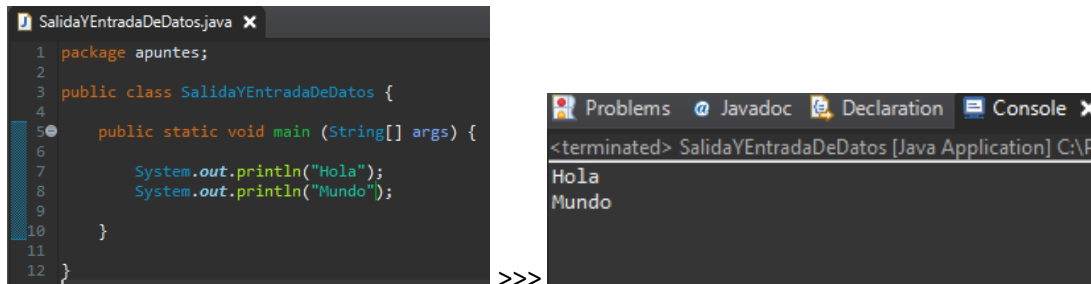


SALIDA DE DATOS

Para la salida de datos vamos a utilizar las sentencias:

`System.out.println();` `System.out.print();`

En los paréntesis pondremos los datos que queremos mostrar en pantalla, la diferencia de que uno tenga *ln* antes de los paréntesis es que si tiene *ln* hará un salto de línea al final del todo:



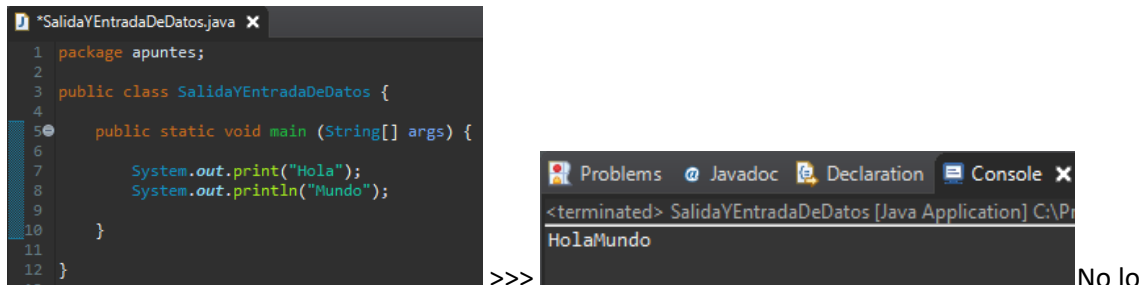
The screenshot shows an IDE with a file named `SalidaYEntradaDeDatos.java`. The code is as follows:

```
1 package apuntes;
2
3 public class SalidaYEntradaDeDatos {
4
5     public static void main (String[] args) {
6
7         System.out.println("Hola");
8         System.out.println("Mundo");
9     }
10 }
11
12 }
```

The console output shows:

```
<terminated> SalidaYEntradaDeDatos [Java Application] C:\P
Hola
Mundo
```

Como veis en la consola hace un salto de línea entre las sentencias, mientras que:



The screenshot shows an IDE with a file named `*SalidaYEntradaDeDatos.java`. The code is as follows:

```
1 package apuntes;
2
3 public class SalidaYEntradaDeDatos {
4
5     public static void main (String[] args) {
6
7         System.out.print("Hola");
8         System.out.println("Mundo");
9     }
10 }
11
12 }
```

The console output shows:

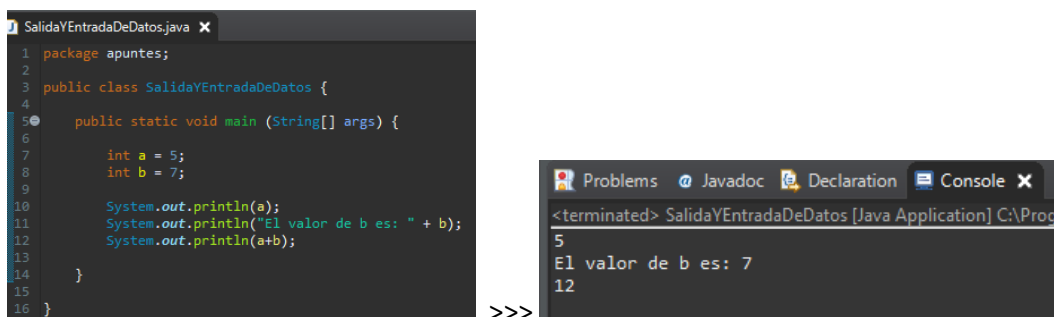
```
<terminated> SalidaYEntradaDeDatos [Java Application] C:\P
HolaMundo
```

No lo

hace, ni siquiera hace un espacio ya que no se lo hemos dicho.

En estas sentencias podemos mostrar “cualquier tipo de dato”, y lo pongo entre comillas ya que los tipos *no primitivos* no pueden ser escritos directamente, aunque los *Strings* y *Doubles* si se puedan por cosas que ya veremos al llegar a “Clases y Objetos”, los *arrays* como ya veremos nos devolverá la dirección de memoria, pero esto ya lo veremos más adelante.

Vamos a imprimir unas variables de diferentes formas.



The screenshot shows an IDE with a file named `SalidaYEntradaDeDatos.java`. The code is as follows:

```
1 package apuntes;
2
3 public class SalidaYEntradaDeDatos {
4
5     public static void main (String[] args) {
6
7         int a = 5;
8         int b = 7;
9
10        System.out.println(a);
11        System.out.println("El valor de b es: " + b);
12        System.out.println(a+b);
13    }
14 }
15
16 }
```

The console output shows:

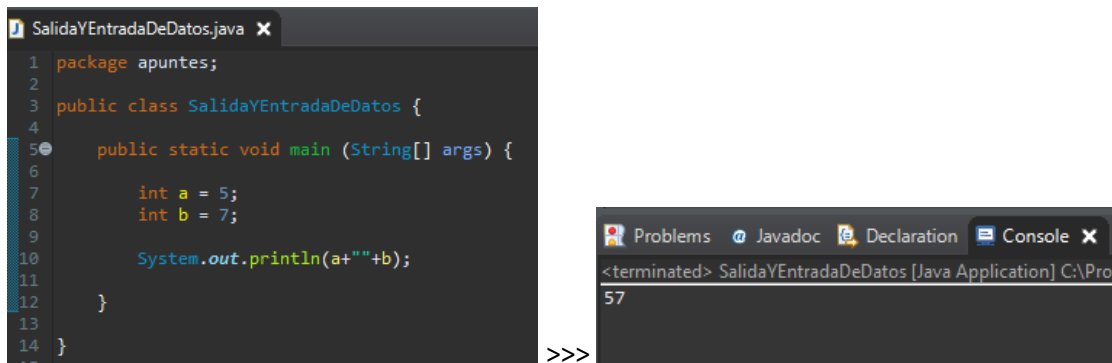
```
<terminated> SalidaYEntradaDeDatos [Java Application] C:\Prog
5
El valor de b es: 7
12
```

Como veis podemos sumar *Strings* a *ints*, y nos devolverá un *String*, o en la línea 12 devolvemos la suma de *a+b*.

Si queremos realizar un salto de línea en algún punto, basta con poner “*\n*” como *String* en algún punto del `System.out`, o que este contenido en un *String*, Ej:

`System.out.println("Hola\nMundo");` -> Esto es una sentencia igual a las dos primeras imágenes de este PDF.

¿Por qué ocurre esto?

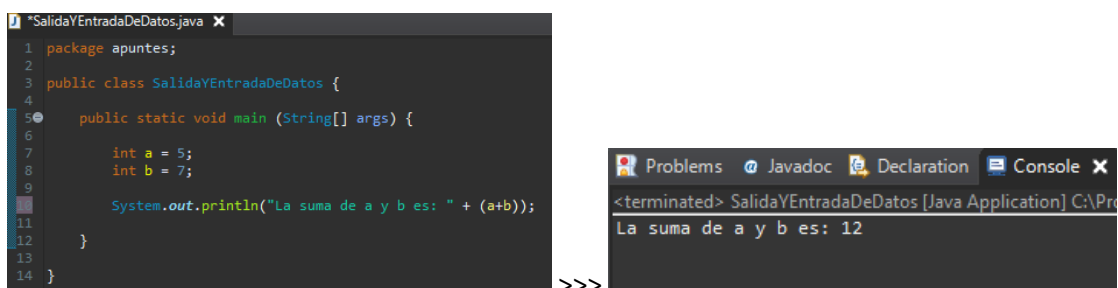


```
1 package apuntes;
2
3 public class SalidaYEntradaDeDatos {
4
5     public static void main (String[] args) {
6
7         int a = 5;
8         int b = 7;
9
10        System.out.println(a+""+b);
11
12    }
13
14 }
```

>>>

```
<terminated> SalidaYEntradaDeDatos [Java Application] C:\Pro
57
```

Esto pasa, por que en la sentencia `(a+""+b)` estamos convirtiendo `a` en un `String`, ya que: `int + String = String`, y luego lo mismo, `String + int = String`, ahora imaginemos que queremos hacer una suma, pero hay un `String` antes:



```
1 package apuntes;
2
3 public class SalidaYEntradaDeDatos {
4
5     public static void main (String[] args) {
6
7         int a = 5;
8         int b = 7;
9
10        System.out.println("La suma de a y b es: " + (a+b));
11
12    }
13
14 }
```

>>>

```
<terminated> SalidaYEntradaDeDatos [Java Application] C:\Pro
La suma de a y b es: 12
```

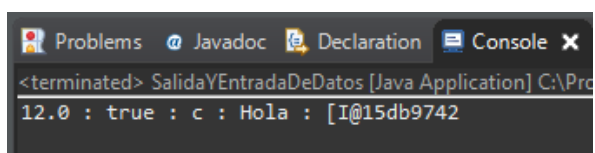
La solución es poner simplemente paréntesis, para que primero haga la operación `int + int`, y luego `String + int`.

Hay que tener cuidado con como sumamos con `Strings`, ya que cualquier cosa sumada a un `String` se convierte en un `String`, y como dije se puede mostrar cualquier tipo *primitivo*:



```
1 package apuntes;
2
3 public class SalidaYEntradaDeDatos {
4
5     public static void main (String[] args) {
6
7         double d1 = 12;
8         boolean verdad = true;
9         char c = 'c';
10
11        String saludo = "Hola";
12        int[] array = {1,2,3};
13
14        System.out.println(d1 + " : " + verdad + " : " + c + " : " + saludo + " : " + array);
15
16    }
17
18 }
```

>>>

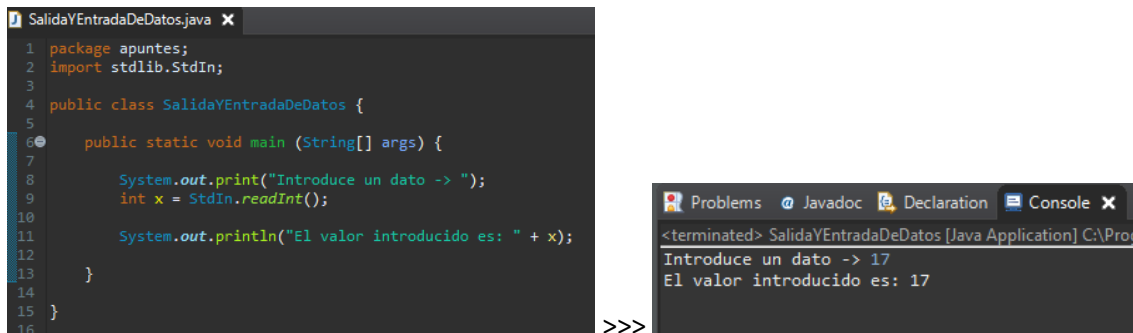


```
<terminated> SalidaYEntradaDeDatos [Java Application] C:\Pro
12.0 : true : c : Hola : [I@15db9742
```

Como veis al tratar de mostrar un `array` en pantalla nos sale la dirección de memoria, mientras que el `String` lo muestra perfectamente, como ya dije esto se verá en “Clases y Objetos”.

ENTRADA DE DATOS

Para introducir datos por teclado vamos a usar la librería *stdlib*, concretamente la clase *StdIn*, para ello vamos a importarla y pondré un ejemplo:




```
1 package apuntes;
2 import stdlib.StdIn;
3
4 public class SalidaYEntradaDeDatos {
5
6     public static void main (String[] args) {
7
8         System.out.print("Introduce un dato -> ");
9         int x = StdIn.readInt();
10
11         System.out.println("El valor introducido es: " + x);
12     }
13 }
14
15 }
16
```

>>>

```
<terminated> SalidaYEntradaDeDatos [Java Application] C:\Pro
Introduce un dato -> 17
El valor introducido es: 17
```

Importo la librería en la línea 2, luego en la línea 8 pido en pantalla que se introduzca un valor, sin *ln* para que se introduzca al lado, luego en la 9, pongo: `int x = StdIn.readInt();`

Es decir, cuando pongo `StdIn.readInt()`, se va a sustituir esa sentencia por el número que yo introduzca, luego en la línea 11 muestro el valor de *x* por pantalla, otro ejemplo:



```
1 package apuntes;
2 import stdlib.StdIn;
3
4 public class SalidaYEntradaDeDatos {
5
6     public static void main (String[] args) {
7
8         int x = StdIn.readInt() + StdIn.readInt();
9
10         System.out.println("El valor introducido es: " + x);
11     }
12 }
13
14 }
15
```

>>>

```
<terminated> SalidaYEntradaDeDatos [Java Application] C:\Pro
5
7
El valor introducido es: 12
```

Introduzco dos datos, y estos se le asignan a *x*, no solo existe `readInt()`, también podemos hacer los siguientes:

- `readInt()` : `int` -> Lee un número entero.
- `readDouble()` : `double` -> Lee un numero con decimales, puede ser tanto *double* como *Double*.
- `readBoolean()` : `boolean` -> Puedes introducir (`true`, `false`, `1`, `0`)
- `readChar()` : `char` -> Puedes introducir una cadena de texto tan larga como quieras, pero solo se quedará con el primer carácter, Ej: "Hola Mundo" -> 'H'
- `readString()` : `String` -> Se quedará con la primera palabra, Ej. "Hola Mundo" -> "Hola"
- `readLine()` : `String` -> Con toda la frase, Ej: "Hola Mundo" -> "Hola Mundo"

Os recomiendo meteros a las clases de la librería *stdlib* y ver que funciones tiene, vienen con una pequeña descripción, pero está en inglés.