

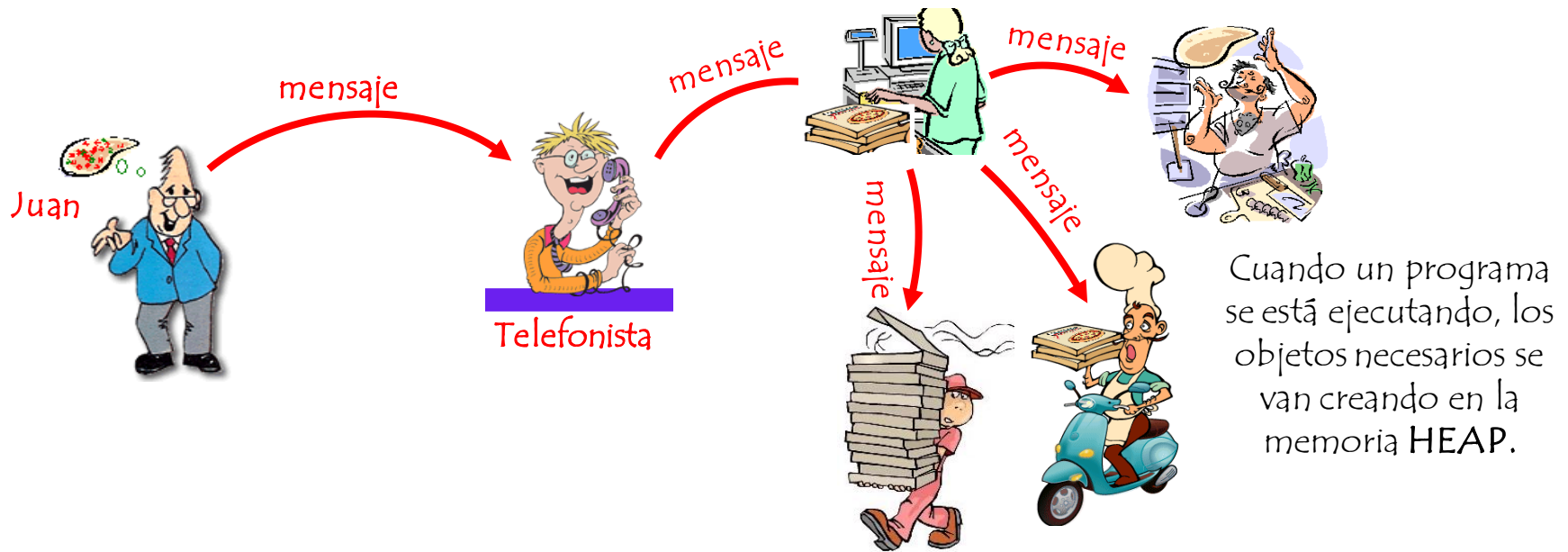
Clase 2

Definición de clases e instanciación

- Definición de una clase java
 - Variables de instancia
 - Métodos de instancia
 - Tipos de variables: referenciales y primitivas
- Instanciar una clase java.
 - El operador **new ()**
- Variables y métodos de clase
- Arreglos

Repaso

- Los programas orientados a objetos están compuestos por varios **objetos**. Estos objetos se comunican entre si mediante el envío de **mensajes**.

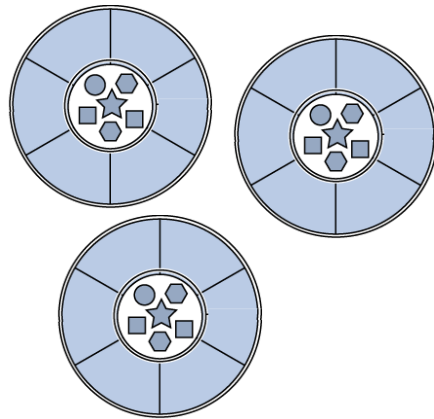


- Cada uno de estos objetos es una entidad de software que combina un **estado o datos** y **comportamiento o métodos**.
- Estos objetos se crean a partir de un molde o **clase**.

¿Cómo definir/declarar una clase en java?

- Una clase java es un **bloque de código o un molde**, que describe el estado y el comportamiento que tendrán los objetos que con ella se creen.
- Un **archivo origen java** debe tener como mínimo:
 - en la primer línea la palabra clave **package** seguida del nombre del paquete.
 - la palabra clave **class** seguida del nombre de la clase.
- Un **archivo origen java** debe guardarse con el mismo nombre que la clase (y con extensión **.java**). Se deben respetar las mayúsculas.

objetos Cliente



Cliente.java

```
package pizzeria;

public class Cliente {
    private int clienteId;
    private String domicilio;
    private String telefono;

    public String getDomicilio() {
    }
    public void setDomicilio(String s) {
    }
    // más métodos de instancia
}
```

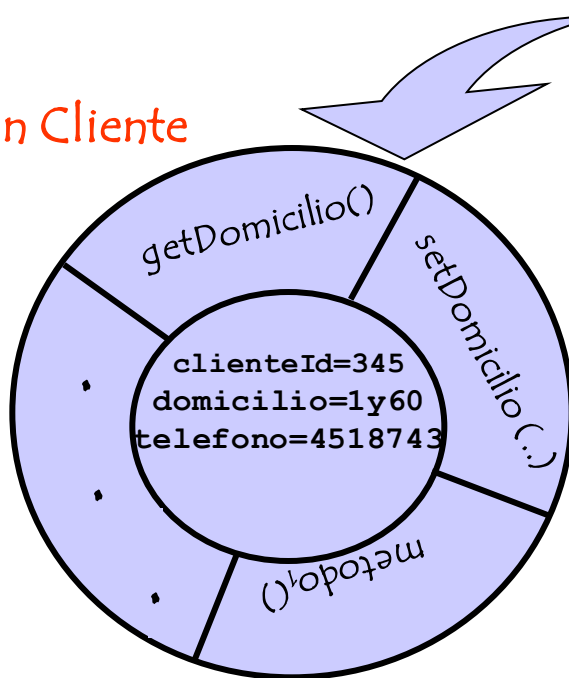
¿Cómo incorporar estado y comportamiento a una clase? (1/2)

Se debe agregar en el cuerpo de la misma:

- **variables de instancia:** constituyen el estado de un objeto. Normalmente, las variables de instancia se declaran **private**, lo que significa que sólo la clase puede acceder a ellas, directamente.
- **métodos de instancia:** definen las operaciones que pueden realizar los objetos de un tipo de clase. Un método es un bloque de código, similar a lo que es una función o procedimiento en los lenguajes procedurales, como PASCAL.

Cliente.java

un Cliente



```
package pizzeria;
```

```
public class Cliente {
```

```
    private int clienteId;  
    private String domicilio;  
    private String telefono;
```

Estado

(variables de instancia)

```
    public String getDomicilio() {  
    }  
    public void setDomicilio(String s) {  
    }  
    // más métodos de instancia
```

Comportamiento

(métodos de instancia)

```
}
```

¿Cómo incorporar estado y comportamiento a una clase? (2/2)

- ◇ En java cada método y cada variable, existen adentro de una clase: **java no soporta funciones o variables globales.**
- ◇ No tiene importancia el orden en que se ubican las variables y los métodos. No obstante, si se ordenan las partes, será más sencillo depurar, corregir y compartir código.

variables de instancia

La declaración de una variables de instancia debe incluir:

- un **identificador** (nombre de la variable).
- un **tipo** (tipo primitivo o de un tipo de una clase).
- un **modificador de acceso**: **public** o **private**. (opcional).

métodos de instancia

La declaración de un método de instancia debe especificar:

- un **nombre**
- una lista opcional de **argumentos**
- un **tipo** de retorno.
- un **modificador de acceso**: **public** o **private**. (opcional)

Cliente.java

```
package pizzeria;

public class Cliente {
    private int clienteId;
    private String domicilio;
    private String telefono;

    public String getDomicilio() {
        return domicilio;
    }
    public void setDomicilio(String s) {
        domicilio = s;
    }
    // más métodos de instancia
}
```

Tipos de datos en Java

En **java** hay 2 categorías de tipos de datos: tipo primitivo y tipo de una clase (referencia).

◇ Tipos primitivos: las variables de tipo primitivo mantienen valores simples y NO son objetos. Existen 8 tipos de datos primitivos:

Declaración e inicialización de variables primitivas

- Entero: **byte, short, int, long**
- Punto flotante: **float y double**
- Un carácter de texto: **char**
- Lógico: **boolean**

```
float pi = 3.14;  
double saldo = 0;  
char letra = 'A';  
int hora = 12;  
boolean es_am = (hora>12);  
                true/false
```

◇ Tipos de una clase: las variables que referencian a un objeto son llamadas *variables referencias* y contienen la ubicación (dirección de memoria) de objetos en memoria.

Declaración e inicialización de variables referencias

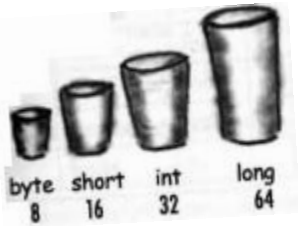
```
Cliente cli;  
cli = new Cliente();  
  
Fecha diaCumple = new Fecha();
```

Tipos de datos en Java

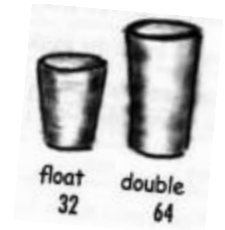
Inicialización

Si la definición de una clase, no inicializa variables de instancia, las mismas toman valores por defecto.

- Las variables de instancia de **tipo primitivo** se inicializan con los siguientes valores por defecto:



Tipo primitivo	Valor por defecto
boolean	false
char	'\u0000' (nulo)
byte/short/int/long	0
float/double	0.0



- Las variables de instancia que son referencias a objetos, se inicializan con el valor por defecto: **null**.



Nota: las variables locales, es decir, las variables declaradas dentro de un método, deben inicializarse explícitamente antes de usarse.

Tipos de datos en Java

Clases wrappers

- Java no considera a los tipos de datos primitivos como objetos. Los datos numéricos, booleanos y de caracteres se tratan en su forma primitiva por razones de eficiencia.
- Java proporciona clases **wrappers** para manipular a los datos primitivos como objetos. Los datos primitivos están envueltos ("*wrapped*") en un objeto que se crea entorno a ellos.
- Cada tipo de datos primitivo de Java, posee una clase *wrapper* correspondiente en el paquete **java.lang**. Cada objeto de la clase *wrapper* encapsula a un único valor primitivo.

Tipo primitivo	Clase Wrapper
char	Character
boolean	Boolean
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Tipos de datos en Java

Clases wrappers

A partir de la versión de Java, **J2SE 5.0**, se dispone de conversiones automáticas de tipos primitivos a objetos y viceversa. Antes de esta versión, se debía convertir explícitamente usando métodos.



¿Qué es Autoboxing?

Es la conversión automática entre objetos de clases wrapper y tipos de datos primitivos.

Boxing es una manera de encapsular un tipo de dato primitivo en un objeto (su wrapper).

```
Integer nro = 3;    (int)
Character letra = 'a'; (char)
```

Versiones anteriores al J2SE 5.0

```
Integer nro = new Integer(3);
Character letra = new Character('a');
```

Unboxing es la conversión automática de objetos (wrapper) a un tipo primitivo.

```
Integer nro = new Integer(3);
int num = nro; (objeto Integer)
```

Versiones anteriores al J2SE 5.0

```
Integer nro = new Integer(3);
int num = Integer.parseInt(nro);
```

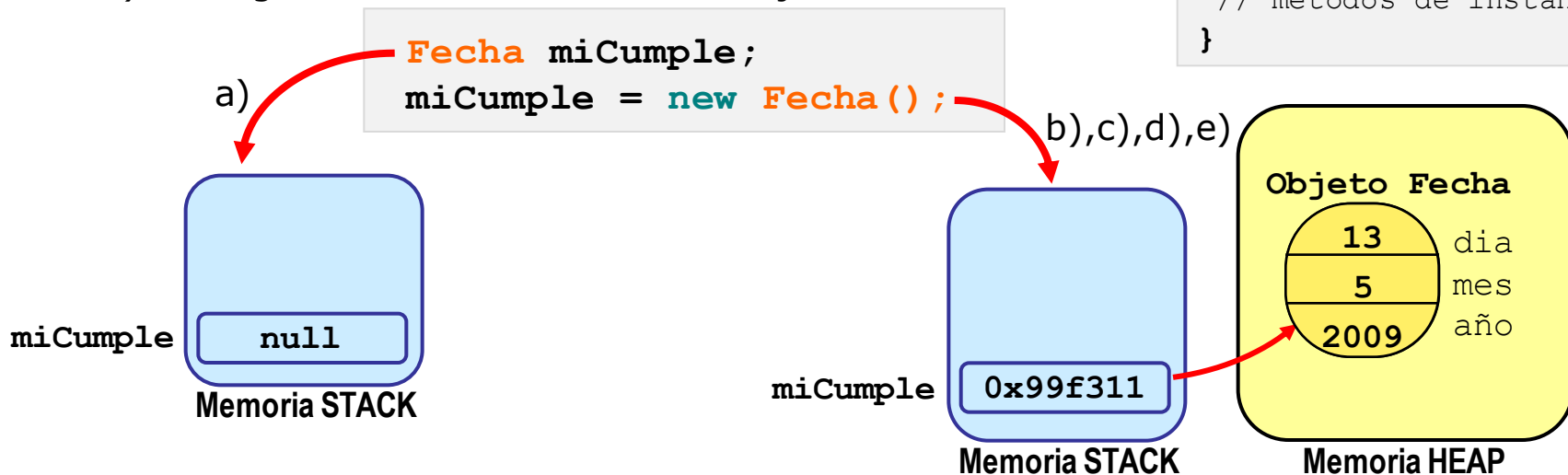


Convierte el Integer que recibe como argumento a int

¿Cómo se instancia una clase?

- ❖ Para instanciar una clase, es decir, para crear un objeto de una clase, se usa el operador **new**.
- ❖ La creación e inicialización de un objeto involucra los siguientes pasos:
 - Se aloca espacio para la variable
 - Se aloca espacio para el objeto en la HEAP y se inicializan los atributos con valores por defecto.
 - Se inicializan explícitamente los atributos del objeto.
 - Se ejecuta el constructor (*parecido* a un método que tienen el mismo nombre de la clase)
 - Se asigna la referencia del nuevo objeto a la variable.

```
public class Fecha {  
    private int dia = 13;  
    private int mes = 5;  
    private int año = 2009;  
    // métodos de instancia  
}
```



¿Cómo manipulo los datos de este objeto de tipo Fecha?

¿Cómo manipular el objeto?

Una vez que se ha creado un objeto, seguramente es para usarlo: cambiar su estado, obtener información o ejecutar alguna acción. Para poder hacerlo se necesita:

- conocer la **variable referencia**
- utilizar el operador **"."**

```
package pizzeria;

public class Cliente {
    private int clienteId;
    private String domicilio;
    private String telefono;

    public int getClienteId(){
        return clienteId;
    }
    public void setClienteId(int c){
        clienteId = c;
    }
    public String getDomicilio(){
        return domicilio;
    }
    public void setDomicilio(String s){
        domicilio = s;
    }
    // más métodos de instancia
}
```

Instanciación de un objeto Cliente e invocación de sus métodos

```
package ayed2011;

public class TestCliente{
    public static void main(String[] args){
        Cliente cli = new Cliente();
        cli.setDomicilio("1 y 60");
        String s = cli.getDomicilio();
    }
}
```

Nota: Se recomienda declarar todos los atributos privados y utilizar métodos públicos para acceder al estado.

Variables y métodos de clase

La palabra clave **static**

La palabra clave **static** declara atributos (variables) y métodos asociados con la clase en vez de con las instancias de la clase.

Variables de Clase

Las variables de clase son **compartidas** por todas las instancias de la clase.

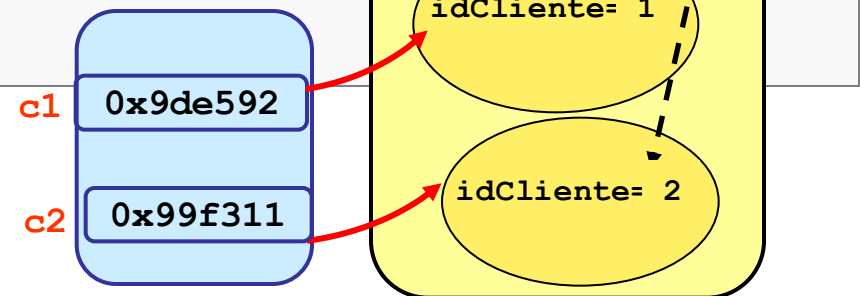
La variable **ultCli** es compartida por todas las instancias de Cliente

```
public class Cliente {  
    private static int ultCli=0;  
    private int idCliente;  
  
    // métodos de instancia  
    public void setIdCliente() {  
        ultCli++;  
        idCliente=ultCli;  
    }  
}
```

Las variables de clase, pueden accederse a través del nombre de la clase. **No es necesario crear instancias!!!**

ultCli es accesible desde cualquier instancia de Cliente

```
public class TestCliente {  
    public static void main(String[] a) {  
        Cliente c1 = new Cliente();  
        Cliente c2 = new Cliente();  
        c1.setIdCliente();  
        c2.setIdCliente();  
    }  
}
```



Variables y métodos de clase

La palabra clave `static`

Métodos de Clase

En algunas situaciones, es necesario ejecutar un método, sin contar con instancias de la clase. Para ello, es necesario declarar **static** los métodos correspondientes a dicho código.

```
public class Cliente {  
    private static int ultCli=0;  
    private int idCliente;  
    // método de clase  
    public static int getUltCli() {  
        return ultCli;  
    }  
    // métodos de instancia  
    public void setIdCliente() {  
        ultCli++;  
        idCliente = ultCli;  
    }  
}
```

No es necesario crear instancias
para usar métodos de clase,
pueden usarse a través del
nombre de la clase

```
public class TestCliente {  
    public static void main(String[] args) {  
        System.out.println("Ultimo = "+Cliente.getUltCli());  
        Cliente c = new Cliente();  
        c.setIdCliente();  
        System.out.println("Ultimo = "+Cliente.getUltCli());  
    }  
}
```

¿Cuál es la salida de TestCuenta?

Ultimo = 0

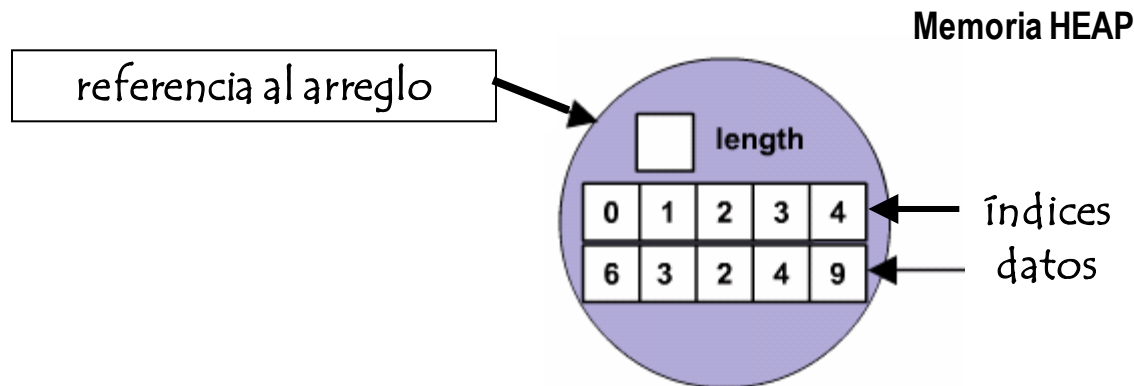
Ultimo = 1

Un método de clase solo tiene acceso a sus variables locales, parámetros y variables de clase.

Arreglos

¿cómo son en java?

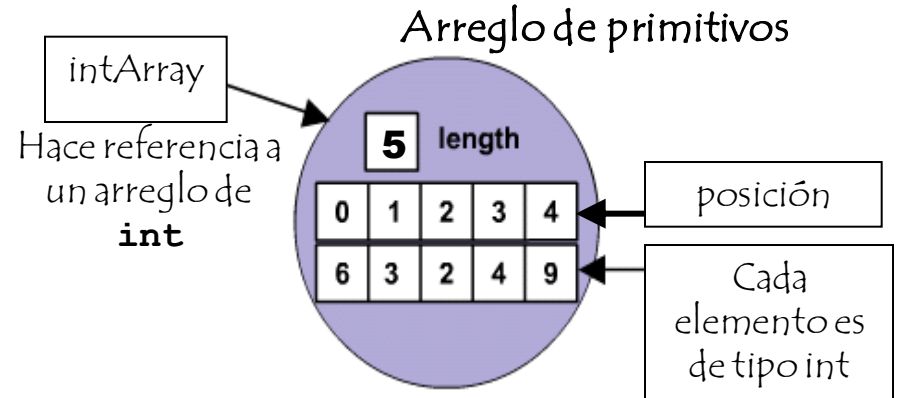
- Un arreglo **es un objeto que hace referencia a un conjunto** de valores primitivos u objetos, a través de una única variable.
- Los arreglos permiten manipular un conjunto de **valores del mismo tipo** de datos usando un único nombre.
- Los datos almacenados en un arreglo, se guardan en **posiciones contiguas**.
- Los arreglos tienen una cantidad fija de objetos o primitivos. Una vez creado la **dimensión no puede cambiar**.
- Cada arreglo mantiene una propiedad **length** con el tamaño del mismo. Al primer elemento del arreglo se le asigna el índice **0**, al siguiente elemento el índice **1**, etc.; por lo tanto, el índice del último elemento del arreglo es **length-1**.



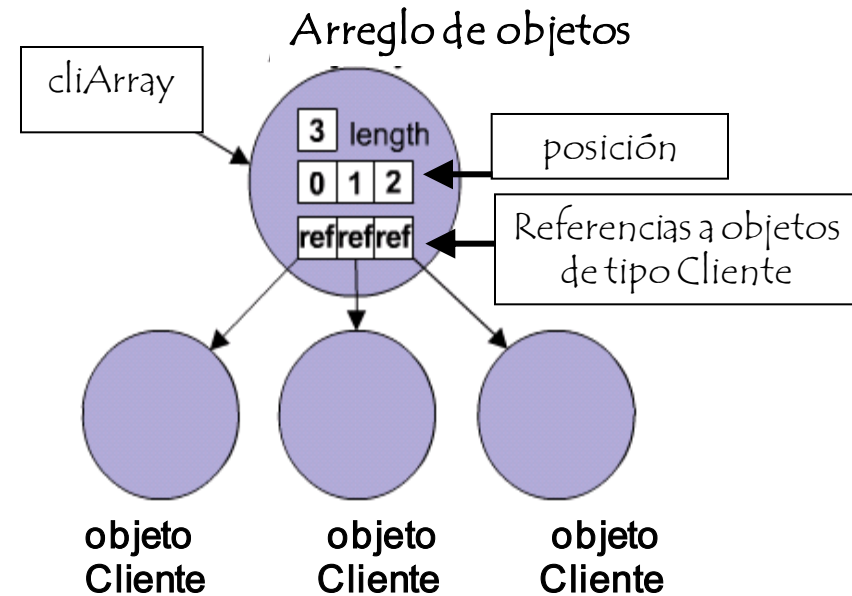
Arreglos

de Tipos Primitivos y de Objetos

```
public class ArreglodePrimitivos {  
    public static void main(String[] args){  
        int[] intArray = new int[5];  
        intArray[0] = 6;  
        intArray[1] = 3;  
        intArray[2] = 2;  
        intArray[3] = 4;  
        intArray[4] = 9;  
    }  
}
```



```
public class ArreglodeClientes {  
    public static void main(String[] args) {  
        Cliente[] cliArray = new Cliente[3];  
        cliArray[0] = new Cliente();  
        cliArray[1] = new Cliente();  
        cliArray[2] = new Cliente();  
    }  
}
```



Arreglos

Declaración e inicialización

La declaración, creación e inicialización de un arreglo pueden hacerse en 1 simple paso o en varios.

Varios Pasos

La declaración crea la variable arreglo, no el objeto arreglo. Para crear el arreglo se usa el operador **new**. Cuando se crea el objeto se debe indicar la cantidad de elementos o longitud del arreglo.

```
int[] intArray;           // sólo declara
intArray = new int[5];    // crea el declarado arriba
double montos[] = new double[12]; // declara y crea
String[] items = new String[4]; // declara y crea
```

Finalmente, se lo inicializa, elemento por elemento. La variable **i** está disponible sólo en el bloque for.

```
for (int i=0; i<4; i++) {
    items[i] = "item"+i;
}
```

Un Paso

La declaración de la variable arreglo, la creación del arreglo y la inicialización, también pueden hacerse en un solo paso. La longitud del arreglo se determina según la cantidad de elementos separados por comas descriptos dentro del bloque { }.

```
int[] intArray = {6, 3, 2, 4, 9};
Cliente[] cliArray = {new Cliente(), new Cliente(), new Cliente()};
String[] items = { "item1", "item2", "item3", "item4"};
```


Arreglos

Recorrido de un arreglo

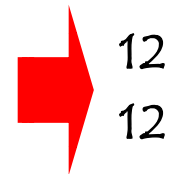
El recorrido de un arreglo puede hacerse de 2 maneras:

```
public class SumaArreglo {  
    public int suma1(int[] a) {  
        int result = 0;  
        for (int i=0; i<a.length;i++)  
            result = result + a[i];  
        return result;  
    }  
  
    public int suma2(int[] a) {  
        int result = 0;  
        for (int i : a)  
            result = result + i;  
        return result;  
    }  
}
```

(1) **For tradicional:** Recorre el arreglo desde la primer posición a la última. El **i** toma el índice actual y **a[i]** es el elemento en esa posición, en cada iteración.

(2) **For mejorado (for-each):** está disponible a partir del j2SE 5.0. Recorre el arreglo desde la primer posición a la última. La línea `for(int i:a)` se lee así: para cada **elemento i de tipo int**, en el arreglo **a**

```
public class Test {  
    public static void main(String[] args) {  
        int[] datos = {2,4,6};  
        SumaArreglo a = new SumaArreglo();  
        System.out.println("La suma es:"+a.suma1(datos));  
        System.out.println("La suma es:"+a.suma2(datos));  
    }  
}
```



Arreglos Multidimensionales

(Matrices)

(2) Usando for-each

```
for( int[] i: notas){
    for( int elto: i)
        System.out.print(elto);
}
```

```
public class Buscador {
    public static void main (String args []){
        int [][] notas =
            {{66,78,78,89,88,90},
            {76,80,80,82,90,90},
            {90,92,87,83,99,94}};

        for (int x = 0; x < notas.length; x ++){
            for(int y = 0; y < notas[x].length; y++) {
                System.out.println(notas[x][y]);
            }
        }
    }
}
```

1º dimensión

2º dimensión

Cant. de filas de la matriz (seria 3)

Cant. de columnas de esa fila (seria 6)

(1) tradicional

66
78
78
89
88
90
76
89

1º fila

salida

[] []	0	1	2	3	4	5
0	66	78	78	89	88	90
1	76	80	80	82	90	90
2	90	92	87	83	99	94

Primera dimensión hace referencia al alumno

notas[2,3] es **83**,
hace referencia al cuarto puntaje del tercer alumno

Segunda dimensión hace referencia a los exámenes