

1. ¡Hola mundo! - Salida de datos por pantalla

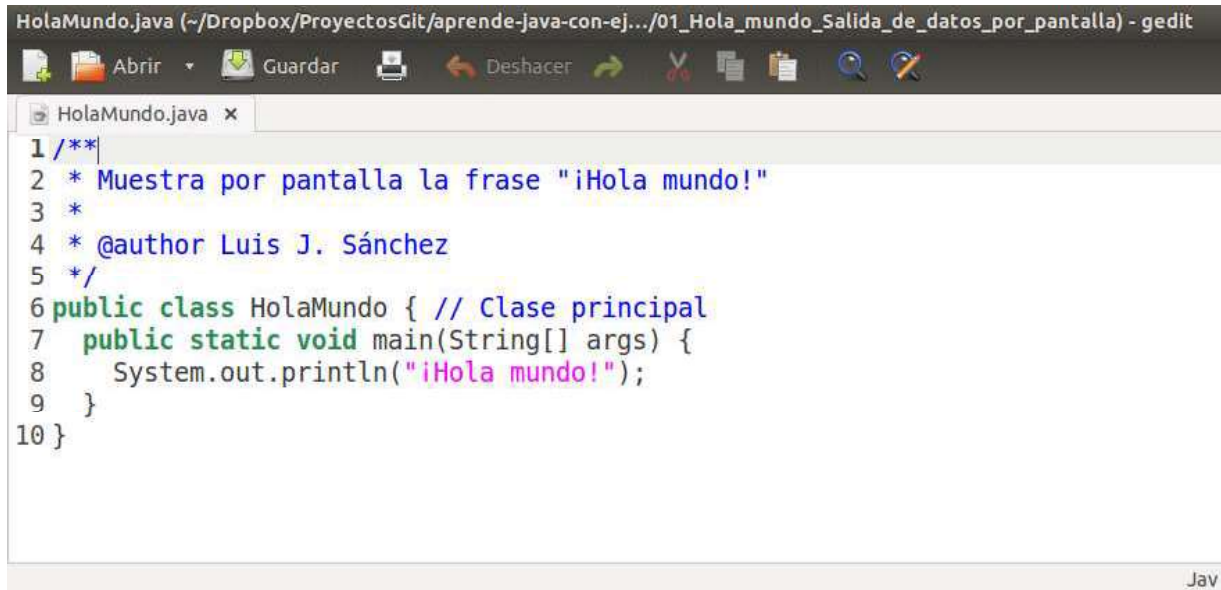
1.1 ¡Hola mundo! - Mi primer programa

El primer programa que aprende a hacer cualquier aspirante a programador es un **Hola mundo**¹. Es seguramente el programa más sencillo que se puede escribir. Se trata de un programa que muestra el mensaje “Hola mundo” por pantalla. Abre el programa **GEdit** que, como hemos comentado en la sección [Instalación y configuración del entorno de programación Java](#), viene instalado por defecto en Ubuntu y teclea el siguiente código:

```
/**
 * Muestra por pantalla la frase "¡Hola mundo!"
 *
 * @author Luis J. Sánchez
 */
public class HolaMundo { // Clase principal
    public static void main(String[] args) {
        System.out.println("¡Hola mundo!");
    }
}
```

Verás que, en principio, el texto del código no aparece en colores. El coloreado de sintaxis es muy útil como comprobarás más adelante, sobre todo para detectar errores. Graba el programa como `HolaMundo.java`. Al grabar con la extensión `.java`, ahora el programa **GEdit** sí sabe que se trata de un programa escrito en Java y reconoce la sintaxis de este lenguaje de forma que puede colorear los distintos elementos que aparecen en el código.

¹En la página <http://helloworldcollection.de/> se muestra el programa **Hola mundo** escrito en más de 400 lenguajes de programación.

A screenshot of the GEdit text editor window. The title bar reads 'HolaMundo.java (~/.Dropbox/ProyectosGit/aprende-java-con-ej.../01_Hola_mundo_Salida_de_datos_por_pantalla) - gedit'. The menu bar includes 'Abrir', 'Guardar', 'Deshacer', and others. The code is as follows:

```
1 /**
2  * Muestra por pantalla la frase "¡Hola mundo!"
3  *
4  * @author Luis J. Sánchez
5  */
6 public class HolaMundo { // Clase principal
7     public static void main(String[] args) {
8         System.out.println("¡Hola mundo!");
9     }
10 }
```

Figura 1.1: Programa `Hola mundo` escrito en el editor GEdit

Fíjate que el nombre que le damos al fichero es exactamente igual que la clase principal seguido de la extensión `.java`. Abre una ventana de terminal, entra en el directorio donde se encuentra el fichero y teclea lo siguiente:

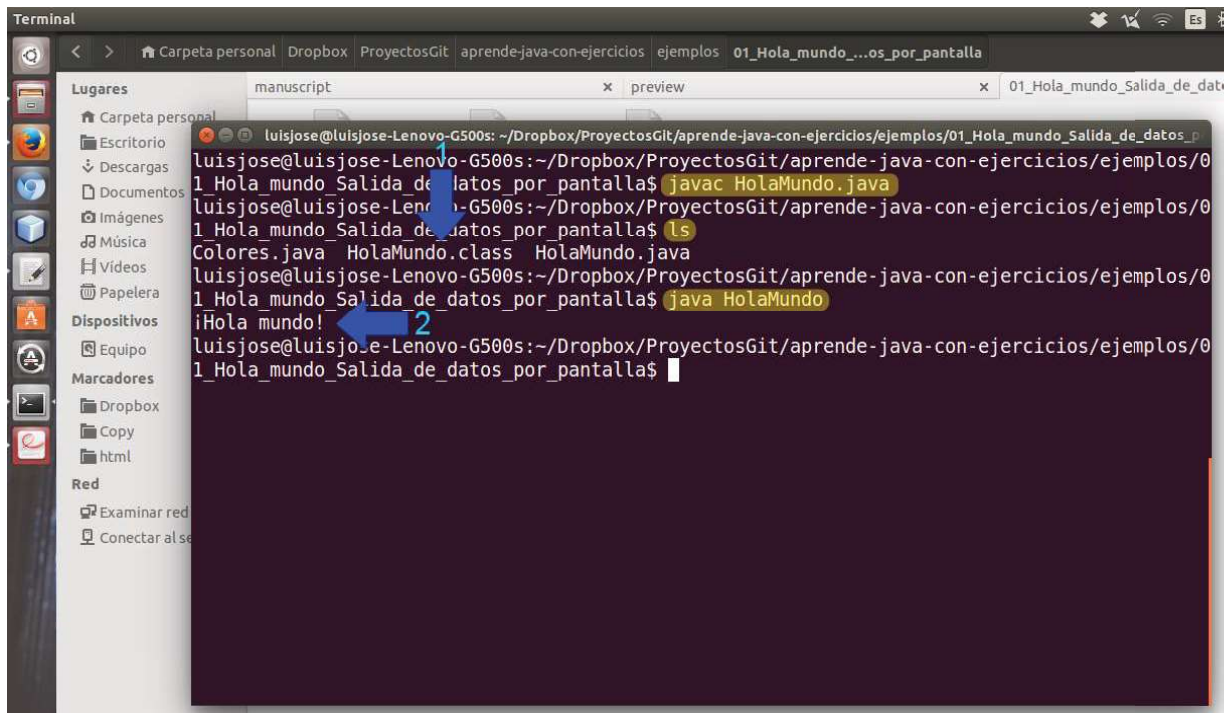
```
javac HolaMundo.java
```

Este comando crea `HolaMundo.class` que es el *bytecode*. Para ejecutar el programa, teclea:

```
java HolaMundo
```

¡Enhorabuena! Acabas de realizar tu primer programa en Java.

En la siguiente captura de pantalla puedes ver la ventana de terminal y los comandos que acabamos de describir.



```
luisjose@luisjose-Lenovo-G500s: ~/Dropbox/ProyectosGit/aprende-java-con-ejercicios/ejemplos/01_Hola_mundo_Salida_de_datos_pantalla$ javac HolaMundo.java
luisjose@luisjose-Lenovo-G500s: ~/Dropbox/ProyectosGit/aprende-java-con-ejercicios/ejemplos/01_Hola_mundo_Salida_de_datos_pantalla$ ls
Colores.java  HolaMundo.class  HolaMundo.java
luisjose@luisjose-Lenovo-G500s: ~/Dropbox/ProyectosGit/aprende-java-con-ejercicios/ejemplos/01_Hola_mundo_Salida_de_datos_pantalla$ java HolaMundo
¡Hola mundo!
```

Figura 1.2: `HolaMundo.class` es el *bytecode* generado (1). El resultado del programa es una línea de texto escrita en pantalla (2).

La instrucción que hemos utilizado para mostrar una frase por pantalla es `System.out.println()`, colocando la frase entre paréntesis. También se puede volcar texto por pantalla mediante `System.out.print()`. La única diferencia radica en que esta última no añade un salto de línea al final, pruébalo en `HolaMundo.java` y compruébalo por ti mismo. Si lo que quieres mostrar es una palabra o una frase, como en este ejemplo, es importante que el texto esté entrecomillado.



- El programa propiamente dicho está dentro de lo que llamamos “clase principal”.
- El fichero debe tener el mismo nombre que la clase principal seguido por la extensión `.java`
- Los comentarios de varias líneas se colocan entre `/* y */`
- Los comentarios de línea se indican mediante `//`
- Para mostrar información por pantalla se utilizan `System.out.println()` y `System.out.print()`.

1.2 Coloreado de texto

Mostrar siempre texto blanco sobre fondo negro puede resultar muy aburrido. El texto que se muestra por pantalla se puede colorear; para ello es necesario insertar unas secuencias de caracteres - que indican el color con el que se quiere escribir - justo antes del propio texto².

Prueba el siguiente programa. Observa que delante de cada palabra hay una secuencia de caracteres. Esa secuencia provoca un cambio de color.

```
/**
 * Coloreado de texto
 *
 * Muestra varias palabras en el color que corresponde.
 *
 * @author Luis José Sánchez
 */
public class Colores {
    public static void main(String[] args) {
        System.out.print("\033[33m mandarina");
        System.out.print("\033[32m hierba");
        System.out.print("\033[31m tomate");
        System.out.print("\033[37m sábanas");
        System.out.print("\033[36m cielo");
        System.out.print("\033[35m nazareno");
        System.out.print("\033[34m mar");
    }
}
```

A continuación se muestra el resultado de la ejecución del programa.

²El coloreado de texto usando este método puede que no funcione en terminales de otros sistemas operativos que no son Linux.



Figura 1.3: Coloreado de texto.

Se ha escrito un espacio entre cada secuencia que establece el color y cada palabra por claridad, pero hay que tener en cuenta que esos espacios se “pintan”, es decir, se colocan huecos. Se podría prescindir del espacio sin ningún problema, por ejemplo escribiendo `System.out.print("\033[31mtomate")` en lugar de `System.out.print("\033[31m tomate")`.

Veamos la gama de colores disponibles.

```
/**
 * Tabla con los colores disponibles
 *
 * Muestra varias palabras en el color que corresponde.
 *
 * @author Luis José Sánchez
 */
public class TablaDeColores {
    public static void main(String[] args) {
        System.out.println(" ");
        System.out.println(" | Código |   Color   | Código |   Color   |");
        System.out.println(" |-----|-----|-----|-----|");
        System.out.print(" | 30 | \033[30m negro \033[39;49m |");
        System.out.println(" | 90 | \033[90m negro claro \033[39;49m |");
        System.out.print(" | 31 | \033[31m rojo \033[39;49m |");
        System.out.println(" | 91 | \033[91m rojo claro \033[39;49m |");
        System.out.print(" | 32 | \033[32m verde \033[39;49m |");
        System.out.println(" | 92 | \033[92m verde claro \033[39;49m |");
        System.out.print(" | 33 | \033[33m amarillo \033[39;49m |");
        System.out.println(" | 93 | \033[93m amarillo claro \033[39;49m |");
        System.out.print(" | 34 | \033[34m azul \033[39;49m |");
        System.out.println(" | 94 | \033[94m azul claro \033[39;49m |");
        System.out.print(" | 35 | \033[35m morado \033[39;49m |");
        System.out.println(" | 95 | \033[95m morado claro \033[39;49m |");
        System.out.print(" | 36 | \033[36m cian \033[39;49m |");
        System.out.println(" | 96 | \033[96m cian claro \033[39;49m |");
        System.out.print(" | 37 | \033[37m blanco \033[39;49m |");
    }
}
```

```

System.out.println("    97    |\033[97m blanco claro \033[39;49m |");
System.out.println("┌──────────┴──────────┴──────────┴──────────┐");
}
}

```

El programa muestra una tabla por consola con los códigos de color disponibles. Usando la plantilla `\033[XXm`, basta cambiar `xx` por código correspondiente al color deseado. Estos códigos están establecidos en el estándar ANSI³.



Código	Color	Código	Color
30	negro	90	negro claro
31	rojo	91	rojo claro
32	verde	92	verde claro
33	amarillo	93	amarillo claro
34	azul	94	azul claro
35	morado	95	morado claro
36	cian	96	cian claro
37	blanco	97	blanco claro

(program exited with code: 0)
Press return to continue

Figura 1.3: Coloreado de texto.

1.3 Sangría

Como puedes ver en todos los ejemplos que hemos mostrado, algunas líneas están ligeramente desplazadas hacia la derecha, es lo que se llama **sangría** o **indentación**. En programación es muy importante sangrar (indentar) bien porque da una idea de qué partes del código son las que contienen a otras. En el último ejemplo tenemos un programa que tiene unos comentarios al principio y luego la clase principal marcada por la línea `public class TablaDeColores {`. Dentro de la clase `TablaDeColores` está el `main` o bloque del programa principal que tiene sangría por estar dentro de la definición de la clase `Colores`. A su vez, dentro del `main` hay una serie de líneas de código que igualmente tienen sangría.

En este libro, tanto el código de los ejemplos como el de las soluciones a los ejercicios siguen el [estándar de Google para el código fuente escrito en el lenguaje de programación Java](https://google.github.io/styleguide/javaguide.html)⁴, por tanto, cada vez que se aplique la sangría será exactamente

³https://en.wikipedia.org/wiki/ANSI_escape_code

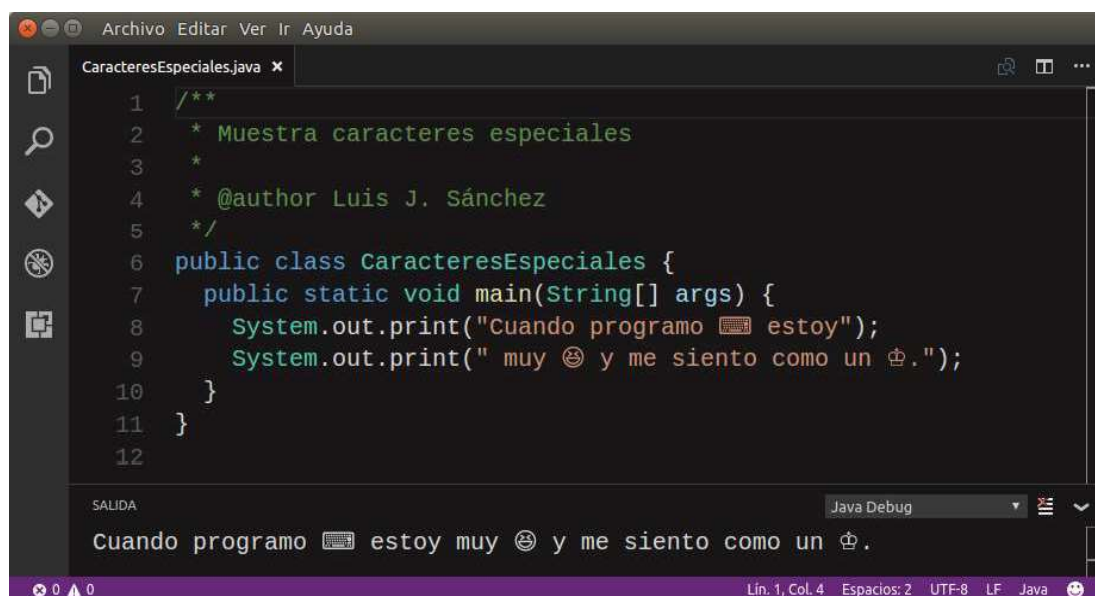
⁴<https://google.github.io/styleguide/javaguide.html>

de **dos espacios**.

1.4 Caracteres especiales

Mediante `System.out.print()` se pueden mostrar palabras o frases, esto es, secuencias de letras, espacios y signos de puntuación como hemos visto. Esta instrucción, además, permite mostrar caracteres especiales - emoticonos, figuras de ajedrez, fichas de dominó, etc. En el [Apéndice C. Caracteres Especiales](#) tienes una selección con muchos de estos caracteres clasificados en diferentes categorías.

Los caracteres especiales se pueden copiar y pegar en el código fuente como se muestra en el siguiente ejemplo⁵.



The screenshot shows an IDE window titled 'CaracteresEspeciales.java'. The code is as follows:

```
1  /**
2   * Muestra caracteres especiales
3   *
4   * @author Luis J. Sánchez
5   */
6  public class CaracteresEspeciales {
7      public static void main(String[] args) {
8          System.out.print("Cuando programo 🎮 estoy");
9          System.out.print(" muy 😊 y me siento como un ♔.");
10     }
11 }
12
```

Below the code editor, the 'SALIDA' (Output) window shows the result of the program: 'Cuando programo 🎮 estoy muy 😊 y me siento como un ♔.'. The status bar at the bottom indicates 'Lín: 1, Col: 4', 'Espacios: 2', 'UTF-8', 'LF', and 'Java'.

Figura 1.4: Caracteres especiales.

En lugar de copiar y pegar, se puede incluir un carácter especial en una cadena de caracteres si se sabe su código en el estándar **Unicode**⁶. Basta con escribir `\u` seguido del código. Por ejemplo, el símbolo de la corchea tiene el código unicode `266A` y el de doble corchea `266A`.

⁵El código de ejemplo funciona en Linux, podría no funcionar en otras plataformas.

⁶Lista de caracteres Unicode: https://en.wikipedia.org/wiki/List_of_Unicode_characters

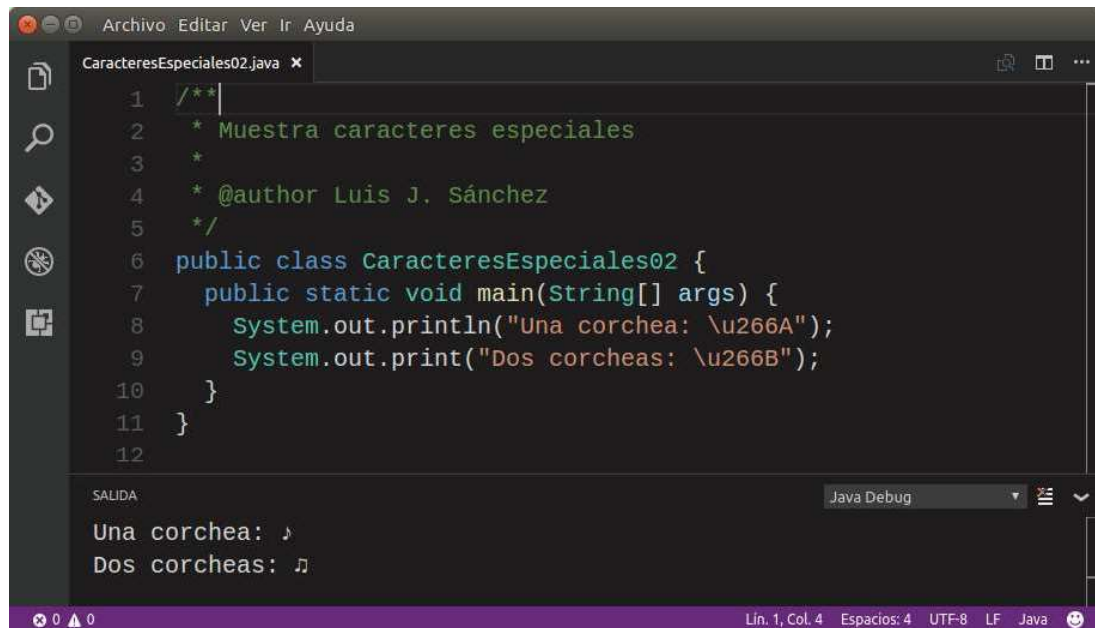


Figura 1.5: Caracteres especiales.

1.5 Salida formateada

Java dispone de la instrucción `System.out.printf()` análoga al `printf()` del lenguaje C. Permite formatear la salida que se pretende mostrar por pantalla.

El ejemplo que se muestra a continuación:

```
/**
 * Salida formateada
 *
 * @author Luis J. Sánchez
 */
public class SalidaFormateada01 {
    public static void main(String[] args) {
        System.out.printf("El número %d no tiene decimales.\n", 21);
        System.out.printf("El número %f sale con decimales.\n", 21.0);
        System.out.printf("El %.3f sale exactamente con 3 decimales.\n", 21.0);
    }
}
```

muestra la siguiente salida:

```
El número 21 no tiene decimales.  
El número 21,000000 sale con decimales.  
El 21,000 sale exactamente con 3 decimales.
```

Veamos en detalle una de las líneas.

```
System.out.printf("El número %d no tiene decimales.\n", 21);
```

Observa que a la instrucción `System.out.printf()` se le pasan dos elementos; por un lado, una plantilla entrecomillada - "El número %d no tiene decimales.\n" - y por otro lado el/los dato/s separado/s por coma. En este caso únicamente se pasa un dato - el número 21 - pero podrían ser varios números o incluso números y palabras. Lo que sucede en esta línea es que el dato - el número 21 - se inserta dentro de la plantilla, justo en la casilla %d. Debe haber tantas casillas con el símbolo % como datos separados por coma. Los datos se insertan siempre por orden en las casillas que le corresponden. En una casilla %d solo se aceptan números enteros (sin decimales).

El carácter '\n' es un salto de línea.

Veamos ahora otra de las líneas del programa.

```
System.out.printf("El %.3f sale exactamente con 3 decimales.\n", 21.0);
```

La casilla %f se debe rellenar con un número que contiene decimales. Si se escribe %f tal cual, se mostrarán varios decimales, aunque no sabemos en principio cuántos. Mediante %.3f se especifica que se deben mostrar exactamente 3 decimales.

La casilla %s dentro de una plantilla que se le pasa a un `System.out.printf()` se debe rellenar con una cadena de caracteres, es decir, con una secuencia de letras, espacios y signos de puntuación.

Veamos un ejemplo en el que se mezcla la salida formateada de cadenas de caracteres, números enteros y números con decimales.

```
/**  
 * Salida formateada  
 *  
 * @author Luis J. Sánchez  
 */  
public class SalidaFormateada02 {  
    public static void main(String[] args) {  
        System.out.println(" Artículo      Precio/caja    Nº cajas");  
        System.out.println("-----");  
        System.out.printf("%-10s      %8.2f      %6d\n", "manzanas", 4.5, 10);  
        System.out.printf("%-10s      %8.2f      %6d\n", "peras", 2.75, 120);  
        System.out.printf("%-10s      %8.2f      %6d\n", "aguacates", 10.0, 6);  
    }  
}
```

En la casilla `%-10s` se inserta una cadena de caracteres. Esta cadena debe ir entrecomillada. Se reservan 10 caracteres; si la cadena es más pequeña, se rellena con espacios en blanco hasta completar las 10 posiciones. El guión indica que la cadena se debe alinear a la izquierda (`%-10s` alinea a la izquierda y `%10s` alinea a la derecha).

Dentro de la casilla `%8.2f` se inserta un número con decimales. Se reservan 8 caracteres para el número completo, incluyendo la coma decimal y se muestran exactamente 2 decimales.

Por último, la casilla `%6d` muestra por pantalla un número entero colocado en un hueco de 6 posiciones. Si el número ocupa menos, se rellena con espacios en blanco.

El resultado de la ejecución del programa es el siguiente:

Artículo	Precio/caja	Nº cajas
manzanas	4,50	10
peras	2,75	120
aguacates	10,00	6

En una cadena formateada se pueden incluir tabuladores con el carácter `\t`, comillas simples con `\'`, comillas dobles con `\"` o incluso la propia barra inclinada con `\\`.