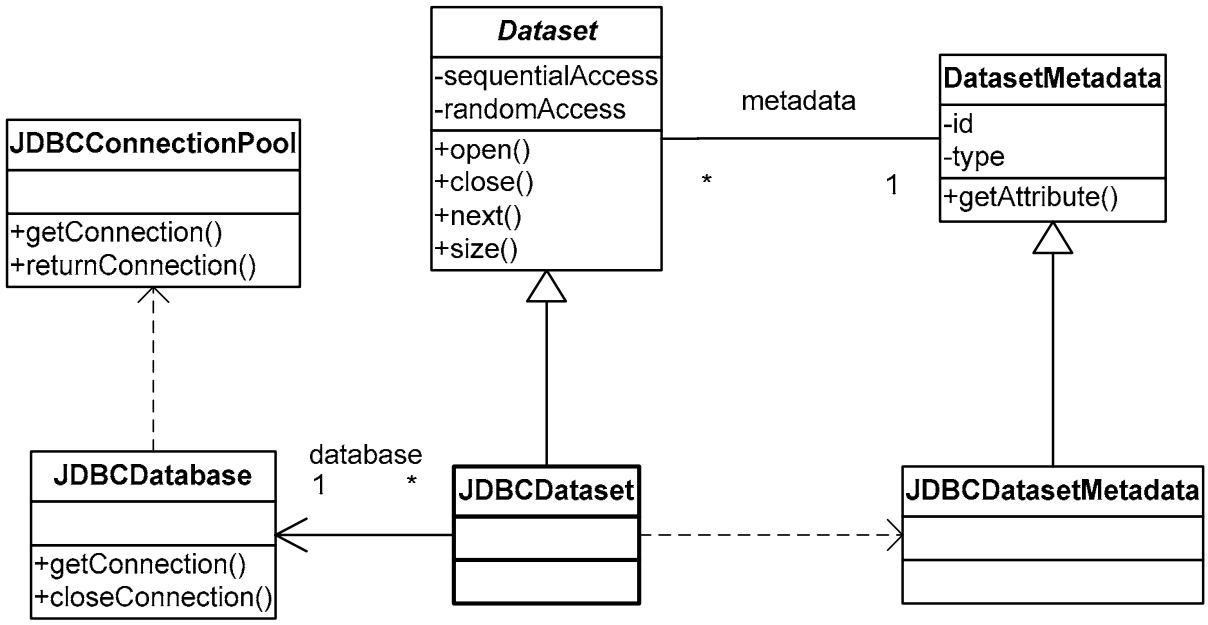
Examen parcial – Convocatoria de febrero de 2006
FUNDAMENTOS DE LA PROGRAMACIÓN

EJERCICIO 1

Defina adecuadamente las clases en Java que se derivan del siguiente diagrama de clases UML. Declare todas las variables de instancia necesarias para representar los objetos de las distintas clases que aparecen en el diagrama y declare (sin implementarlos) todos los métodos que considere oportunos.




```
class JDBCConnectionPool
{
  public Connection getConnection ()
  {…}

  public void returnConnection (Connection connection)
  {…}
}

class JDBCDatabase
{
  public Connection getConnection ()
  {…}

  public void closeConnection (Connection connection)
  {…}
}
```

```
class Dataset
{
    private boolean sequentialAccess;
    private boolean randomAccess;
    private DatasetMetadata metadata;

    public void open()
    {…}

    public void close()
    {…}

    public Object next()
    {…}

    public int size()
    {…}
}


class JDBCDataset extends Dataset
{
    private JDBCDatabase database;
}


class DatasetMetadata
{
    private String  id;
    private String  type;
    private Dataset datasets[];

    public String getAttribute(String id)
    {…}
}


class JDBCDatasetMetadata extends DatasetMetadata
{
}
```

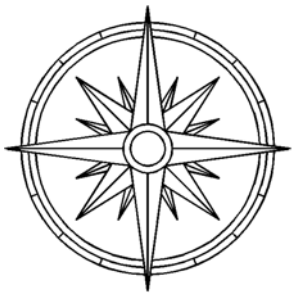
Examen parcial – Convocatoria de febrero de 2006
FUNDAMENTOS DE LA PROGRAMACIÓN

EJERCICIO 2

Escriba un método `encuentra()` para la clase `Matriz` que, dado un vector de números enteros que recibe como parámetro, nos indique la posición dentro de la matriz donde se encuentra el vector. El vector puede estar "oculto" dentro de la matriz igual que una palabra en una sopa de letras: puede aparecer horizontal, vertical o diagonalmente (al derecho y también al revés).

A	S	A	T	P	O	I	D	A	O	P	Z	Q	O
C	N	I	A	S	A	T	A	N	A	X	O	S	P
V	Y	O	S	L	H	L	J	V	T	L	I	F	S
R	L	Y	I	D	P	V	U	A	I	W	C	W	H
X	R	S	V	X	S	D	H	V	F	T	O	Q	C
G	P	X	B	H	J	Y	I	J	R	Z	W	M	B
N	R	R	D	G	T	N	Y	R	O	E	Z	W	Y
F	E	H	A	I	O	H	I	A	M	I	J	X	W
J	P	H	P	P	B	R	K	E	O	N	U	A	Q
L	A	S	A	P	U	T	B	J	R	R	Y	E	P
I	U	K	A	P	O	F	I	L	I	T	A	K	S
B	N	D	V	S	B	Y	K	Q	P	B	P	Y	X
B	J	J	Q	Q	A	P	W	S	N	Z	H	F	P
L	Q	C	H	C	D	Q	F	W	M	O	G	W	Q

```
public class Matriz
{
  private int datos[][];

  …

  public int[] encuentra (int vector[])
  {
       int i,j;
       int pos[] = new int[2];

       pos[0] = -1;
       pos[1] = -1;

       for (i=0; i<datos.length; i++) {
           for (j=0; j<datos.length; j++) {
               if (encuentraPos3(vector,i,j)) {
                  pos[0] = i;
                  pos[1] = j;
               }
           }
       }

       return pos;
  }

  private boolean encuentraPos (int vector[], int x, int y)
  {
       int     dx,dy;
       boolean encontrado = false;

       for (dx=-1; dx<=1; dx++) {
           for (dy=-1; dy<=1; dy++) {
               if (encuentraDir(vector,x,y,dx,dy)) {
                  encontrado = true;
               }
           }
       }

       return encontrado;
  }
```



```
  private boolean encuentraDir
                  (int vector[], int x, int y, int dx, int dy)
  {
      int     k, px, py;
      boolean ok = true;

      px = x;
      py = y;

      for (k=0; k<vector.length; k++) {

            if (  (px>=0) && (px<datos.length)
               && (py>=0) && (py<datos[0].length)) {

               if (datos[px][py]!=vector[k])
                  ok = false;

            } else {
               ok = false;
            }

            px += dx;
            py += dy;
      }

      return ok;
  }
```

Variantes y mejoras

1. Eliminación de algunas iteraciones innecesarias:

```
encuentra

     for ... && (pos[0]==-1) ...

encuentraPos

     for ... && !encontrado ...

encuentraDir

     for ... && ok ...
```

2. Reducción del número de comprobaciones en `encuentraDir`

```
private boolean encuentraDir2
        (int vector[], int x, int y, int dx, int dy)
{
    int     k, px, py;
    int     finx, finy;
    boolean ok = true;

    finx = x + (vector.length-1)*dx;
    finy = y + (vector.length-1)*dy;

    if (  (finx>=0) && (finx<datos.length)
       && (finy>=0) && (finy<datos[0].length)) {

       px = x;
       py = y;

       for (k=0; k<vector.length; k++) {

           if (datos[px][py]!=vector[k])
              ok = false;

           px += dx;
           py += dy;
       }

    } else {
       ok = false;
    }

    return ok;
}
```

3. Implementación alternativa de `encuentraPos`

```
private boolean encuentraPos2 (int vector[], int x, int y)
{
     return encuentraDir(vector,x,y, 1, 0)
         || encuentraDir(vector,x,y,-1, 0)
         || encuentraDir(vector,x,y, 0, 1)
         || encuentraDir(vector,x,y, 0,-1)
         || encuentraDir(vector,x,y, 1, 1)
         || encuentraDir(vector,x,y, 1,-1)
         || encuentraDir(vector,x,y,-1, 1)
         || encuentraDir(vector,x,y,-1,-1);
}
```

4. Por fuerza bruta (en las 8 direcciones)

```
private boolean encuentraPos3 (int vector[], int x, int y)
{
    int     k;
    boolean encontrado;


    encontrado = true;

    for (k=0; k<vector.length; k++)
        if ( (x+k>=datos.length) || (datos[x+k][y]!=vector[k]) )
           encontrado = false;


   if (!encontrado) {
      encontrado = true;

      for (k=0; k<vector.length; k++)
          if ( (x-k<0) || (datos[x-k][y]!=vector[k]) )
             encontrado = false;
    }


   if (!encontrado) {
      encontrado = true;

      for (k=0; k<vector.length; k++)
          if ( (y+k>=datos[0].length) || (datos[x][y+k]!=vector[k]) )
              encontrado = false;
    }


   if (!encontrado) {
      encontrado = true;

      for (k=0; k<vector.length; k++)
          if ( (y-k<0) || (datos[x][y-k]!=vector[k]) )
             encontrado = false;
   }
```

```
    // Diagonales


    if (!encontrado) {
       encontrado = true;

       for (k=0; k<vector.length; k++)
           if ( (x+k>=datos.length) || (y+k>=datos[0].length)
               || (datos[x+k][y+k]!=vector[k]) )
              encontrado = false;
    }


    if (!encontrado) {
       encontrado = true;

       for (k=0; k<vector.length; k++)
            if ( (x+k>=datos.length) || (y-k<0)
               || (datos[x+k][y-k]!= vector[k]) )
              encontrado = false;
    }


    if (!encontrado) {
       encontrado = true;

       for (k=0; k<vector.length; k++)
           if ( (x-k<0) || (y+k>=datos[0].length)
               || (datos[x-k][y+k]!=vector[k]) )
              encontrado = false;
    }


    if (!encontrado) {
       encontrado = true;

       for (k=0; k<vector.length; k++)
            if ( (x-k<0) || (y-k<0)
               || (datos[x-k][y-k]!=vector[k]) )
              encontrado = false;
     }


     return encontrado;
}
```

EJERCICIO 3

```
public boolean f (int a, int b)
{
  if (a==b)
     return g(a);
  else
     return g(a) && f(a+1,b);
}

private boolean g (int x)
{
  int i,j;

  for (i=0; i<datos.length; i++)
      for (j=0; j<datos[i].length; j++)
          if (datos[i][j] == x)
             return true;

  return false;
}
```

Versión iterativa

```
  public boolean fwhile (int a, int b)
  {
       boolean resultado = true;

       while (a<=b) {
            resultado = resultado && g(a);
            a++;
       }

       return resultado;
  }

  public boolean ffor (int a, int b)
  {
       int x;
       boolean resultado = true;

       for (x=a; (x<=b) && resultado; x++)
            resultado = g(x);

       return resultado;
  }
```