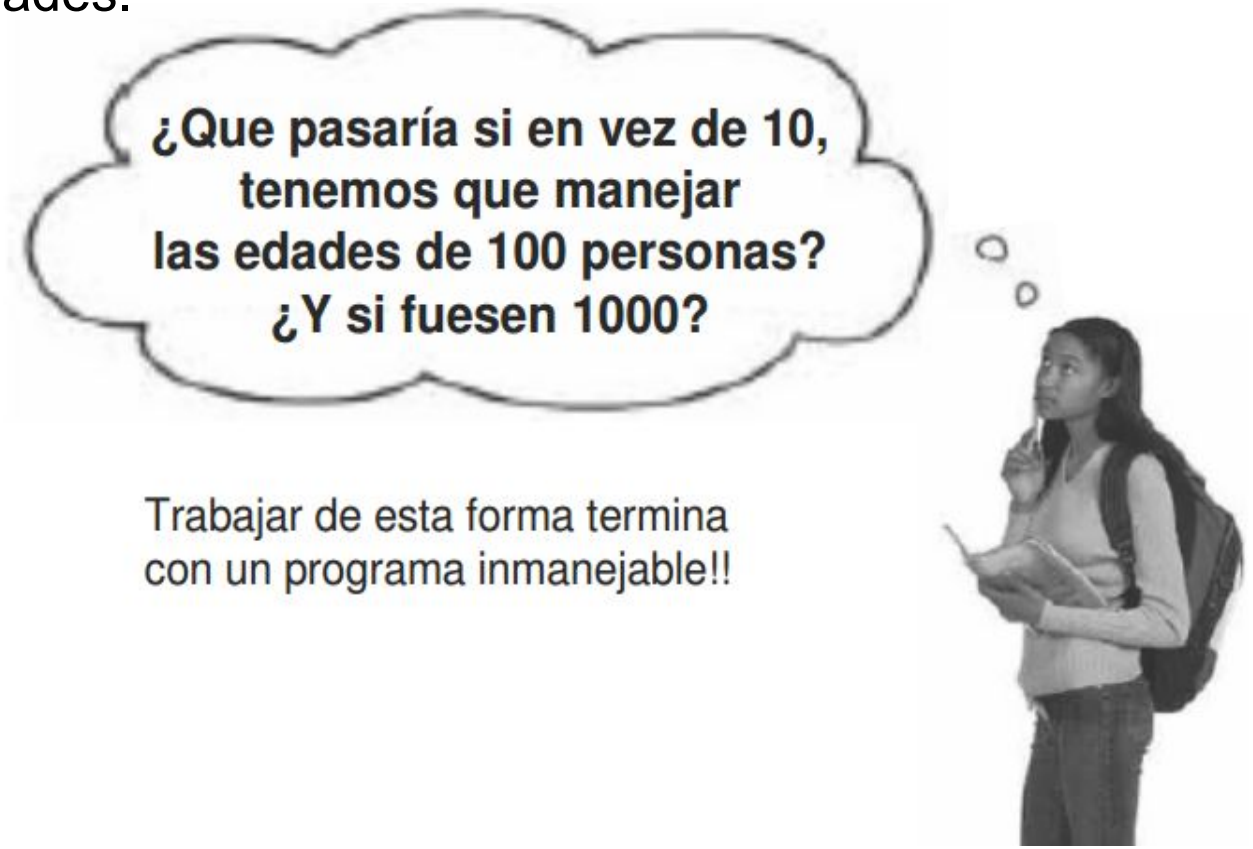
Temas:

- Arreglos de tipos primitivos y de Objetos
- Beneficios y limitaciones del uso de Arreglos
- Sentencia de iteración: for each
- Matrices: Arreglos bidimensionales

Arreglos

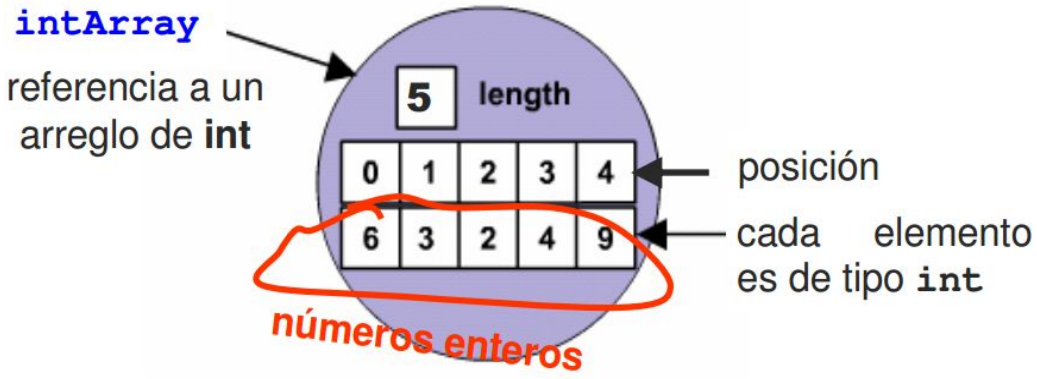
Supongamos que necesitamos hacer un programa que maneje las edades de 10 personas. Una posible solución a nuestro problema es definir 10 variables para almacenar cada uno de las 10 edades.

```
int edadUno = 10;
int edadDos = 32;
int edadTres = 44;
int edadCuatro = 39;
int edadCinco = 56;
int edadSeis = 11;
int edadSiete = 18;
int edadOcho = 8;
int edadNueve = 23;
int edadDiez = 45;
```

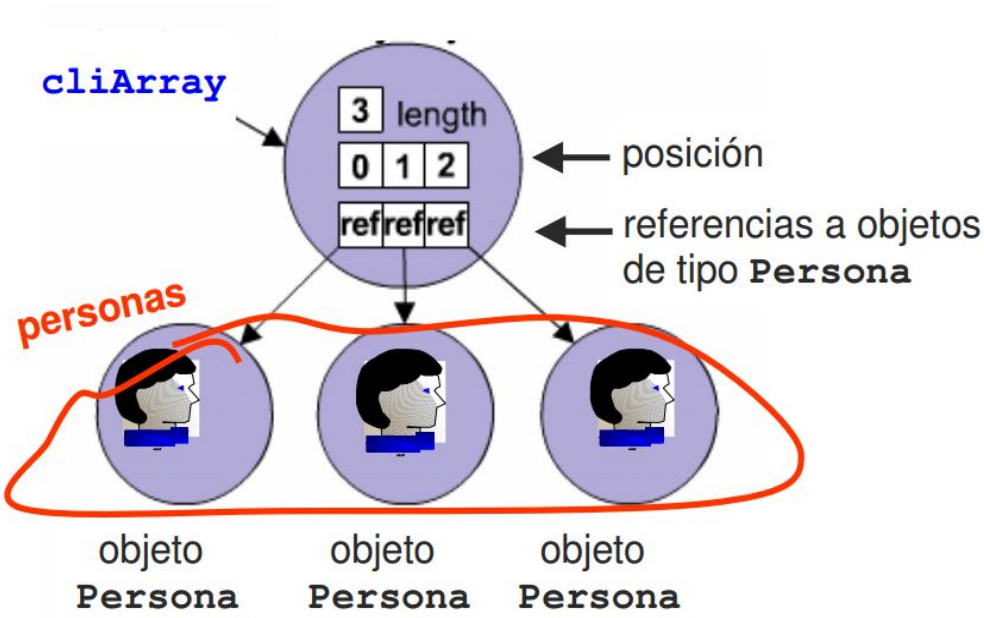


Java permite agrupar múltiples valores del mismo tipo usando arreglos unidimensionales

Arreglo de primitivos

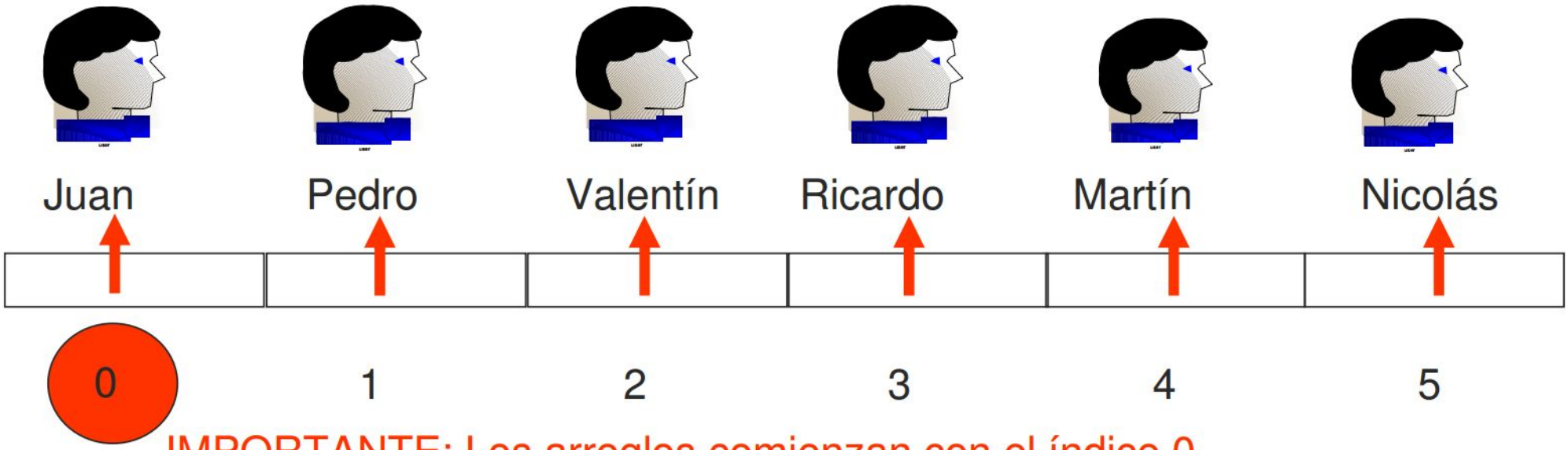


Arreglo de objetos



Un arreglo es una disposición **ordenada** de **cosas**

Supongamos que tenemos un arreglo de personas. Cada parte del arreglo es un elemento. Se puede acceder a cada uno de los elementos del arreglo por su posición, llamada índice



IMPORTANTE: Los arreglos comienzan con el índice 0.

Sintaxis para declarar un arreglo ***unidimensional***

tipo[] identificadorArreglo;

Donde:

- **tipo:** tipo de dato primitivo o al tipo del objeto
- **los corchetes([]):** informan al compilador que se está declarando un arreglo
- **identificadorArreglo:** es el nombre del objeto arreglo

Declaración de un arreglo **edades** cuyos elementos son de tipo **int** **int [ ] edades;**	Declaración de un arreglo **palabras** cuyos elementos son de tipo Objetos **String** **String [ ] palabras;**
Declaración de un arreglo **letras** cuyos elementos son de tipo **char** **char [ ] letras;**	Declaración de un arreglo **personas** cuyos elementos son de tipo Objetos **Persona** **Persona [ ] personas;**

También es posible declarar un arreglo con la siguiente sintaxis

```
tipo [ ] IdentificadorArreglo ;
```

Cuando se declara un arreglo el compilador y la Máquina Virtual de Java (JVM) no conocen el tamaño, dado que se declaró la variable de referencia y no se la inicializó.

```
tipo[] IdentificadorArreglo = new tipo [largo] ;
```

Utilización de la palabra clave new para la instanciación como cualquier objeto

Indicación de la cantidad de elementos que contiene como máximo el arreglo

Instanciacion de un arreglo edades de **int** de 100 elementos `edades = new int[100];`	Instanciacion de un arreglo palabras de **Strings** de 186 elementos `palabras = new String[186];`
Instanciacion de un arreglo letras de primitivos **char** de 24 letras `letras = new char[24];`	Instanciacion de un arreglo personas de objetos Persona de 500 elementos `personas = new Persona[500];`

Cuando se instancia un arreglo de primitivos, cada elemento es inicializado de la siguiente manera:

- numérico (int, float, etc.) es inicializado con el valor **0 (cero)**
- char, cada elemento es inicializado con el valor **\u0000** (caracter null en UNICODE)
- boolean, cada elemento es inicializado con valores **FALSE**

Para un arreglo de Objetos (String, Date, Persona, etc.) cada elemento del arreglo es inicializado con **NULL**.

Después de la creación de un objeto arreglo, se pueden llenar sus elementos por posición:

```
identificadorArreglo [indice]= valor ;
```

Inicialización del arreglo **edades**

```
edades[0]=24;
edades[1]=2;
edades[2]=65;
```

Inicialización del arreglo **palabras**

```
palabras[0]="Hola";
palabras[1]="Mundo";
palabras[2]=new String("Casa");
```

Inicialización del arreglo **letras**

```
letras[0]='a';
letras[1]='b';
letras[2]='z';
```

Inicialización del arreglo **personas**

```
personas[0]=new Persona();
personas[1]=new Persona("Juan");
personas[2]=new Persona("Pedro",25);
```

utilización de constructores sobrecargados

Si al momento de declarar un arreglo se conocen los valores que se desean almacenar, se puede realizar su declaración, instanciación e inicialización en la misma línea de código.

```
tipo[]idArreglo= { lista de valores o expresiones
                   separadas por " , "};
```

Donde:

- **tipo:** representa al tipo de dato primitivo o al tipo del objeto.
- **los corchetes([]):** informan al compilador que se está declarando un objeto
- **idArreglo:** es el nombre que se está asignando para referirse al objeto
- **Lista de valores o expresiones separados por coma:** representa la lista de valores que se desea almacenar en el arreglo o una lista de expresiones cuyos resultados serán almacenado en el arreglo.

Inicialización del arreglo **edades**

```
edades[0]=24;
edades[1]=2;
edades[2]=65;
```

Inicialización del arreglo **palabras**

```
palabras[0]="Hola";
palabras[1]="Mundo";
palabras[2]=new String("Casa");
```

Inicialización del arreglo **letras**

```
letras[0]='a';
letras[1]='b';
letras[2]='z';
```

Inicialización del arreglo **personas**

```
personas[0]=new Persona();
personas[1]=new Persona("Juan");
personas[2]=new Persona("Pedro",25);
```

utilización de constructores sobrecargados

Cada elemento de un arreglo es accedido usando su índice

Para acceder a un valor en un arreglo, hay que indicar dentro de los corchetes el número de índice.

Si tenemos este ejemplo:

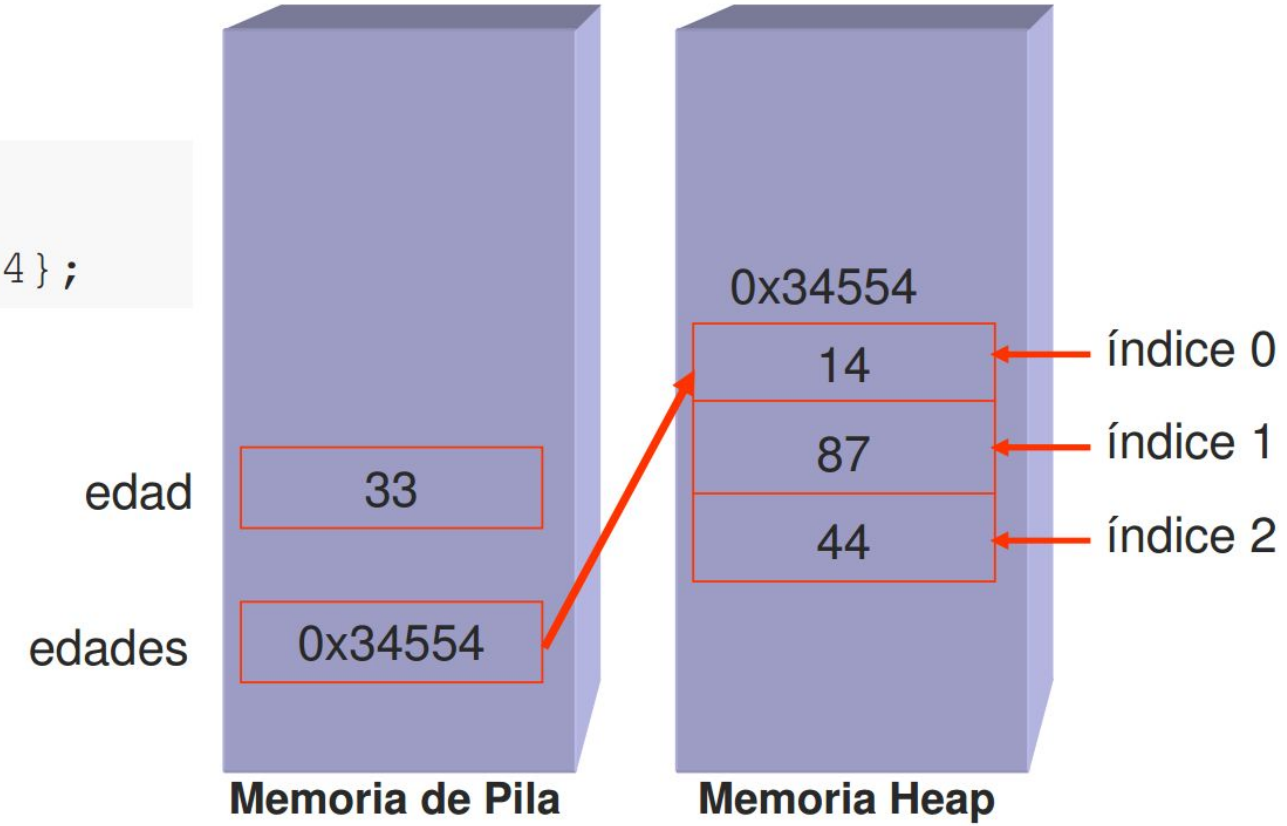
```
int[ ] edades = {24,2,65};
```

1. ¿Cómo hacemos para acceder al valor 24:?
2. ¿Qué retorna la siguiente línea?: edades[2]
3. Y ¿la siguiente línea?: edades[3]

¿Cómo se almacenan los arreglos en memoria?

```
int edad = 33;

int[] edades = {14,87,44};
```



¿Cómo se almacenan los arreglos en memoria?

```
Persona persona = new Persona("Valentin",25);
Persona [] personas = {new Persona("Jose",14), new Persona(Juan,65)};
```

Todos los objetos Array tienen una variable atributo **length** que contiene el **largo** del arreglo.

Para el siguiente ejemplo

int [] edades = new int [10] los límites del mismo son:

A. Desde: edades[0] hasta edades [10]
B. Desde: edades[1] hasta edades [10]
C. Desde: edades[1] hasta edades [9]
D. Desde: edades[0] hasta edades [9]

for each

Esta sentencia de iteración se incorporó en J2SE 5.0

Supongamos que tenemos un arreglo con los primeros 10 naturales:

int[] numeros ={1,2,3,4,5,6,7,8,9,10}

y queremos imprimirlo:

```
public void listar (int[] num){
for (int i: num)
System.out.print(i);
System.out.print(“ ”);
}
```

Se usan : en lugar de ;

La variable i no es un índice, sino que representa cada uno de los elementos del arreglo.

Básicamente se trata de una simplificación al momento de escribir código, ya que el compilador convierte el código en una sentencia for convencional:

```
public void listar (int[] num){
for (int i = 0; i<num.length; i++)
System.out.print(num[i]);
System.out.print(“ “);
}
```

Como podemos observar en el ejemplo e es de tipo Empleado y empleados es un arreglo de Empleado

```
Empleado[] empleados= new Empleado[2];
Empleado e = new Empleado();
e.setNombre("Juan");
Empleado e1 = new Empleado();
e1.setNombre("Pedro");
empleados[0]=e;
empleados[1]=e1;
public void listar (){
for (Empleado e :empleados)
System.out.println(e.getNombre());
```

Ejemplo:

```
public double sum(double[ ] ar){
double sum = 0.0;
for (double d : ar) {      d contiene cada valor de ar
sum = sum+d;
    }
}
```

Dado el siguiente arreglo:
double[] valores = {1.2, 3.0, 0.8};

¿Qué devuelve si invocamos sum(valores)?

Matrices: Arreglos bidimensionales

Un **arreglo bidimensional** es similar a una planilla o tabla con múltiples filas y múltiples columnas (cada columna representa un arreglo o lista de ítem).

Los **arreglos bidimensionales** requieren un par de corchetes adicionales. El proceso para crear y usar **arreglos bidmensionales** es el mismo que para crear arreglos unidimensionales.

- Una matriz se declara de la siguiente manera: int [ ][ ] notas;
- Una matriz se instancia de la siguiente manera: new notas [12][3];

Ejemplo

Supongamos que queremos representar un boletín de calificaciones de un alumno del colegio secundario y que cursa 12 materias en cada uno de los 3 trimestres del ciclo lectivo.
Consideremos que las notas son números enteros.
Declaremos una matriz para almacenar las notas:
int[ ][ ] notas;
Ahora instanciamos la matriz:
notas = new notas[12][3];
El primer corchete nos indica la cantidad de filas y el segundo la cantidad de columnas.
Si queremos declarar e inicializar una matriz de notas en la misma línea de código:
int[ ][ ] notas = {{9,0,0},{8,0,0},{7,0,0}, {6,0,0}};

	1º semestre	2º semestre	3º semestre
Materia 1			
Materia 2			
Materia 3			
Materia 4			
Materia 5			
Materia 6			
Materia 7			
Materia 8			
Materia 9			
Materia 10			
Materia 11			
Materia 12			

Vamos a inicializar el boletín utilizando la sentencia **for**:

```
for (int i=0; i<12,i++){
    for (int j=0; j< 3,j++){
            notas[i][j]=0;
        }
}
```

Para imprimir en pantalla la nota de la materia 1 del segundo semestre, hacemos:

```
System.out.println(notas[0][1]);
```

¿Cómo podríamos generalizar esta inicialización?

```
for (int i=0; i<notas[j].length,i++){
    for (int j=0; j<notas[i].length,j++){
            notas[i][j]=0;
        }
    }
```

Supongamos que se aprueba con 6 e imprimimos en pantalla TODAS las materias que el alumno desaprobó.

```
for (int i = 0; i<notas.length; i++)
    for (int j=0;j<notas[i].length; j++)
        if (notas[i][j]<6)
            System.out.println("Fila "+i+" Columna "+j);
```

¿Y si queremos recorrer las materias hasta encontrar SOLO UNA desaprobada?

```
int i,j=0;
boolean ok = false;
while ((i<notas.length) && (!ok)){
    while ((j<notas[i].length) && (!ok))
        if (notas[i][j]<6) ok=true;
        else j++;
    if (!ok) i++;
}
if (ok) System.out.println("Fila "+i+" Columna "+j);
else System.out.println("El alumno NO desaprobó ninguna materia");
```

Ejemplo completo de matrices

```
public class Boletin{
private int [][] notas;
private int cantMaterias = 12;
private String[] trimestres ={"1º semestre", "2º semestre","3º
semestre"};
private String[] materias = {"matemática", "educación
cívica","biología", "música", "lengua", "historia", "geografía",
"inglés", "física", "educación física", "dibujo", "química" };
public Boletin(){
   this(materias);
}
public Boletin (String[] materias){
   this.materias=materias;
   this.cantMaterias=materias.length;
   this.notas=new int[cantMaterias][3];
}
public int[][] getNotas() {
   return notas;
}
```

```
public void actualizar(int nota, int trimestre, String materia){
  int i = 0;
  boolean encontre = false;
  while ((i<materias.length)&&(! encontre)){
    if (materias[i].equals(materia)) encontre=true;
    else i++;
  }
  if (encontre) notas[i][trimestre-1]=nota;
}

public int getCantMaterias() {
  return cantMaterias;
}

public String[] getMaterias() {
   return materias;
}
}//Fin clase Boletin
```

Ejemplo completo continuación

```
public class TestBoletin {
    public static void main(String[] args) {
    String[] mate = {"matemática", "lengua y literatura", "historia"};
    Boletin b = new Boletin(mate);
    b.actualizar(10,1,"matemática");
    b.actualizar(9,1,"lengua y literatura");
    b.actualizar(8,1,"historia");
    b.actualizar(4,2,"matemática");
    b.actualizar(5,2,"lengua y literatura");
    b.actualizar(6,2,"historia");
    b.actualizar(7,3,"matemática");
    b.actualizar(7,3,"lengua y literatura");
    b.actualizar(7,3,"historia");
    int[][] notas = b.getNotas();
    for (int i=0;i<b.getCantMaterias();i++){
        System.out.print(b.getMaterias()[i]+" ");
        for (int j=0;j<3;j++)
                System.out.print(notas[i][j]+" ");
        System.out.println("");
        }
    }
}
```

matemática 10 4 7
lengua y literatura 9 5 7
historia 8 6 7