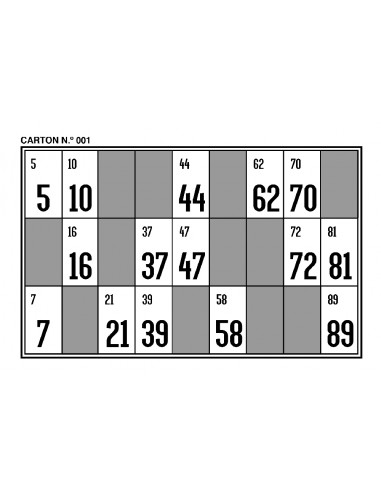
Bingo

Per Xavi Garcia. DAM-Vi A

# Introducció.

Vaig decidir fer els cartrons dels jugadors com els del Bingo normal, com els de la següent imatge:



Els cartrons de Bingo normals tenen 9 columnes. Aquestes 9 columnes es divideixen en 3 caselles y a cada columna se l’asigna una desena. La primera columna té números del 1 al 10, la segona del 11 al 20, la tercera del 21 al 30 i així successivament fins a arribar al 90, que es el número máxim.

A part de la numeració, el cartró normal del bingo te només 15 caselles amb números i 12 buides (tapadas d’ara en endavant). Cada columna només té 1 o 2 caselles tapadas, el que fa que s’assegura que cada columna té com a mínim un número. Tenint això en compte també les files sumen un total de 5 números per fila.

A rel d’aquesta idea van sorgir diferents reptes. Faré una secció per cada un amb un índex per millor accessibilitat.

# Index

1. El disseny del cartró.
2. Tapar les caselles.
3. Omplir caselles amb números.
4. Funcions
   1. isBingo
   2. isFila
   3. inicializar
   4. avanzarPartida.
   5. encontrarNumero.
   6. numerarCasillas
   7. llenarPartida.
   8. llenarVacias
   9. vaciarCasillas
   10. contarX
   11. imprimirTablero
   12. llenarTablero

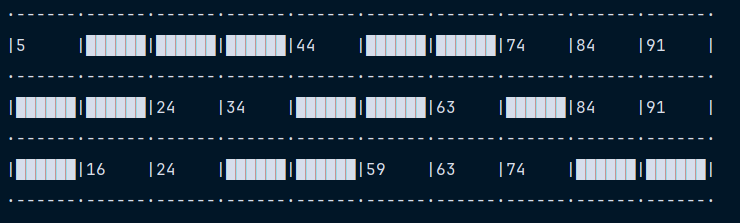
# Disseny del cartó.

He fet un cartó amb un disseny de caselles emmarcades. Amb un símbol "|" pels laterals, un "-" per amunt i avall i a les cantonades, per no deixar-ho en blanc o amb un guió, he utilitzat un "·" perquè faci la mateixa alçada que els guions i la mateixa amplada que els "|".

Codi:



Sortida:



*Els espais entre files els fa el IDE. No existeix una fila entre mig. amb la resta del codi es veurá.*

# Tapar Caselles.

Files i columnes.

Per tal de fer un cartó realista, he hagut de crear una funció anomenada "vaciarCasillas" amb la finalitat de fer-ho coherent. Cada columna té com a molt 2 caselles tatxades. A més, cada fila té 5 caselles buides i 5 caselles amb números.

Cada columna té associada una desena, de forma ascendent. És a dir: la primera columna conté els nombres de l'1 al 10, la segona columna els té del 21 al 30, la tercera del 31 al 40 i així fins al final del cartó.

Primer amb la funció de "vaciarCasillas" he creat un "Random" (rand d'ara endavant) entre 1 i 2. En cas que el "rand" en surti 1, la casella estarà tapada i no podran haver-hi números. Al final de la primera fila comprovo que hi hagi 5 caselles tapades en total. En cas negatiu entra en un bucle que utilitza un rand que contengui el nombre de caselles disponibles. En cas de surtis una casella que ja està tapada es repetirà el procés fins que surti una que no ho estigui i sumo al comptador. Així fins que arriba a 5.

La segona fila fa exactament el mateix codi.

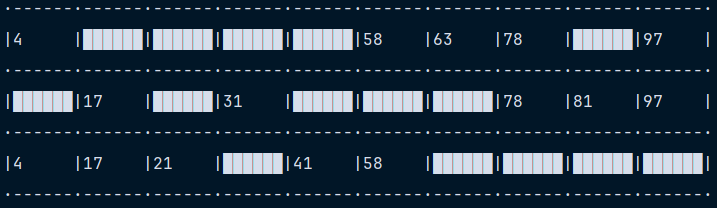
La tercera és la diferent, perquè depèn de les altres dues per generar les caselles buides. Haig d'utilitzar una funció anomenada "contarX" per comptar columna per columna quantes caselles tatxades té. Si la columna en té 1 casella buida, s'usarà un "rand" d'1 o 2 per saber si es tatxa també. En cas d'1, es tatxa, en cas de 2 no.

Al final de l'algoritme, si hi ha menys de 5 caselles tatxades, es fa un altre bucle igual que els anteriors però, a més, he ficat a la condició la funció "contarX" perquè tingui en compte:

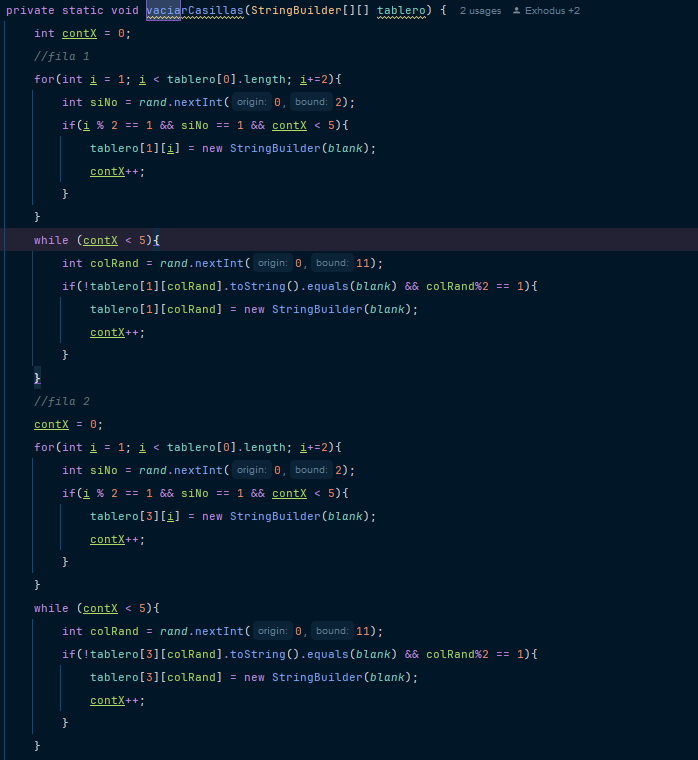
* Si la casella es buida
* Si hi ha menys de 2 caselles tatxades a la mateixa columna
* Si hi ha menys de 5 caselles tatxades a la mateixa fila.

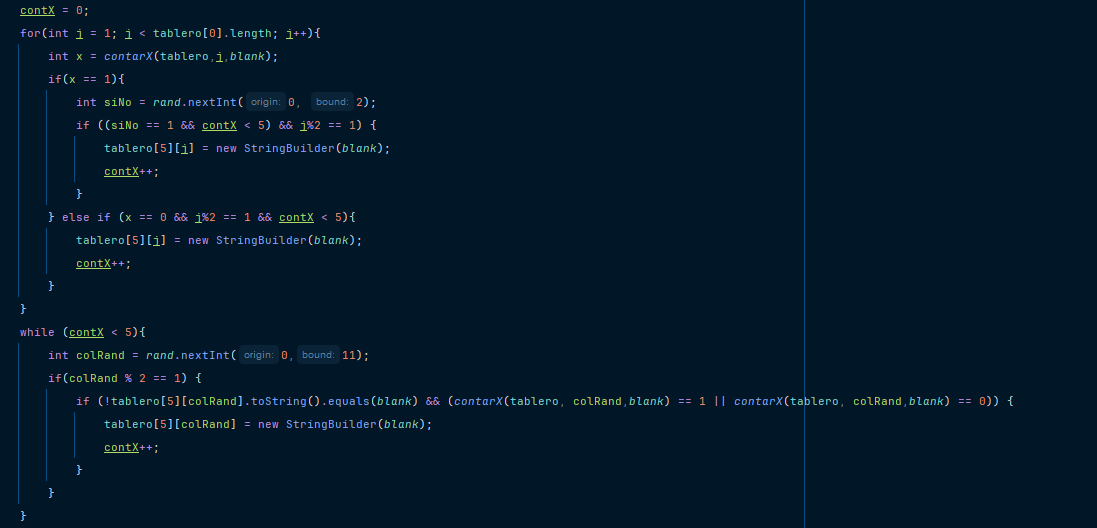
*En aquest cas fico primer la sortida perquè el codi és molt llarg.*

Sortida:



Codi:



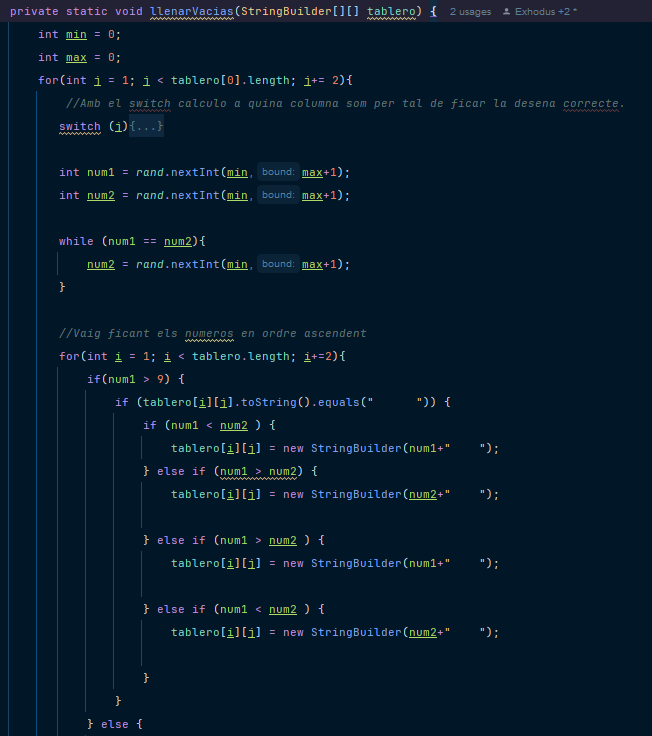


# Omplir casellas amb numeros.

Aquí he creat una funció anomenada "llenarVacias" per omplir els vuits del cartó del jugador amb números segons la seva columna. He creat 2 variables: un mínim i un màxim, que varien segons la columna utilitzant un "switch" amb la columna. Després he creat dues variables més: "num1" i "num2", inicialitzant-les amb un "rand" amb els límits "min" i "max+1" per saber el rang que haig de crear.

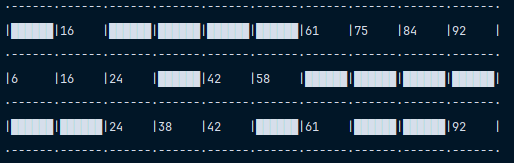
Tot seguit, he comparat els números per saber quin dels 2 és més petit i ordenar-los.

*El codi es de 100 líneas, per tant he plegat el switch per tal de que es pugui visualitzar millor*





Sortida:



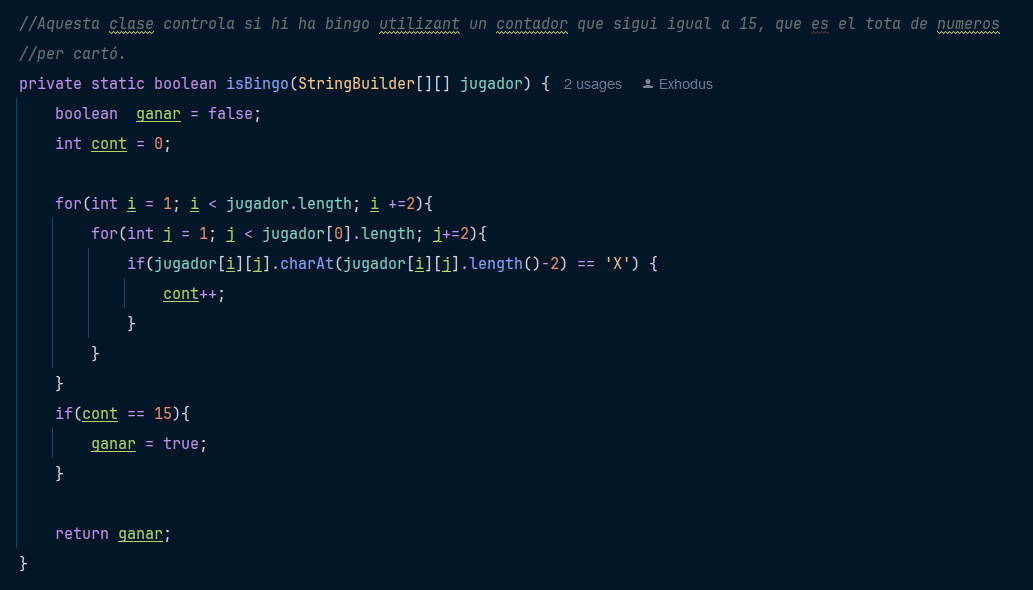
***\*Fent el codi sencer d'aquesta forma m'asseguro que cada cartó és diferent de l'anterior i que cada cop que es juga el cartó és diferent. Com si fos un bingo de veritat.***

# Funcions.

### isBingo().

Funció booleana. Utilitzo 2 variables. La variable "ganar" que serà l'output final i un comptador.

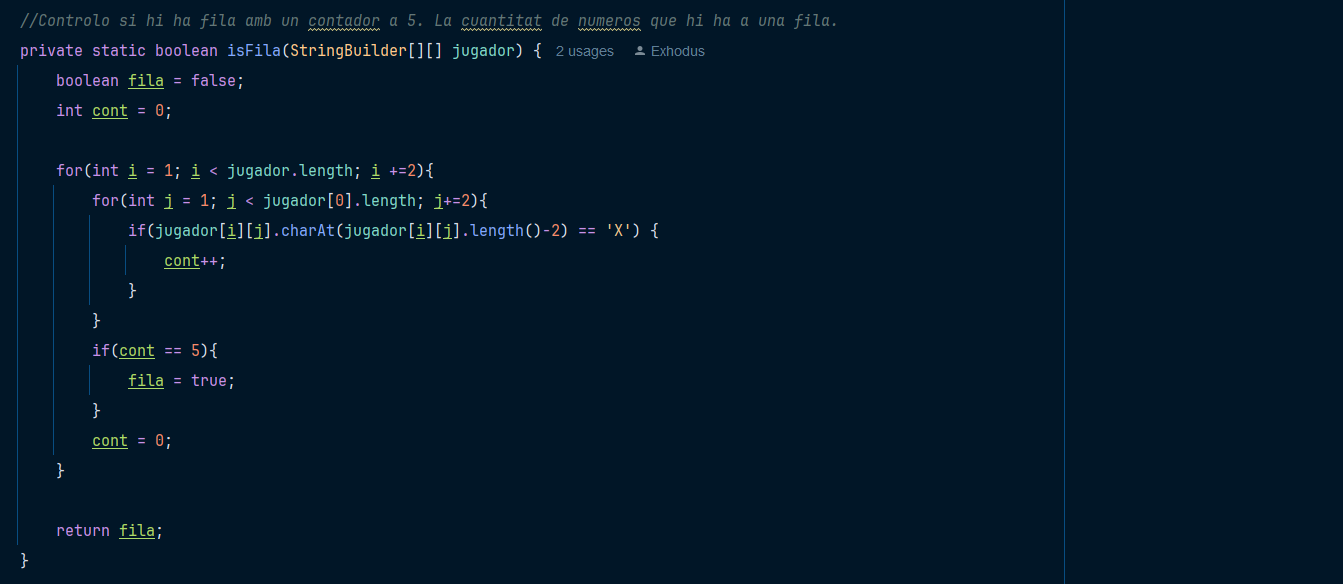
Recorro la matriu sencera del cartó del jugador comptant quantes caselles tenen el caràcter "X".Per cada casella afegeix-ho 1 al comptador. Si el comptador arriba a 15 al final de la matriu "ganar" passa a ser positiu perquè representa que hi ha Bingo. Independentment de si el comptador ha arribat a 15 o no, la variable "ganar" passa el seu valor a la variable la qual s'ha igualat a la funció.



### isFila().

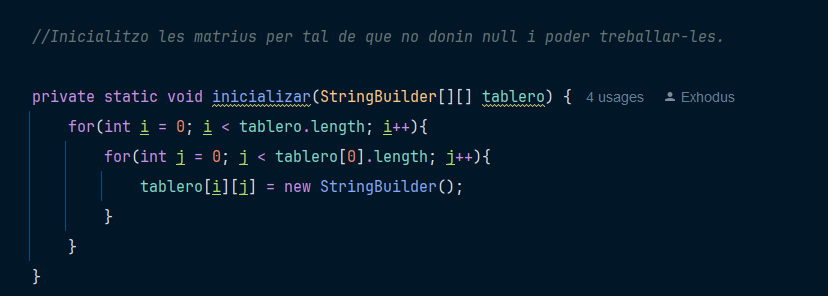
Al igual que la funció “isBingo”, utilitzo 2 variables. La variable "fila" que serà l'output final i un comptador. En aquest cas miro si al acabar les columnes o el “matriu[0].length” el comptador es igual a 5, sinó es passa a la següent fila i el comptador es reinicia.

la funció retorna el valor de la variable “fila”.



### inicializar().

Un doble bucle per tal d’inizialitzar tots els StringBuilders de las matrius i que no siguin nulls.

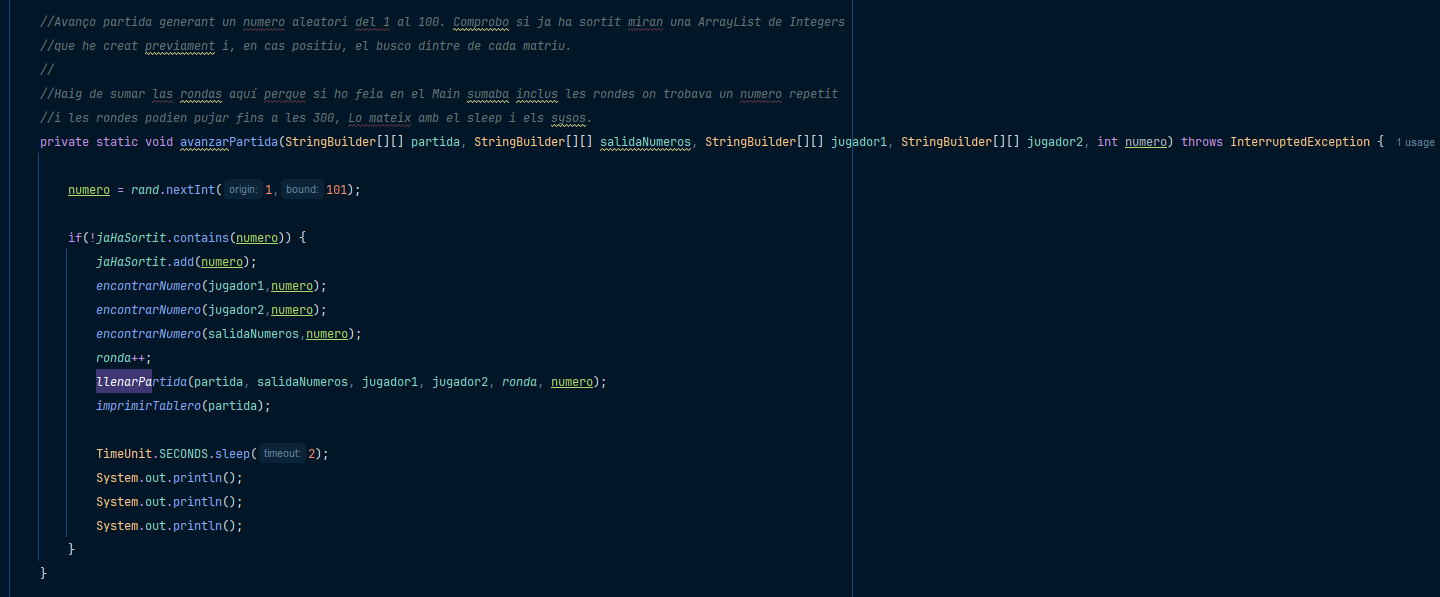


### avanzarPartida().

Es una funció dissenyada per que la partida avanci, crida a altres funcions, imprimeix la partida per consola i fica un “delay” de 2s.

La funció comença creant un número entre l’1 i el 100. Tot seguit, comprovo si el número ja ha sortit abans amb una ArraiList<Integer> que he creat estàtic amb anterioritat amb el “contains”. En cas negatiu s’afegeix a la llista i es busca el número en cada cartró amb una funció anomenada “encontrarNumero”, es suma 1 a les rondes, s’ompla la matriu “Partida” amb totes les matrius de jugadors i taulell de numeros i s’imprimeix “Partida” per consola.

Hi ha un “delay” de 2s i 3 salts de línea per deixa espai i que sigui coherent el que es veu per pantalla.



### encontrarNumero().

Recorro tota la matriu i comprovo si el número existeix dintre d'aquesta. Si existeix se’l afegeix una “X” a la penúltima posició.

He escollit la penúltima perquè sigui més fàcil de diferenciar i no s'enganxi a paret de la dreta.



### numerarCasillas().

Inicio amb un contador a 1, que será el numero de la primera casella. Recorro tota la matriu desde la posició i=1 i j =1 i li sumo 2 a cada volta, per tal de que sempre recorri els espais en blanc. Vaig ficant el numero del comptador a cada casella fins que arriva a 100.

Aquesta funció es exclusiva pel cartró dels números (“salidaNumeros”).

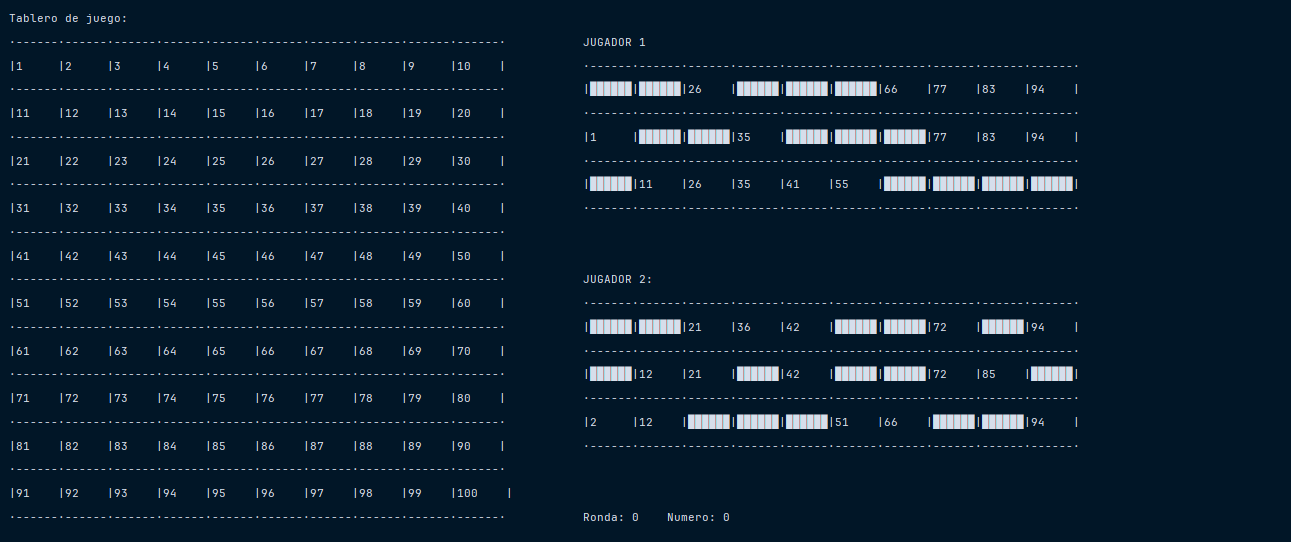


### llenarPartida().

Recorro tota la matriu de “Partida” que es una matriu de 21x43. fins que la J es igual a 21, omplo la matrius “Partida” amb la matrius “salidaNumeros”, després afegerix-ho un espai en blanc en una columna i tot seguit fico el cartró dels jugadors. Avaix de tot fico el numero que ha sortit en aquesta ronda i el número de la ronda.



Sortida:



### llenarVacias().

Creo un parell de variables anomenadas min i max i las inizialitzo en 0. Faig un switch per saber a quina columna soc del cartró dels jugadors i donar-li els valors corresponents a les variables min i max. Tot seguit, creo 2 variables més, num1 i num2. Les inicialitzo amb un “rand” entre el min i el max+1. Comprovo que no sigui el mateix número i després els agrego a la matriu en ordre ascendent.

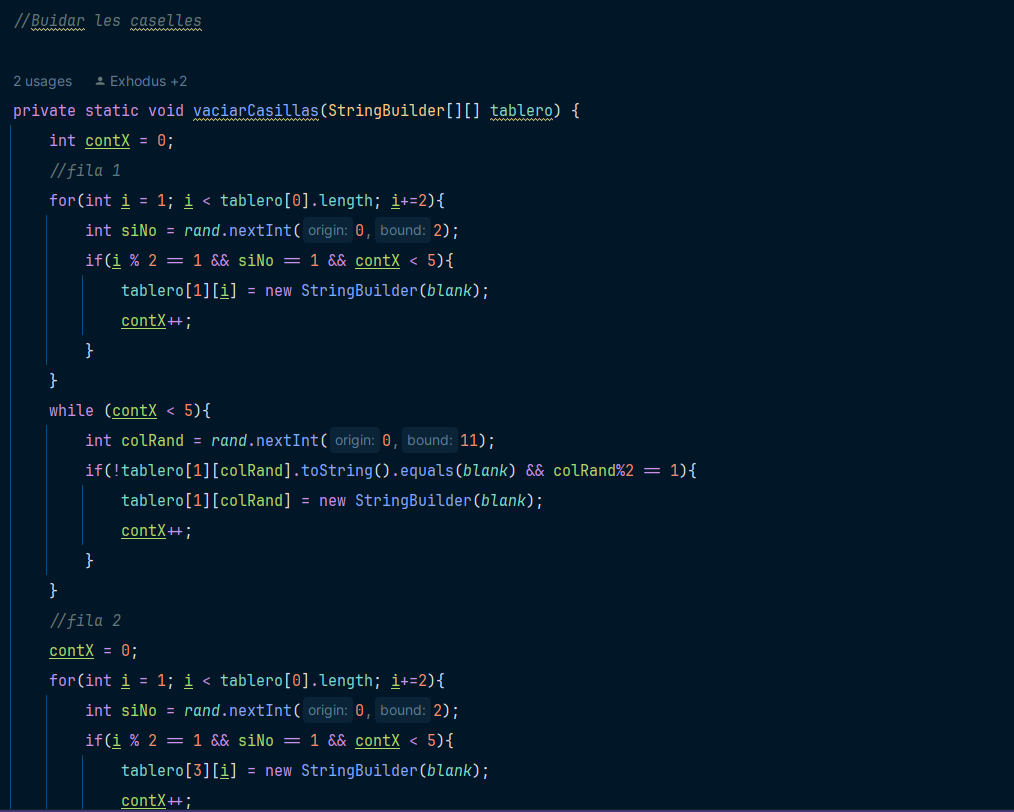
**(el codi és més amunt)**

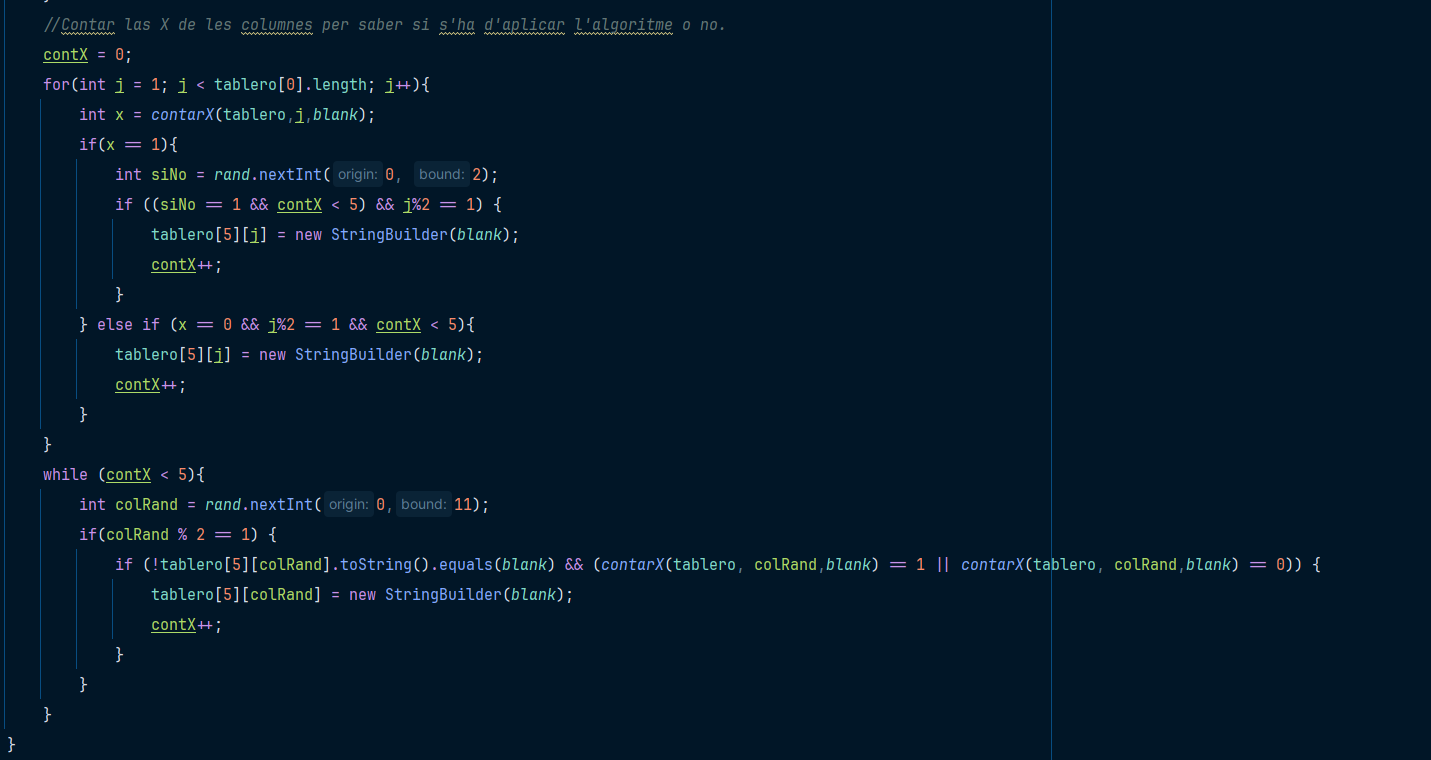
### vaciarCasillas().

Inicio la funció amb una variable “contX” a 0. La variable es diu així perquè fent proves vaig ficar “XXXXX” a les caselles que s'havien de tapar. Recorro la primera fila columna per columna amb un rand de 0 o 1. Si es 1 i el contX está per sota de 5, la casella es tapa. En cas de que al acabar la fila sencera no hi hagin encara 5 casellas tapades, entra dintre d’un bucle “while” fins que el contX = 5. Inicialitzo una variable int amb un rand entre 0 i 11 que es el numero de columnes que hi ha dintre de la matriu. Comprovo que la columna sigui impar i que la columna no estigui tapada amb anterioritat. Si no está tapada afegeixo +1 al contador i la tapo.

A la tercera fila comprovo si a la mateixa columna de J hi ha alguna casella tapada utilitzant una funció: “contarX”. Si hi ha 1, utilitzo un rand amb 0 o 1, essent 1 positiu per tapar la casella. No contemplo la posibilitat de 2 perque en aquest cas no s’ha d’afegir.

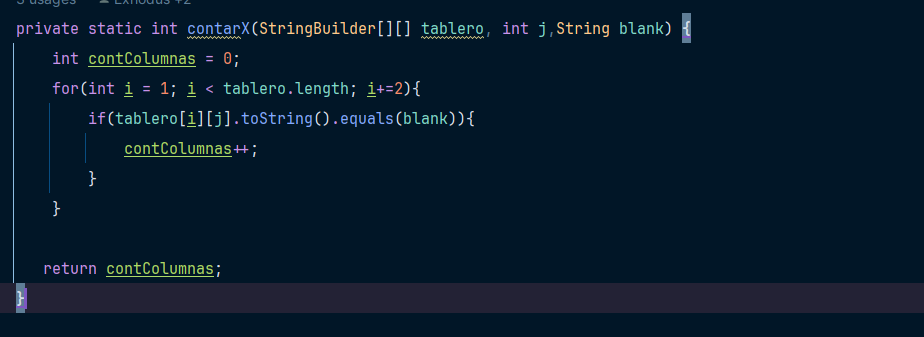
Tot seguit, utilitzo un “while” per mirar el comptador “contX”. Funciona exactament igual als 2 anteriors però al if també comprobo si la funció “contarX” retorna 1 o 0.



****

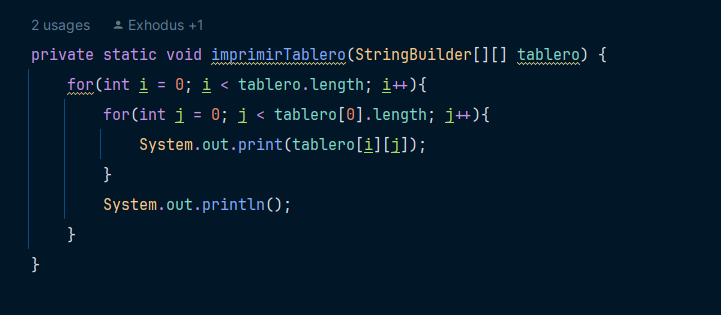
### contarX().

Compto cuantes caselles estan tapades dintre de la mateixa columna.



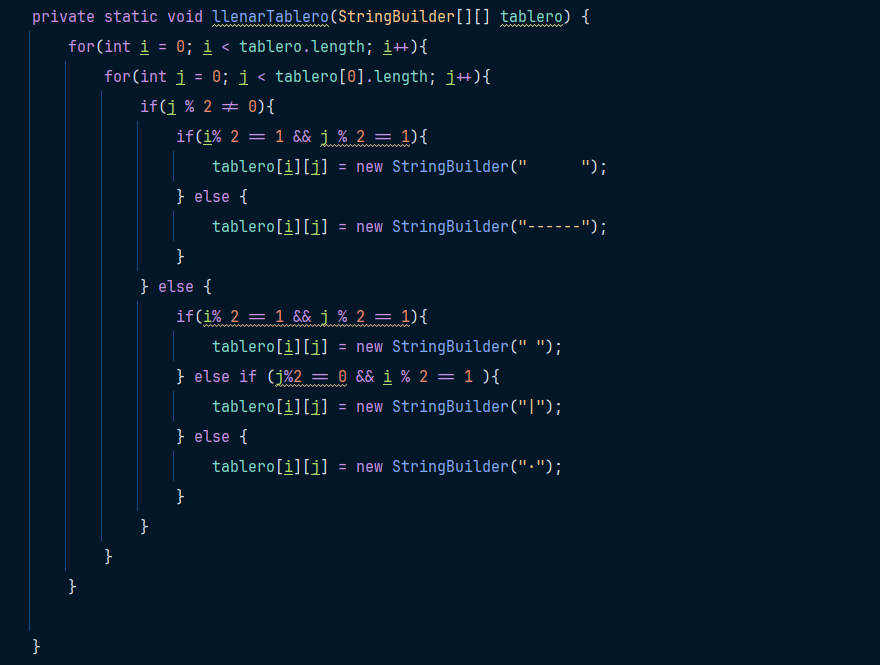
### imprimirTablero().

Imprimeix la matriu.



### llenarTablero().

Fa els marcs del tauler de joc.



# Bibliografía.

1. Delay:
   1. StackOverFlow.com : <https://stackoverflow.com/questions/24104313/how-do-i-make-a-delay-in-java>