Informática Aplicada a la Ingeniería Civil
Solución del Primer Parcial de Java
7 de abril de 2008

El Buscaminas (en inglés: Minesweeper) es un videojuego para un jugador inventado por Robert Donner en 1989. El objetivo del juego es despejar un campo de minas sin destapar ninguna mina.

El juego ha sido programado para muchos sistemas operativos, pero debe su popularidad por la versión que viene con el sistema operativo Microsoft Windows.

El juego consiste en despejar todas las casillas de una pantalla que no oculten una mina, para ello se tiene la siguiente información (véase la Figura 1):

Figura 1: Representación del problema del Buscaminas.

1. Algunas casillas tienen un número, este número indica las minas que suman todas las casillas circundantes. Así si una casilla tiene el número 3 significa que de las ocho casillas que hay alrededor (si no está en una esquina o borde) hay 3 con minas y 5 sin minas. Si se descubre una casilla sin número indica que ninguna de las casillas vecinas tiene mina y éstas se descubren automáticamente.

2. Si se descubre una casilla con una mina se pierde la partida.

3. Se puede poner una marca en las casillas que el jugador piensa que hay minas para ayudar a descubrir las que están cerca.

4. El juego también posee un sistema de récords para cada uno de los tres niveles en el que se indica el menor tiempo en terminar el juego. Los niveles son (para las nuevas versiones):

Nivel principiante: 9×9 casillas y 10 minas.

Nivel intermedio: 16×16 casillas y 40 minas.

Nivel experto: 30×30 casillas y 100 minas.

En versiones anteriores a Windows 2000 la pantalla sólo mide 8×8 y fue agrandada para evitar que la probabilidad de hacer clic en una mina fuera la misma que en el nivel intermedio: $10/(8 \times 8) = 10/64 = 40/256 = 40/(16 \times 16)$.

También se puede personalizar la dificultad de juego según el tamaño de la pantalla y el número de minas.

Una vez conocidas las reglas del juego se va a proceder a elaborar un programa para implementarlo. El tablero consiste internamente en una matriz cuadrada de dimensiones acordes al nivel de dificultad y en el que se dispondrán un número determinado de minas al azar. Para que el ordenador reconozca la existencia de una mina en la posición correspondiente de la matriz se ubicará un -1. En las demás casillas se pondrán números enteros correspondientes con el número de minas que hay en las casillas de alrededor.

Para la elaboración del programa se comienza creando una clase denominada **Buscaminas** que tiene las características siguientes:

1. La clase pertenece al paquete **buscaminas**. (2 puntos)

   ```
   package buscaminas;

   public class Buscaminas {
   ```

2. Se genera un constructor para permitir la inicialización de un objeto de esta clase sin dar ningún tipo de argumento mediante la sentencia `super();`. (2 puntos)

   ```
   public Buscaminas() {
       super();
   }
   ```

3. Las variables de instancia que caracterizan un objeto de la clase **Buscaminas** son: las dimensiones (**dim**) de la matriz, que es cuadrada (mismo número de filas que de columnas), la propia matriz en la que se almacenarán las minas y los demás datos numéricos (**data**), y el número de minas que se colocarán en el tablero (**nm**). Inicialmente sólo se declaran la variables sin inicializarlas porque aún no se dispone de información para reservar espacio en memoria. Se recomienda usar la variable entera que menos espacio en memoria ocupe. (4 puntos)

   ```
   byte dim;
   byte data[][];
   byte nm;
   ```

4. Antes de proceder a la creación del constructor que permite inicializar los valores de las variables (**dim**), (**nm**) y (**data**) en función del nivel de dificultad, se van a crear tres métodos que posteriormente se emplearán en el constructor.

a) La primera se llama **initmina** permite la inicialización del tablero ubicando las minas de forma aleatoria. Lo único que hace es seleccionar las ubicaciones tanto por filas como por columnas mediante la sentencia `Math.ceil(dim*Math.random())`, que utiliza dos métodos `static` de la clase `Math` y que devuelve un número aleatorio de tipo `double`[1] entre 1 y `dim`. Ha de tenerse en cuenta que no se pueden ubicar más de dos minas en la misma posición y que han de colocarse exactamente `nm` minas, ni más ni menos. En la posición obtenida al azar hay que colocar un -1, que implica que esa posición contiene una mina. (10 puntos)

```
public void initmina(){
    byte naux = 0;
    byte i,j;
    while(naux<nm){
        i = (byte) Math.ceil(dim*Math.random());
        j = (byte) Math.ceil(dim*Math.random());
        if(data[i-1][j-1]!= -1){
            data[i-1][j-1]=-1;
            naux +=1;
        }
    }
}
```

b) EL segundo método que se llama **numinas** se emplea para calcular el número de minas que hay alrededor de una casilla dada. Nótese que las casillas interiores tienen 8 casillas alrededor, las casillas laterales tienen 5 y las de esquina tienen 3, por lo tanto ese número varía entre 0 y 8 como máximo. Recibe como argumentos las posiciones (**i**) y (**j**) de la casilla y devuelve el número de minas alrededor de la misma (**nma**). Nótese que con `i=1` nosotros hacemos referencia a la primera posición del vector y es importante tenerlo en cuenta a la hora de programar. (10 puntos)

```
public byte numinas (byte i, byte j){
    byte nma=0;
    byte k,l,auxi,auxj;
    for(k=1;k<=3;k++){
        for(l=1;l<=3;l++){
            if(!(k==2 && l==2)){
                auxi=(byte) (i+k-2);
                auxj=(byte) (j+l-2);
                if(!(auxi<1 || auxi>dim || auxj<1 || auxj>dim)){
                    if(data[auxi-1][auxj-1]==-1){
                        nma+=1;
                    }
                }
            }
```

[1] Ojo con esto.

```
            }
        }
        return(nma);
    }
```

c) Una vez ubicadas todas las minas, la función **rellena** ha de colocar en cada casilla *en la que no haya una mina* un número que se corresponde con el número de minas que hay alrededor de la casilla correspondiente. Empléese la función **numinas** del apartado anterior. (10 puntos)

```
public void rellena(){
    for(byte i=1;i<=dim;i++){
        for(byte j=1;j<=dim;j++){
            if(data[i-1][j-1]!=-1){
                data[i-1][j-1]=this.numinas(i,j);
            }
        }
    }
}
```

5. El segundo constructor recibe una de las siguientes cadenas de caracteres: "principiante", "intermedio" y "experto" que se encargan de inicializar las variables de instancia de la clase a sus valores correspondientes según el nivel. En caso de que el usuario introduzca una cadena de caracteres incorrecta se escribirá un mensaje en pantalla indicándolo y se seleccionará por defecto el nivel de principiante. Una vez reservado espacio en memoria, mediante los métodos definidos anteriormente se colocan las minas y se rellena la matriz con los números correspondientes. (10 puntos)

   Nota: Para este apartado puede resultar útil el método de instancia **equals(String str)** de la clase String que compara la cadena de caracteres desde la que se invoca al método con la cadena de caracteres `str` argumento de la función.

```
public Buscaminas(String nivel) {
   super();
   if(nivel.equals("principiante")){
       dim=9;
       nm=10;
   } else if(nivel.equals("intermedio")){
       dim=16;
       nm=40;
   } else if (nivel.equals("experto")){
       dim=30;
       nm=100;
   } else {
       System.out.println("Ese nivel no existe, ha de elegir entre principiante,
          intermedio, o experto. Por defecto se escoje principiante.");
       dim=9;
       nm=10;
   }
   data=new byte[dim][dim];
   this.initmina();
```

```
        this.rellena();
    }
```

6. La clase contiene un método que se llama **comprobar** que devuelve una variable booleana con valor `true` si hay una mina en la posición (**i**) y (**j**) de la casilla seleccionada. Las posiciones (**i**) y (**j**) se dan como argumentos al método. Nótese que con `i=1` nosotros hacemos referencia a la primera posición del vector y es importante tenerlo en cuenta. (10 puntos)

```
public boolean comprobar(byte i, byte j){
    boolean haymina= false;
    if(data[i-1][j-1]==-1){
        haymina= true;
    }
    return(haymina);
}
```

7. Por último, se crea un método que escribe los valores actuales (solución) de las casillas del Buscaminas y que se llamará **write**. La rutina ha de escribir cada fila en una línea distinta con los números alineados por la derecha como se muestra a continuación para un buscaminas ficticio de 5×5 con 8 minas:

```
-1  2  0  1 -1
-1  4  2  2  1
 3 -1 -1  2  1
-1  4  3  3 -1
 1  2 -1  2  1
```

Para este apartado puede ser de utilidad el método `System.out.print("texto")` que escribe lo que haya entre paréntesis sin pasar de línea, es decir, lo que se escriba después se escribe inmediatamente después de `texto` en la misma línea. Nótese que el número mayor que se puede escribir es 8, que implica que la casilla está totalmente rodeada por minas. (10 puntos)

```
public void write(){
    for(byte i=1;i<=dim;i++){
        for(byte j=1;j<=dim;j++){
            if(data[i-1][j-1]==-1){
                System.out.print(data[i-1][j-1]+" ");
            } else {
                System.out.print(" "+data[i-1][j-1]+" ");
            }
            if(j==dim){
                System.out.println(" ");
            }
        }
    }
}// Fin de la clase Buscaminas
```

A continuación se genera una segunda clase llamada **Principal** que contiene el método `main`. Todo lo que se pide a continuación se supone dentro del método `main` y no hace falta que generéis el código correspondiente:

```
package buscaminas;

public class Main {

    public static void main(String[] args) {
```

1. Declarar e inicializar una variable de tipo **Buscaminas** con vuestro nombre y que genere un buscaminas con nivel de principiante. (4 puntos)

   ```
   Buscaminas roberto = new Buscaminas("principiante");
   ```

2. Declarar e inicializar dos variables llamadas **i** y **j** que representan el índice de fila y de columna de la casilla $(3, 9)$, respectivamente. Nótese que el tipo ha de ser coherente con el tipo de variables usadas para las posiciones en la clase Buscaminas. Además tener en cuenta que el compilador JAVA, por defecto, interpreta que cualquier número entero es de la clase `int`. (4 puntos)

   ```
   byte i = 3;
   byte j = 9;
   ```

3. Declara una variable booleana que se llame **haymina** e inicializar su valor a verdadero o falso en función de si en la posición definida por las variables **i** y **j** del apartado anterior hay mina o no. Utilizar alguno de los métodos definidos en la clase Buscaminas. (4 puntos)

   ```
   boolean haymina = roberto.comprobar(i, j);
   ```

4. Calcular el número de minas que hay alrededor de la casilla **i** y **j** y almacenarlo en una variable. (4 puntos)

   ```
   byte nma = roberto.numinas(i, j);
   ```

5. Escribir en pantalla lo siguiente en función de si hay mina o no en la casilla $(3, 9)$ pero sin usar directamente los números 3 y 9: (8 puntos)

   ```
   Bien hecho, en la casilla (3,9) no hay ninguna mina,
   y a su alrededor hay 1 minas.
   ```

   ó

   ```
   En la casilla (3,9) hay una mina, has perdido."
   ```

```
if(haymina){
    System.out.println("En la casilla ("+i+","+j+") hay una mina, has perdido.");
} else {
    System.out.println("Bien hecho, en la casilla ("+i+","+j+") no hay ninguna mina,");
    System.out.println("y a su alrededor hay "+nma+" minas.");
}
```

6. Escribir la solución final en pantalla con una única sentencia. (4 puntos)

```
roberto.write();
```

7. Crear en otra variable un buscaminas con nivel experto y escribir la tabla generada (solución) en pantalla. (4 puntos)

```
Buscaminas roberto2 = new Buscaminas("experto");
roberto2.write();
```

Tiempo: 2h.