Juego del Buscaminas.

observa una explicación del juego y observaciones sobre su código en esta página http://www.genbetadev.com/java-j2ee/crea-tu-propio-buscaminas

Lo que pone en está página te puede ayudar pero realmente ya estás preparado para solucionarlo ya que utiliza los conceptos de celdas adyacentes y recursividad que conoces.
Para familiarizarte con el juego puedes echarte una partida en un busca minas online.

De esa página resaltamos que podemos usar las siguientes ideas:

- Utilizar dos matrices paralelas. Una almacena de cada casilla un número de 0-9 y la otra almacena de cada casilla si está destapada o no. La posibilidad de marcar una casilla como sospechosa de mina (la banderita del juego gráfico) puedes omitirla para simplificar.
  No es obligatorio en absoluto que uséis matrices paralelas. Hay otros enfoques, por ejemplo usar una única matriz donde cada elemento sea de la clase Celda que será un objeto con dos campos, uno para almacenar valor y otro estado(tapado/destapado). Y hay muchos otros enfoques.
- cómo calcular las adyacencias.
  Repasa el boletín de arrays multidimensionales si no recuerdas o no dominas el concepto.
- destapar una casilla con valor 0 =>hacer un destapado recursivo. Si una celda vale 0 entonces tengo que ir mirando "alrededor" si puedo destapar sus posibles 8 vecinas.

Además observa que:

- en la página, hace, en mi opinión, un muy buen uso del pseudocódigo, no se dedica a hacer un programa detallado en pseudocódigo, si no que esboza "las partes difíciles".
- no separa la E/S de la lógica del juego, no usa mil clases, no usa get/set y esto lo hace breve y conciso. No obstante nosotros por motivos didácticos en esta práctica estamos obligados a hacerlo con dos capas.
- y por último observa que el "recorrido por 8 vecinos" se usa en dos situaciones: al colocar minas y al destapar casillas con valor 0

MINI EXPLICACIÓN DE COLOCACIÓN DE MINAS.

La colocación se basa en el concepto de celdas adyacentes y por tanto quizá esta explicación sea superflua. Veámosla de todas formas.

En el código de página recomendada, para generar el tablero inicial de juego, va poniendo minas aleatoriamente, y cada vez que coloca una nueva mina actualiza el valor de las celdas adyacentes. Como la matriz de valores es de int, usar los números de 0 a 8 para indicar las minas que tiene alrededor una celda. En el caso de que la celda sea una mina usamos otro valor por ejemplo 9.
Ejemplo:
en principio el tablero estará vacío de minas

0	0	0

0	0	0
0	0	0

si coloco una mina en [0][2], el tablero será

0	1	9
0	1	1
0	0	0

ahora supongo que tengo que colocar otra mina en [1][1]

el valor antiguo de [1][1] se sobreescribe y pasa a valer 9 y se actualizan las 8 celdas posibles alrededor:
[0][0]++
[0][1]++
[0][2] hay mina y no se incrementa
[1][0]++
[1][1] no es alrededor, es la celda en que puse la mina
[1][2]++
[2][0]++
[2][1]++
[2][2]++

nos queda el tablero

1	2	9
1	9	2
1	1	1

ahora tengo que colocar otra mina en [0][0]

Su "alrededor" es
[-1][-1] no existe
[-1][0] no existe
[-1][1] no existe
[0][-1] no existe
[0][0] es donde voy a colocar la nueva mina
[0][1]++
[1][-1] no existe
[1][0]++
[1][1] hay una mina y no se toca
nos queda

9	3	9

2	9	2
1	1	1

Otro enfoque para generar el tablero inicial sería primero poner aleatoriamente las 10 minas y luego barrer toda la matriz para calcular las adyacencias de cada celda. Son dos enfoques con igual resultado e igual dificultad.