Practica 2 Programació modular.

En està practica vos passem la estructura de l'objecte TaulerJocBuscamines per a que la completeu seguin les instruccions del document i feu unes partides al famós joc.

Juntament amb l'estructura de l'objecte TaulerJocBuscamines, teniu dos fitxers més, un anomenat Casella que conté tota la informació necessària per gestionar les diferents caselles del Joc i la part gràfica que podem comentar a classe si teniu curiositat.

Construir l'objecte TaulerJocBuscamines.

Les dades que necessitem per construir l'objecte són:

numFiles, numColumnes i numMines, per tant el constructor de l'objecte ha d'assignar valor a aquestes tres variables i cridar a la funció inicialitzarCaselles.

```
public TaulerJocBuscamines(int numFilesIn, int numColumnesIn, int numMinesIn) {



inicialitzarCaselles();
}
```

Inicialitzant les Caselles inicialitzarCaselles();

Fixeu-vos que la primera variable global que es declara en l'objecte TaulerJocBuscamines és exactament això, el tauler, una matriu de n*n que contindra n*n objectes de tipus Casella.

Com creem un objecte nou de tipus casella?

```
new Casella(i,j);
```

Aquesta acció tan senzilla cridarà a l'objecte Casella i li assignarà una fila i una columna a la casella.

Però això ho hem de fer per a tot el TaulerJocBuscamines en la matriu caselles[][] que ja està declarada, ja sabeu com.

Quan acabem cridarem a la funció generarMines();

```
public void inicialitzarCaselles(){

   generarMines();
}
```

Posem les mines al Joc generarMines();

En esta funció hem de posar el numero de mines indicat en la variable numMines dins del nostre joc, ho farem generant aleatòriament un número de fila i un número de columna, comprovarem si en la (fila,columna) generades ja hi ha alguna mina, si no és aixina la posarem.

Per a fer-ho disposem de dos mètodes del objecte casella (esMina i ponMina), recordeu-vos que heu de posar un número de mines concret, per tant aneu contant-les.

En acabar la lògica de generarMines cridarem a la funció actualitzarNumeroMinesAlVoltant.

```
private void generarMines(){

        actualitzarNumeroMinesAlVoltant();
    }
```

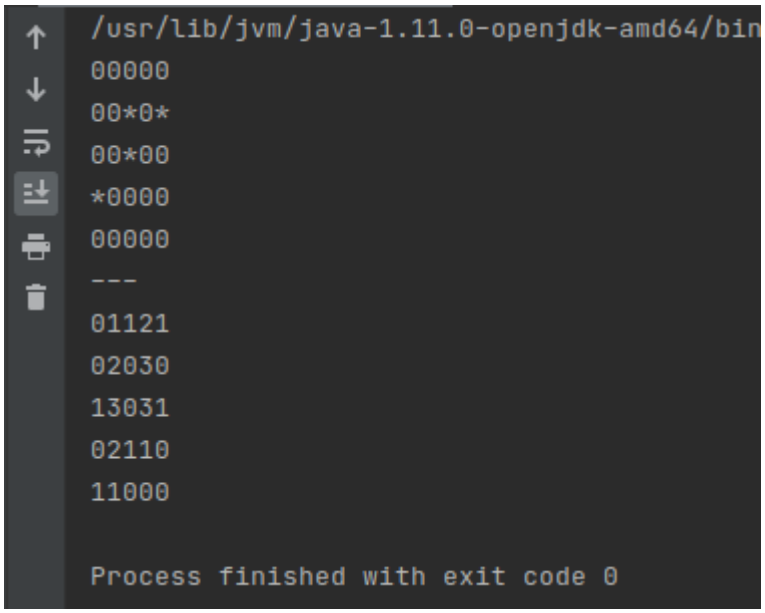
Calculant el número de mines en cada casella actualitzarNumeroMinesAlVoltant();

Com sabeu el buscamines mostra en cada casella que no hi ha una mina, el numero de mines que té en les restants 8 caselles adjacents, Això és exactament el que ha de fer la funció. Per cada casella del tauler que no està ocupada per una mina, calcular el número de mines que hi ha al seu voltant.

Com és una operació complexa, la hem dividit en dos funcions.
ActualitzarNumeroMinesAlVoltant: que recorrerà totes les caselles del tauler.

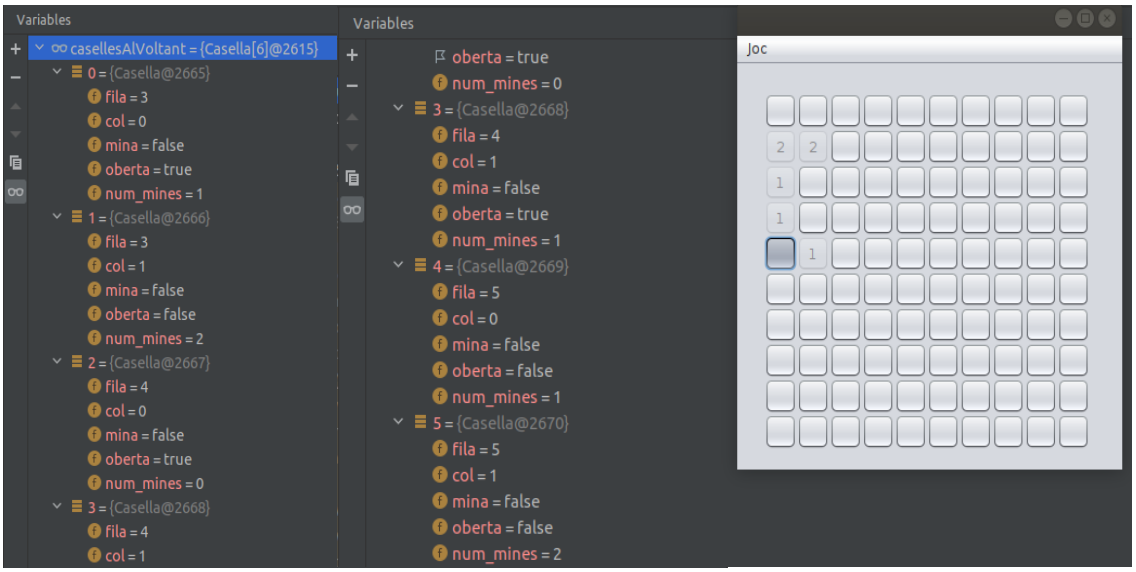
CalculaNumMines: Per cada casella que li passa la funció anterior retornarà un integer amb el número de mines que li correspon a la casella.

Per comprovar si totes les funcions desenvolupades fins ací són correctes, cal implementar dos funcions molt senzilles que ens mostre per una banda la disposició de les mines en el tauler i el càlcul de mines per altra banda. Seran les funcions anomenades **imprimirTauler()** i **imprimirPistes()**.



Array de caselles adjacents obtenirCasellesAlVoltant.

Per programar el joc amb la funcionalitat completa, cal obtindre un array de les caselles que hi ha al voltant d'una casella (fila,col) d'aquesta manera podrem destapar totes les caselles d'una àrea de la matriu que no tenen mines. Es dir, si l'usuari marca la casella (0,4) de la figura cal un array amb les caselles {(3,0},(3,1),(4,0),(4,1),(5,0),(5,1)}, tal i com podeu observar en la imatge.



Array caselles amb mines **obtenirCasellesAmbMines**

Per gestionar la dinàmica del joc d'una manera més eficient, també necessitarem un array de caselles amb mines. Cal implementar aquesta funcionalitat.

Si tota la implementació és correcta, ja podeu començar a jugar al buscamines.