Anexo: III.- Códigos de escape ANSI

Los códigos de escape ANSI se utilizan para dar formato a la salida de una terminal de texto y se basan en un estándar ANSI, ANSI X3.64 (también llamado ECMA-48). Cuando se intenta representar un código de escape, la terminal lo intercepta y actúa en función a este código, **cambiando el color o moviendo el cursor**. Las posibilidades de las secuencias de escape son muy variadas y se utilizan para dar formato y cambiar el aspecto del que se va a mostrar por la terminal.

Todos los códigos de escape empiezan por los caracteres **ESC** (carácter ASCII número 27 decimal, **1B** en hexadecimal) seguido del carácter **[**.

En la siguiente tabla podemos ver algunos de los códigos de escape más importantes

Secuencia	Acción
ESC[nA	Mueve el cursor hacia arriba n filas. Si el cursor se encuentra en la parte superior de la pantalla, no tiene ningún efecto. Si no se especifica n el cursor sube una fila.
ESC[nB	Mueve el cursor n filas hacia abajo. Igual que con el desplazamiento hacia arriba, si el cursor se encuentra en la parte inferior de la pantalla el cursor no se mueve, y si no se especifica n baja una fila.
ESC[nC	Mueve el cursor n columnas hacia la derecha. Si el cursor se encuentra en la última columna no tiene efecto. Si no se especifica n el desplazamiento es de una columna.
ESC[nD	Mueve el cursor n columnas a la izquierda. Si el cursor se encuentra en la última columna no tiene efecto. Si no se especifica n el desplazamiento es de una columna.
ESC[n;mf	Mueve el cursor a la fila n y columna m. Si n no se especifica el cursor se mueve a la primera fila.
ESC[H	Mueve el cursor al lado superior izquierdo de la pantalla.
ESC[nJ	Borra parte de la pantalla. Si n vale **0** se borra desde el cursor hasta el final de la pantalla. En caso de que n valga 1 se borra desde la posición del cursor hasta el principio. Si n vale **2** se borra toda la pantalla
ESC[nK	Borra parte de la línea. Si n es **0**, desde el cursor al final de la línea, en caso de que valga 1 se borra hasta el principio. Si n vale **2** se borra toda la línea.
ESC[s	Guarda la posición actual del cursor.
ESC[u	Coloca el cursor en la posición guardada anteriormente.
ESC[b;fg;bgm	Establece la intensidad (b), el color del primero plano (fg) y el color de fondo (bg) del texto. La existencia de b , fg y bg y el rango de valores que le asignamos puede variar en función del terminal donde se ejecute nuestro programa. Podemos a ir a lo seguro si utilizamos los códigos de 3 bits (8 colores), también podemos utilizar códigos de 8 bits (256 colores), de 16 bits (64K colores) e incluso 24 bits (16M colores) pero dependen de que el terminal donde se ejecute soporte ese número de colores.

La representación de ESC[en unicode sería "`\u001B[`".

Cuando escribimos datos haciendo uso de System.out lo que estamos haciendo en realidad es enviar los

datos a una memoria temporal (buffer) y cuando el sistema lo considera oportuno, las envía todas de golpe hacia el flujo de salida, ya sea la pantalla, un archivo, etc. Este comportamiento se hace por qué es más eficiente escribir varios datos de golpe que no una por una.
Podemos forzar a que los datos enviados al flujo de salida (System.out.*) sean interpretadas inmediatamente mediante la siguiente instrucción:

```
System.out.flush();
```

Ejemplo: para borrar la pantalla y situar el cursor al inicio (lado superior izquierdo):

```
System.out.print("\u001B[H\u001B[2J");
System.out.flush();
```

Códigos de colores

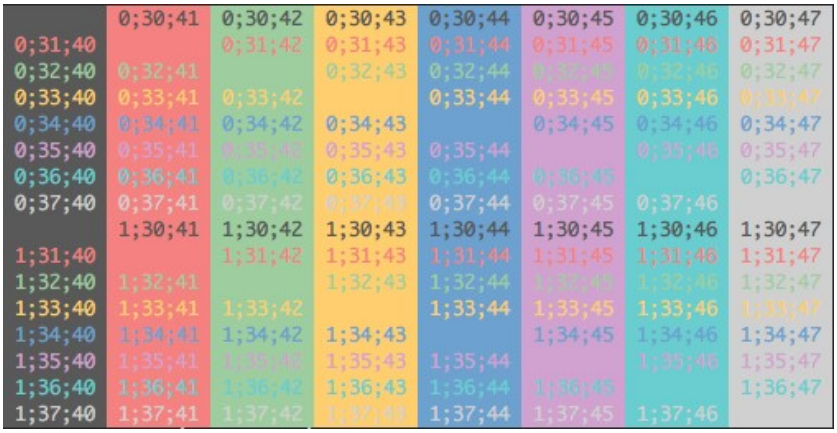
Cómo acabamos de mencionar los códigos de colores que asignamos pueden variar en función del grado de compatibilidad que deseemos que soporte nuestro programa.

Códigos de 3/4 bits

La especificación inicial solo soportaba **8 colores**. La forma de especificar el códigos de colores es la siguiente:

- El primer número opcional (b bold) nos permite especificar la intensidad en la que aparece el texto. Un valor 0 aparecerá normal y un valor 1 aparecerá en negrita. En terminales que lo soportan se pueden poner más valores, por ejemplo 4 sería subrayado.
- El segundo número (fg foreground) nos permite especificar el color del primer plano utilizando números del 30 al 37. El 3 indica color de primer plano y del 0 al 7 son los colores que podemos asignar, negro, rojo, verde, amarillo, azul, magenta, cyan, blanco respectivamente.
- El tercer número (bg background) nos permite especificar el color de fondo utilizando números del 40 al 47. El 4 indica color de fondo y del 0 al 7 son los colores que podemos asignar, negro, rojo, verde, amarillo, azul, magenta, cyan, blanco respectivamente.

En la siguiente tabla podemos ver la combinación de colores que soportaba la especificación inicial:



Ejemplo, para escribir “Hola” en negrita, color de fondo amarillo y texto azul haríamos lo siguiente:

```
System.out.println("\u001B[1;34;43m Hola \u001B[0m");
```

Hay que recalcar que aunque los códigos de colores de la especificación son los que son, la representación que cada terminal hace de ellos puede variar. Es decir, el color amarillo será amarillo en todos los terminales, pero dentro de la gamma de colores amarillos cada terminal hace su propia representación.

Códigos de 8 bits

Al poco de tiempo de hacerse la especificación inicial era evidente que con 8 colores las aplicaciones tenían muchas limitaciones. Por ello la ITU (International Telecommunication Union) formalizó una nueva especificación que diera cabida a la antigua representación para mantener la compatibilidad, pero que en hardware y software más nuevo permitiera representar 256 colores.

La forma de representar colores en esta nueva especificación difiere del anterior:

- El primer número opcional (b bold) nos permite especificar la intensidad en la que aparece el texto. Un valor 0 aparecerá normal y un valor 1 aparecerá en negrita. En terminales que lo soportan se pueden posar más valores, por ejemplo 4 sería subrayado.
- Los siguientes 3 números (segundo, tercero y cuarto) nos permiten especificar el color. El segundo número según el valor que tenga nos permite diferenciar entre color de primer plano o color de fondo.
  - Si es 38 color de primer plano
  - Si es 48 color de fondo

  El tercer número siempre será el valor 5

  El cuarto número representará el color de 8 bits (un valor de 0 a 255).

  Por ejemplo para escribir “Hola” en negrita y color de primero plano rojo haríamos:

  ```
  System.out.print("\u001B[1;38;5;9m Hola \u001B[0m");
  ```

  Si queremos especificar además un color de fondo, se tiene que añadir otro carácter de escape, per ejemplo para escribir “Hola” en negrita, color de primer plano blanco y color de fondo azul haríamos:

  ```
  System.out.print("\u001B[1;38;5;15m\u001B[1;48;5;4m Hola \u001B[0m");
  ```

Códigos de 24 bits

En 1995 la ISO (International Organization for Standarization) y la IEC (International Electrotechnical Commission) con colaboración con la ITU (International Telecommunication Union) desarrollaron una nueva especificación, ISO-8613-3 que permitiera representar 16 millones de colores en las terminales.
Si bien todo el hardware actual soporta sin ningún problema esta especificación, no podemos decir el mismo del software. La mayor parte de los terminales de sistemas Linux y Unix si apoyan a color de 24 bits pero en el SO Windows no, por lo tanto su uso es reducido.
Aunque en los ejercicios no vayamos a hacer uso de códigos de 24 bits, se muestra como los

representaríamos en un terminal:

```
ESC[ … 38;2;r;g;bm  Para representar un color de primer plano RGB (Red, Green, Blue)


ESC[ … 48;2;r;g;bm  Para representar un color de fondo RGB (Red, Green, Blue)
```

Dónde:

- r representa la cantidad de rojo [0-255]
- g representa la cantidad de verde [0-255]
- b representa la cantidad de azul [0-255]