Introducción a Ciencias de la Computación
Tarea 5: Busca minas

Pedro Ulises Cervantes Gonzáalez
confundeme@ciencias.unam.mx

Emmanuel Cruz Hérnandez
emmanuel_cruzh@ciencias.unam.mx

Adriana Sánchez Del Moral
adrisanchez@ciencias.unam.mx

Víctor Zamora Gutiérrez
agua@ciencias.unam.mx

Fecha de entrega: Jueves 30 de abril 2020 11:59pm

1 Preguntas

Puedes crear un archivo prueba.java para ayudarte a responder la siguiente seccion, adjunta tu archivo a la tarea.

1. Supón que tenemos el siguiente objeto en nuestro código

```
char[] arregloCaracteres = new char [1];
```

- ¿Cuál es la forma correcta de comparar el valor por default? tal que el resultado de la comparación sea true.
  (a) arregloCaracteres[0] == ”
  (b) arregloCaracteres[0] == null
  (c) arregloCaracteres[0] == ””
  (d) arregloCaracteres[0] == 0
  (e) arregloCaracteres[0] == ’ ’
- ¿Qué regresa la terminal si ejecuto lo siguiente?

```
arregloCaracteres[0] = 'a';
System.out.println(arregloCaracteres[0]);
```

  ¿Y si cambiamos el indice a lo siguiente?

```
System.out.println(arregloCaracteres[-1]);
```

  ¿Y cambiando el indice nuevamente a lo siguiente?

```
int maxIndex = arregloCaracteres.length;
System.out.println(arregloCaracteres[maxIndex]);
```

  En conclusión, hay que tener cuidado con los limites de nuestro arreglo.

2. ¿Para qué sirve Character.forDigit(a, 10)? ¿Qué valor nos regresaría el siguiente código?

```
int minas = 4;
char minasC = Character.forDigit(minas, 10);
int minas2 = 11;
char minasC2 = Character.forDigit(minas2, 10);
System.out.println(minasC);
System.out.println(minasC2);
```

2 Buscaminas

- Requerimiento:

  Nos piden implementar el juego de Buscaminas, tal que permita al usuario elegir dos **coordenadas** en el tablero y mostrar el contenido de la casilla seleccionada, ademas, como en el juego original vamos a destapar mas espacios libres de bombas: n espacios hacia arriba, n hacia abajo, n a la izquierda y n a la derecha, formando una cruz a partir del espacio seleccionado.
  Ademas, por cada casilla destapada podremos saber si hay minas al rededor de ella utilizando la idea del juego original. Tendremos 4 niveles en el juego

  - Nivel principiante: 8 8 casillas y 10 minas.
  - Nivel intermedio: 16 16 casillas y 40 minas.
  - Nivel experto: 16 30 casillas y 99 minas.
  - Nivel personalizado: en este caso el usuario personaliza su juego eligiendo el número de minas y el tamaño de la cuadrícula.

- Especificaciones:
  El juego va a estar modelado con dos arreglos bidimensionales de caracteres que simulan los tableros a utilizar.
  ¿Por qué dos tableros? En el tablero "tablero" vamos a definir la información tras bambalina del juego, es decir, la información que el usuario no sabe que existe y en la que basaremos su juego, es decir, las posiciones de las bombas, los números de bombas por contorno de espacio, etc. En el segundo tablero "tableroJugado" sera en el que el usuario va a tirar.

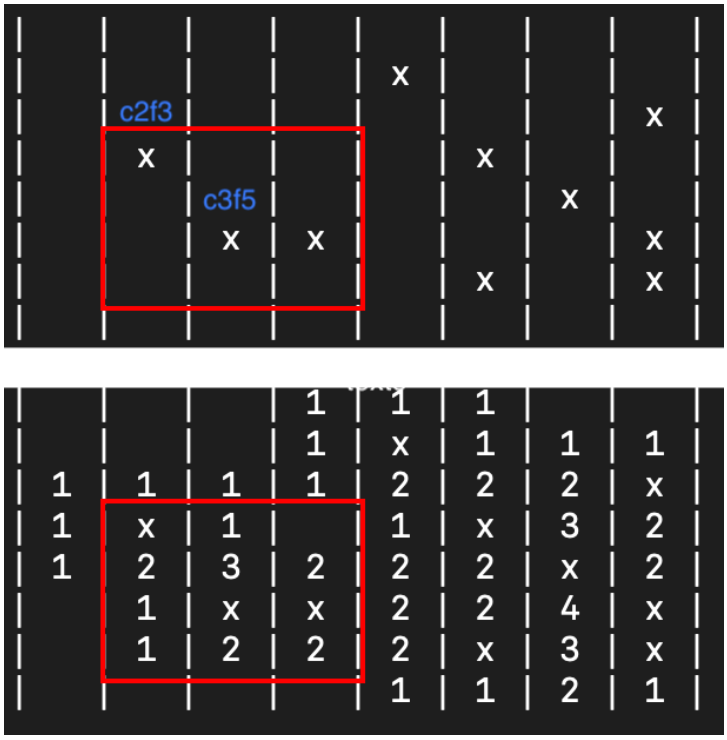
  1. **Constructor**
     El constructor del buscaminas tendrá solo un parámetro llamado "dificultad" con el que definiremos el nivel de dificultad a jugar (tamaño y numero de minas descrito en requerimiento), con el que sabremos el tamaño de la matriz/tablero y también llamaremos a la función "setMinas(n)" y setNumMinas(); Ademas, habrá una variable "descubre" que indica el numero de casillas a descubrir junto con la posición, no cambiar este numero.
  2. **setMinas**
     El método setMinas recibe un parámetro de tipo entero que nos indica el numero de minas que el tablero "tablero" tendrá.
     Las minas serán puestas en posiciones aleatorias x y y tales que vayan desde cero hasta n-1 donde n es el tamaño de cada arreglo o subarreglo. Las bombas serán representadas por un char 'x'.

     ```
     int x = rand.nextInt(tablero.length);
     int y = rand.nextInt(tablero[0].length);
     ```

     No olvides importar Random.
  3. **setNumMinas**
     Este método no recibe ningún parámetro, regresa el numero de bombas que tiene cada casilla en sus casillas vecinas (3 arriba, una izquierda, una derecha y 3 abajo).
     Por ejemplo en la casilla (c2f3) en la columna 2 fila 3 sabemos que de sus vecinos solo la casilla columna 2 fila 4 tiene una bomba.
     En cambio la casilla (c3f5) en la columna 3 fila 5, tiene 3 bombas a su alrededor por lo que tendrá un valor de 3, recuerda que nuestro arreglo es de caracteres, por lo que deberás convertir el int en char.

     Las bombas no deben ser sobreescritas con ningún valor, así que si caemos en el caso de una casilla con valor 'x' la evitamos. Utiliza un continue para continuar las iteraciones pero evitar ejecutar el resto del código

     Este es el método mas complejo que necesitaran implementar, por lo que no olvides tomar en cuenta las restricciones de las orillas.



4. **tirar**
   Este método recibe dos enteros como parámetros, x y y, los cuales definen los indices en donde el usuario desea descubrir lo que hay en el interior de la posición, ya sea una bomba, casilla vacía o numero de bombas al rededor.
   Si la casilla es una bomba, entonces el usuario pierde.

   Ademas, este método descubre:
   - n posiciones hacia arriba de x.
   - n posiciones hacia abajo de y
   - n posiciones hacia la derecha de y
   - n posiciones hacia la izquierda de y

   Formando la cruz de la que se hablaba en requerimientos.
   Sin embargo si la casilla a descubrir es una bomba, ya no seguimos descubriendo a partir de esa casilla.

   Esta información la modelaremos en el tableroJugado y asignaremos el valor 'o' para las casillas abiertas.

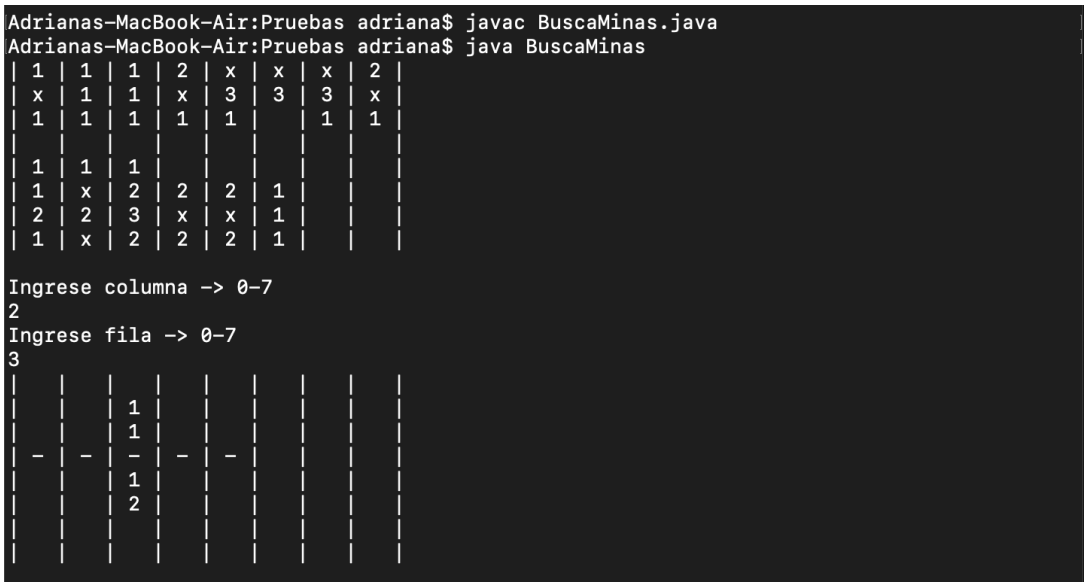
5. **imprimir**
   Este método no recibe nada pero regresa una cadena. En ella representamos el juego final del usuario, tomando en cuenta el tablero "tablero" y el tablero "tableroJugado". Si en nuestro "tableroJugado" hay una 'o' es por que esas casillas ya están abiertas para el usuario y podemos mostrar lo que el tablero "tablero" guardaba.
   Este método ya esta terminado. No cambies nada de su código.

6. **terminaElJuego**
   Este método regresa un booleano:
   True: si el usuario perdió marcando una bomba
   True: si el usuario lleno el tablero sin perder
   False: caso contrario a los anteriores.
   En caso de terminar el juego imprimimos en consola si el usuario gano o perdió la partida

7. **main**
   Genera un menú para el usuario en donde seleccione el nivel a jugar y una vez inicializado

el objeto BuscaMinas permitir que el usuario elija las posiciones a tirar hasta que el juego termine.

Todo junto, el juego debería verse como sigue:
(el primer tablero fue impreso con la ayuda del método imprimirBack comentado en el archivo)



Adjunto a esta tarea un archivo BuscaMinas.java en el cual podrás encontrar código auxiliar para tu implementación, revísalo antes de decidir empezar tu implementación desde cero, te puede ser de ayuda.

No cambies las firmas de los métodos, ni el tipo de accesos de las variables globales. El cambio a alguno de estos se te penalizara con -1 punto global.