Trabajo optativo para navidad – Juegos

Los nombres de los títulos son los mismos que los comentados en el código, para que sea más legible el pdf.

1) Inicialización de variables y sus usos durante el programa.

- Array para el cartón del jugador: `int[][] cartonesBingo = new int[3][5];`
- Array para los números que han salido y tiene el jugador:
  `boolean[][] numsMarcados = new boolean[3][5];`
- Array con 90 posiciones, ya que en el bingo se sacan 90 números en total:
  `int[] numsBombo = new int[90];`
- Booleano que comprueba si el número que va a salir ya había salido anteriormente:
  `boolean[] numsSacados = new boolean[90];`
- Importé Random para poder generar los números de forma aleatoria:
  `Random random = new Random();`
- Booleano que por defecto estará en falso hasta que se de el caso contrario: `boolean ganador = false;`

2) Bucle para generar los números del bombo.

Este bucle generará los números del bombo, que como dije anteriormente, irán del 1 al 90. Por cada iteración del bucle, i será el número asignado para cada posición.

Le sumamos 1 para que empiece desde el 1 en vez de desde el 0, ya que en el bingo no hay 0.

```
for (int i = 0; i < 90; i++) {
    numsBombo[i] = i + 1;
}
```

3) Se mezclan los números del bombo.

Para este apartado considero que es mejor explicarlo línea por línea para que se entienda bien que hace cada cosa.

El bucle lo que hace principalmente es recorrer el array a la inversa, es decir, desde el final hasta el principio.

Inicializo la variable i con la posición del último número del array, es decir, como son 90, se inicializaría en 89, ya que en Java comenzamos en 0, no en 1.

El bucle seguirá siempre que i sea mayor a 0, es decir, que el bucle se detendrá cuando llegue a 1.

Al contrario de lo que solemos hacer en clase, por cada iteración disminuimos i en 1, en vez de aumentarlo, lo que hace movernos de atrás hacia delante como expliqué al inicio:

```
for (int i = numsBombo.length - 1; i > 0; i--) {
```

Lo siguiente sería crear una variable nueva que lo que hará será generar en cada iteración un número aleatorio, que será j, que se encuentre entre 0 e i. Al igual que antes, le sumo 1 para que no esté el 0.

```
int j = random.nextInt(i + 1);
```

Después, cree otra variable con el objetivo de guardar de forma temporal el valor que tenemos en la posición i.

```
int nums = numsBombo[i];
```

En la siguiente línea coloco el número de la posición j en la posición i.

```
numsBombo[i] = numsBombo[j];
```

Por último, he colocado el número original de la posición i, que se encontraba guardado de forma temporal en nums, en la posición j.

```
numsBombo[j] = nums;
```

El código final se vería de la siguiente manera:

```
for (int i = numsBombo.length - 1; i > 0; i--) {
    int j = random.nextInt(i + 1);
    int nums = numsBombo[i];
    numsBombo[i] = numsBombo[j];
    numsBombo[j] = nums;
}
```

4) Generar el cartón del usuario.

Una vez que ya tenía los arrays para tener los 90 números, y el bucle que me ayudaría a mezclarlos durante las partidas, lo siguiente era darle un cartón al usuario:

Creo el array usados que me permitirá hacer que el programa lleve un registro de los números que ya se han usado en el cartón:

```
int[] usados = new int[90];
```

Los siguientes bucles lo que harán será recorrer las filas, que son 3, seguido de las columnas, que son 5:

```
for (int fila = 0; fila < 3; fila++) {
    for (int col = 0; col < 5; col++) {
```

Genero otra variable, llamada numero, que guardará el número aleatorio que se genere. Será el siguiente bucle el que generé los números aleatorios que he dicho. En resumen, el bucle se ejecutará hasta que encuentre un número que no haya sido usado anteriormente. La línea de while es la que verifica si el número ya se había utilizado.

```
int numero;
do {
    numero = random.nextInt(90) + 1;
} while (usados[numero - 1] == 1);
```

Cuando se encuentra un número que no se haya utilizado, se marcará como usado en el array usados (nombres repetitivos, pero así me aclaro):

```
usados[numero - 1] = 1;
```

Ahora, se asignaría el número que haya salido, en la posición que le corresponda dentro del array cartonesBingo:

```
cartonesBingo[fila][col] = numero;
```

Por último, el array numsMarcados se inicializa en falso, es decir, como si no estuviera marcado ese número hasta que se de el caso contrario:

```
numsMarcados[fila][col] = false;
```

El código en conjunto se vería así:

```
int[] usados = new int[90];

for (int fila = 0; fila < 3; fila++) {
    for (int col = 0; col < 5; col++) {
        int numero;
        do {
            numero = random.nextInt(90) + 1;
        } while (usados[numero - 1] == 1);

        usados[numero - 1] = 1;
        cartonesBingo[fila][col] = numero;
        numsMarcados[fila][col] = false;
    }
}
```

5) Mostrar al usuario su cartón.

La parte principal la hice para que no fuera tan soso el Bingo y personalizarlo para que usará el nombre que el jugador le proporcionará durante el juego:

```
System.out.println("Bienvenido al juego del BINGO");
Pantalla.escribirSaltoLinea();
String nombre = Teclado.leerString("Escribe tu nombre, jugador: ");
Pantalla.escribirSaltoLinea();
System.out.println(nombre + ", tu cartón es el siguiente: ");
Pantalla.escribirSaltoLinea();
```

La siguiente parte la busqué, ya que, aunque no fuera necesaria le quería dar un toque especial. Ya habíamos visto la estructura de control de excepciones try/catch en clase (o para errores), que por lo que entendí, maneja las excepciones que se puedan dar durante la ejecución de tu código, concretamente dentro del bloque try.

Por otra parte, thread es una clase en java que representa un hilo de ejecución y sleep es un método estático que pertenece a la clase thread, lo que permite pausar el código durante un tiempo en específico. En mi caso puse

1*1000, que sería 1 segundo, aunque para ser sincera, probe varías veces los números hasta que di con un tiempo que no fuera excesivamente largo, pero que fuera suficiente para darle esa pequeña pausa dramática que estaba buscando.

En caso de que ocurriera una excepción durante la ejecución de thread, el programa saltaría a catch, que capturará cualquier tipo de excepción que pueda haber (excepcion e) y acto seguido se lo mostrará por pantalla al usuario:

```
try {
    Thread.sleep(1*1000);
  } catch (Exception e) {
       System.out.println(e);
}
```

Ambos bucles se ejecutarán hasta que se recorran las 3 filas y 5 columnas, que imprimirá cada número seguido de "\t", que es sencillamente tabular, para dar un espacio mayor entre números y que se puedan ver con más claridad. Cada vez que se complete la fila imprimirá una línea nueva para que quede debajo de la anterior:

```
for (int fila = 0; fila < 3; fila++) {
    for (int col = 0; col < 5; col++) {
        System.out.print(cartonesBingo[fila][col] + "\t");
    }
    System.out.println();
}
```

Por último, quise añadir estas dos líneas cuyas únicas funciones son hacerlo más bonito visualmente:

```
System.out.println("-----------------------------------");
Pantalla.escribirSaltoLinea();
```

En conjunto, el código se vería así:

```
System.out.println("Bienvenido al juego del BINGO");
Pantalla.escribirSaltoLinea();
String nombre = Teclado.leerString("Escribe tu nombre, jugador: ");
Pantalla.escribirSaltoLinea();
System.out.println(nombre + ", tu cartón es el siguiente: ");
Pantalla.escribirSaltoLinea();

try {
    Thread.sleep(1*1000);
  } catch (Exception e) {
       System.out.println(e);
}

for (int fila = 0; fila < 3; fila++) {
    for (int col = 0; col < 5; col++) {
        System.out.print(cartonesBingo[fila][col] + "\t");
    }
    System.out.println();
}

System.out.println("-----------------------------------");
Pantalla.escribirSaltoLinea();
```

6) Sacar los números y marcarlos en el cartón.

La variable indiceBombo sirve para llevar la cuenta de los números que se han sacado, empezando desde el índice, que es 0.

El bucle while continuará ejecutándose siempre que se cumplan las condiciones que se ven entre paréntesis, es decir, continuará mientras que no haya un ganador O mientras que no se hayan sacado todos los números.

La variable numeroSacado obtiene el siguiente número a salir usando el índice actual (la variable que se inicializa al principio).

La siguiente línea hace que el booleano numsSacados, que se inicializó al principio como falso, se ponga en verdadero cuando un número haya salido, de esta manera no podrá volver a salir, ya que no puede haber repeticiones en el Bingo. Por otra parte, se le resta 1 a numeroSacado porque se usa como índice, y aunque en el juego haya que empezar con el 1, en java se debe empezar por el 0.

Por último, se incrementa indiceBombo en uno en cada bucle hasta que este llegue a 90, o hasta que se cante Bingo:

```
int indiceBombo = 0;
while (!ganador || indiceBombo < 90) {
    int numeroSacado = numsBombo[indiceBombo];
    numsSacados[numeroSacado - 1] = true;
    indiceBombo++;

    System.out.println("El número es... ");
    Pantalla.escribirSaltoLinea();
```

Las siguientes líneas las añadí un poco por estética y por darle algo más de vida al juego, ya que inicialmente sin los tiempos de pausa, te mostraba el nuevo número y tu cartón ya marcado de golpe, lo cual creo que le quitaba un poco de gracia al juego:

```
System.out.println("El número es... ");
Pantalla.escribirSaltoLinea();

try {
    Thread.sleep(2*1000);
   } catch (Exception e) {
        System.out.println(e);
 }

System.out.println("El " + numeroSacado);
Pantalla.escribirSaltoLinea();

try {
    Thread.sleep(1*1000);
   } catch (Exception e) {
       System.out.println(e);
 }
```

7) Marcar el número en el cartón.

Estos bucles anidados, al igual que anteriormente, recorren los números de cada fila y cada columna, lo que hace que el if pueda comparar el número que se acaba de sacar, con cada número que haya en el cartón.

En caso de que sea igual el número, cambiará a verdadero el booleano numsMarcados.

```
for (int fila = 0; fila < 3; fila++) {
    for (int col = 0; col < 5; col++) {

        if (cartonesBingo[fila][col] == numeroSacado) {
            numsMarcados[fila][col] = true;
        }
```

8) Verificar si hay línea o Bingo.

Primero inicializamos estos dos booleanos. El primero se inicializa en falso ya que lo que quiero es que el programa busque una línea completa y que, si lo hace, se cambie a verdadero.

Por otro lado, inicialicé en verdadero bingo, ya que es más fácil o rápido de esta manera detectar cuando no hay bingo. Con que solo encuentre un número sin marcar el booleano cambiará a falso.

```
boolean linea = false;
boolean bingo = true;
```

El siguiente bucle es el que verifica si hay una línea entera marcada, por lo que recorre las 3 filas y si encuentra algún número no marcado, considerará la fila como incompleta, cambiando así el booleano iniciado dentro del bucle como verdadero, a falso.

```
for (int fila = 0; fila < 3; fila++) {

    boolean filaCompleta = true;

    for (int col = 0; col < 5; col++) {
        if (!numsMarcados[fila][col]) {
            filaCompleta = false;
            break;
        }
```

Sin embargo, si al recorrerla está entera marcada, el booleano filaCompleta se quedará en verdadero, lo que dará paso al siguiente if, que hará que el booleano linea cambiará a verdadero.

```
    if (filaCompleta) {
        linea = true;
        break;
    }
```

Ahora verificamos si hay bingo. Como expliqué antes, es un proceso sencillo ya que una vez detecte un número sin marcar se cambiará a falso. Ese es el funcionamiento de este bucle, recorre cada fila y columna hasta encontrar un número sin marcar, si lo encuentra el booleano bingo cambia a falso.

```
for (int fila = 0; fila < 3; fila++) {
    for (int col = 0; col < 5; col++) {
        if (!numsMarcados[fila][col]) {
            bingo = false;
            break;
        }
```

Por último, se comunica al jugador si ha hecho línea o ha ganado el Bingo. En caso de ganar el Bingo, el booleano ganador se cambiará a verdadero y finalizará el juego:

```
if (bingo) {
    Pantalla.escribirSaltoLinea();
    System.out.println("¡Bingo! "+ nombre + " has ganado.");
    Pantalla.escribirSaltoLinea();
    ganador = true;

} else if (linea) {
    System.out.println("¡Línea! " + nombre + " ha hecho línea.");
    Pantalla.escribirSaltoLinea();
}
```

9) Imprimir el cartón actualizado.

Cada vez que salga el cartón, indicará el nombre del jugador, dando a entender que es su cartón (tampoco hay más jugadores, así que no tiene mucha perdida, pero me parecía más bonito así).

Se repite una vez más el bucle para recorrer el cartón, solo que esta vez tiene más complejidad. Esta parte no se ha visto en clase, ya que es un operador ternario, pero me pareció una forma curiosa y efectiva de hacer una especie de ifs anidados, pero en una sola línea, ya que, dependiendo del resultado, se dará una condición u otra.

Para explicarme mejor, primero imprime el número que le corresponde a esa fila y columna, esta parte no cambia, pero a continuación lo concatena con el booleano numsMarcados seguido de una interrogación (el operador ternario). En caso de que el booleano sea verdadero, el número se tachará (es decir, los dos asteriscos que puse, que ha sido la forma más visual que encontré de que se viera que estaba marcado) y si es falso agrega un espacio, es decir, se queda sin nada, sin tachar.

Toda esa línea se concatena con la tabulación para dar espacio entre números y más abajo se imprime una línea para la separación de las filas (aunque esto ya lo expliqué más arriba):

```
System.out.println("Cartón de " + nombre + ": ");
Pantalla.escribirSaltoLinea();


for (int fila = 0; fila < 3; fila++) {
    for (int col = 0; col < 5; col++) {
        System.out.print(cartonesBingo[fila][col] + (numsMarcados[fila][col] ? "**" : " ") + "\t" );
    }
    System.out.println();

}
```

De nuevo, creo la separación con guiones para que quede mejor visualmente.

El String que he creado es únicamente para que sea el jugador el que controle el ritmo del juego o que le dé la opción de abandonarlo. Inicialmente lo tenía puesto con tiempo, pero consideraba que podía llegar a resultar pesado, de este modo creo que al jugador se le da por lo menos la sensación de participar en cada ronda, aunque únicamente sea poniendo un sí o un no.

```
System.out.println("-----------------------------------");
Pantalla.escribirSaltoLinea();

String opcion = Teclado.LeerString(nombre + " ¿te atreves a seguir jugando? (SI/NO) ");
```

Si el jugador elige que no quiere seguir el juego, entonces le mostrará el siguiente mensaje y finalizará:

```
if (opcion.equals("NO")) {
    Pantalla.escribirSaltoLinea();
    System.out.println(nombre + " ha finalizado el juego. Gracias por jugar.");
    Pantalla.escribirSaltoLinea();
    break;
}
Pantalla.escribirSaltoLinea();
```

Elementos que me hubiera gustado añadir:

Principalmente me hubiera gustado poder añadir la opción de que haya más de un jugador, pero al final me ha llevado mucho más tiempo del que pensé que me llevaría hacer solo el bingo. Igualmente me gustaría en un futuro retomar el juego y volver a intentarlo.

Otro punto que me hubiera gustado añadir era hacer un switch en otra clase y llamarlo con una función para que cada vez que saliera un número, saliera con su frase correspondiente que se usa en el bingo (como el 90, que es el abuelo). Al igual que en el punto anterior, lo estuve intentando, pero al final fui incapaz.

Por otra parte, los cartones de bingo tienen un orden en cada columna que no supe cómo poner y, además, no me gustó del todo como quedaban los números una vez marcados, pero tampoco se me ocurrió como hacerlo de otra manera, ya que probé poniendo una X junto al número, pero tampoco quedaba del todo bien.

Un fallo que he visto que tiene el programa y que no supe arreglar, es que una vez que canta línea, no se borra esa línea en el resto del juego. No es algo que influya, pero visualmente no queda bien.

Bibliografía

Estos son los sitios que usé para orientarme en los apartados donde me quedé atascada:

https://es.stackoverflow.com/questions/296944/como-desordenar-mezclar-barajar-un-array-en-java

https://codegym.cc/es/groups/posts/es.659.metodo-thread-sleep-en-java

https://www.w3schools.com/js/js_strings.asp

https://www.tokioschool.com/noticias/operador-ternario-java/