**MANUAL TÉCNICO**

**CALCULADORA MATRICES**

**POR:** JUAN FELIPE LÓPEZ VERGARA

**MATERIA:** Lógica III

**PROFESOR**: Roberto Flórez

**FECHA:** 06 de febrero de 2015

UNIVERSIDAD DE ANTIOQUIA

DEPARTAMENTO INGENIERÍA DE SISTEMAS

MEDELLÍN

2015

Contenido

[1. INTRODUCCIÓN 4](#_Toc411131935)

[2. ALCANCE 4](#_Toc411131936)

[3. OBJETIVOS 5](#_Toc411131937)

[3.2. Objetivo general 5](#_Toc411131938)

[3.3. Objetivos específicos 5](#_Toc411131939)

[4. CONTENIDO 5](#_Toc411131940)

[4.1 . Definición del problema 5](#_Toc411131941)

[4.1.1. Análisis del problema 7](#_Toc411131942)

[1. Solucionar con Cramer 7](#_Toc411131943)

[2. Solucionar con Gauss-Jordan 7](#_Toc411131944)

[3. Mostrar la matriz original todo el tiempo 8](#_Toc411131945)

[4.2.1. Clase AnalizadorTextos.java 8](#_Toc411131946)

[4.2.2. Clase NodoDoble 8](#_Toc411131947)

[4.2.2.1. Variables globales 9](#_Toc411131948)

[ Object dato 9](#_Toc411131949)

[ NodoD ld 9](#_Toc411131950)

[ NodoD li 9](#_Toc411131951)

[4.2.2.2. Contructores 9](#_Toc411131952)

[ Constructor con dato 9](#_Toc411131953)

[4.2.2.3. Asignaciones y retornos 9](#_Toc411131954)

[ asignaDato 9](#_Toc411131955)

[ retornaDato 9](#_Toc411131956)

[ asignaLd 10](#_Toc411131957)

[ asignaLi 10](#_Toc411131958)

[ retornaLd 10](#_Toc411131959)

[ retornaAnterior 10](#_Toc411131960)

[4.2.3. Clase Tripleta 10](#_Toc411131961)

[4.2.3.1. Variables 10](#_Toc411131962)

[ Variables de tipo Int 10](#_Toc411131963)

[ Variables de tipo Object: 11](#_Toc411131964)

[4.2.3.2. Métodos 11](#_Toc411131965)

[ Métodos de asignación 11](#_Toc411131966)

[ Métodos de retorno 11](#_Toc411131967)

[ Constructor 11](#_Toc411131968)

[4.2.4. Clase MatrizEnForma1 11](#_Toc411131969)

[4.2.4.1. Variables globales 12](#_Toc411131970)

[ Variables para almacenar datos: 12](#_Toc411131971)

[4.2.4.2. Constructor: 12](#_Toc411131972)

[4.2.4.3. Métodos 12](#_Toc411131973)

[ Método nodoCabeza 12](#_Toc411131974)

[ Método primerNodo 12](#_Toc411131975)

[ Método esVacia 13](#_Toc411131976)

[ Método finDeRecorrido 13](#_Toc411131977)

[ Método mostrarMatriz 13](#_Toc411131978)

[ Método construyeNodosCabeza 14](#_Toc411131979)

[ Método conectaPorFilas 14](#_Toc411131980)

[ Método conectaPorColumnas 15](#_Toc411131981)

[ Método intercambiaColumnas 16](#_Toc411131982)

[ Método intercambiaFilas 17](#_Toc411131983)

[ Método retornaValorTripleta 18](#_Toc411131984)

[ Método buscarFila 18](#_Toc411131985)

[ Método buscarFilaColumna 19](#_Toc411131986)

[ Método sacarDeterminante 20](#_Toc411131987)

[ Método copiarMatriz 21](#_Toc411131988)

[ Método cambiaFIlasPivoteCero 22](#_Toc411131989)

[ Método igualdadPorColumnas 23](#_Toc411131990)

[ Método multiplicidadPorFilas 25](#_Toc411131991)

[ Método casosEspeciales 27](#_Toc411131992)

[ Método cramer 28](#_Toc411131993)

[ Método gaussJordan 30](#_Toc411131994)

[4.2.5. Clase Ventana 32](#_Toc411131995)

[4.2.5.1. Variables globales 32](#_Toc411131996)

[ Variables para almacenar datos: 32](#_Toc411131997)

[4.2.5.2. Constructor: 32](#_Toc411131998)

[1.2.1.1. Métodos 33](#_Toc411131999)

[ Método botonIngresarMatrizActionPerformed: 33](#_Toc411132000)

[ Método botonGaussActionPerformed: 33](#_Toc411132001)

[ Método botonCramerActionPerformed: 34](#_Toc411132002)

[ Método botonLimpiarActionPerformed: 34](#_Toc411132003)

[2. CONCLUSIONES 35](#_Toc411132004)

[3. LISTA DE ILUSTRACIONES 35](#_Toc411132005)

# INTRODUCCIÓN

La finalidad de este manual técnico es proporcionar al lector la lógica con la que fue desarrollada la aplicación. Este manual no pretende ser guía de aprendizaje para el lector, sino documentar el proceso de desarrollo de la aplicación y proceso para la solución del problema general presentado para la creación del sistema.

# ALCANCE

Este manual va dirigido a un público con conocimientos técnicos sobre la programación orientada a objetos, en él se describen las clases usadas para la solución del problema y los métodos con su respectivo funcionamiento, el lenguaje bajo el que fue creado es java, usando la interfaz Netbeans IDE 8.0.2.

# OBJETIVOS

## Objetivo general

Elaborar un programa que dé solución al problema presentado a continuación procurando continuamente que sea lo más amable posible con el usuario.

## Objetivos específicos

* Diseñar el programa con una interfaz gráfica sencilla y legible.
* Elaborar un algoritmo estructurado y con una lógica entendible para el público con conocimientos técnicos en el tema.

# CONTENIDO

# . **Definición del problema**

**DEFINICIÓN DEL PROBLEMA**

Desarrollar un programa que permita resolver un sistema de ecuaciones lineales (tres variables, tres incógnitas). Para ello el usuario debe ingresar las tres ecuaciones del sistema, las cuales deben ser representadas mediante una matriz. Además la aplicación debe contar con dos opciones: una asociada al método de Gauss-Jordan y la otra al método Cramer para hallar el valor correspondiente de cada variable.

**I. Método de Gauss-Jordan**

1. Para resolver sistemas de ecuaciones lineales aplicando este método, se debe en primer lugar anotar los coeficientes de las variables del [sistema](http://www.monografias.com/trabajos11/teosis/teosis.shtml) de ecuaciones lineales en su notación matricial:

|  |  |  |
| --- | --- | --- |
| a1x + b1y + c1z = d1  a2x + b2y + c2z = d2  a3x + b3y + c3z = d3 |  |  |

|  |  |
| --- | --- |
| 2. Una vez hecho esto, utilizando las operaciones elementales se procede a convertir dicha matriz en una matriz [identidad](http://www.monografias.com/trabajos14/cambcult/cambcult.shtml): | 3. En donde la solución para el sistema de ecuaciones es:  X = d’1  Y = d’2  Z = d’3 |

**Operaciones Elementales**

a) Multiplicar una fila por un escalar:

ʎR => R

b) Multiplicar una fila por un escalar y sumarla a otra fila:

ʎR1 + R2 => R2

c) Intercambiar filas

R2 ⬄ R3

**II. Método Cramer**

El método de Cramer sirve para resolver sistemas de ecuaciones lineales, en donde se cumplan las dos condiciones siguientes:

* El número de ecuaciones es igual al número de incógnitas.
* El determinante de la matriz de los coeficientes es distinto de cero.

Dado el sistema de ecuaciones:

|  |  |  |
| --- | --- | --- |
| a1x + b1y + c1z = d1  a2x + b2y + c2z = d2  a3x + b3y + c3z = d3 |  |  |

El procedimiento que se debe realizar para resolver dicho sistema por este método es:

1. Primero se debe hallar el determinante de la matriz de los coeficientes.

|  |  |
| --- | --- |
|  | Sea Δ el determinante de la matriz de coeficientes |

2. Sustituir la columna de los términos independientes en cada una de las columnas de la matriz de coeficientes y hallar su respectivo determinante:

En donde:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Luego, la solución para el sistema de ecuaciones es:

|  |  |  |
| --- | --- | --- |
|  |  |  |

### 4.1.1. Análisis del problema

Se requiere leer y almacenar un sistema de ecuaciones en una matriz representada como lista ligada en forma 1, por ende se requiere definir además la clase tripleta, dado que el dato que se almacenará en cada nodo es un conjunto de tres datos dos enteros que representan la fila y la columna para la ubicación en la matriz y un double para el valor que se encuentra en dicha posición.

1. Solucionar con Cramer**:** se debe crear una copia de la matriz ingresada por el usuario y trabajar sobre esta, para resolver por este método se deben resolver las determinantes de cada matriz de forma individual y mostrar sus resultados haciendo uso de un área de resultado e internamente haciendo uso de una variable de tipo String. Se debe tener en cuenta que hay casos especiales en los que el sistema es inconsistente y no tiene solución o tiene infinitas soluciones, en estos casos simplemente se muestra en el área de resultado un mensaje en el que se informa cuál es el motivo de dicha inconsistencia en caso de existir.

En caso de no presentarse alguna excepción simplemente se muestra en el área de resultado cada una de las matrices a las cuales se les va a sacar el determinante, luego de cada matriz se muestra su respectivo determinante y finalmente el resultado correspondiente a cada variable.

1. Solucionar con Gauss-Jordan**:** se crea nuevamente una copia de la matriz ingresada por el usuario y sobre esta se realizan todas las funciones necesarias para reducir hasta llegar a la matriz identidad haciendo uso de las operaciones básicas de suma, multiplicación por un escalar y cambio de filas en caso de ser necesario. Es importante tener en cuenta los casos excepción como son las soluciones infinitas y las inconsistencias, en estos casos simplemente se muestra en el área de resultado un mensaje en el que se informa cuál es el motivo de dicha inconsistencia en caso de existir.

En el área de resultado se muestra el paso a paso para llegar a la solución, para ello se usa internamente un String en el que se almacena absolutamente todo y finalmente se retorna para imprimirlo, el resultado de cada una de las variables se muestra en la parte final del campo de resultado.

1. Mostrar la matriz original todo el tiempo**:** Basta con imprimir la matriz ingresada por el usuario en cada ocasión en el área de resultado. Clases

### Clase AnalizadorTextos.java

Esta es la clase principal del programa la cual crea la ventana.

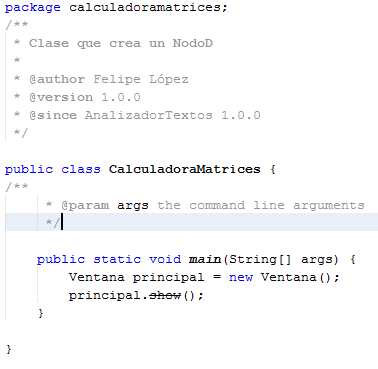


Ilustración 1. Clase CalculadoraMatrices .java

### Clase NodoDoble

Esta es la clase que contiene toda la configuración del tipo de nodo que se va a usar dentro de la lista que es el nodo doble, lo que significa que contiene un apuntador al siguiente nodo y un apuntador al anterior, además de una objeto y hay un constructor especial que permite además añadir al nodo un contador usado para contar la repetición de las palabras.

#### Variables globales

* Object dato**:** contiene la palabra que será almacenada en el nodo.
* NodoD ld**:** se usa para determinar el siguiente nodo.
* NodoD li**:** se usa para determinar el anterior nodo.

#### Contructores

* Constructor con dato**:** permite crear un nodo, asignarle una dato y sus campos de **ld** y **li** se asignan nulos.

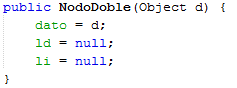


Ilustración 2. Constructor con dato

#### Asignaciones y retornos

Se usan para asignar y retornar cada una de las variables al nodo y se muestran a continuación.

* asignaDato**:** este método asigna un dato al nodo, recibe como parámetro de entrada un Object dato y lo asigna a la variable global dato, no retorna nada.



Ilustración 3. Método asignaDato

* retornaDato**:** este método no recibe ningún parámetro de entrada y retorna el dato que contiene el nodo.



Ilustración 4. Método retornaDato

* asignaLd**:** recibe un NodoDoble **ld**, lo asigna a la variable global **ld** y no retorna nada.



Ilustración 5. Método asignaLd

* asignaLi**:** recibe un parámetro de entrada de tipo NodoDoble **li** y lo asigna a la variable **li**, no retorna nada.



Ilustración 6. Método asignaLi

* retornaLd**:** método que retorna el valor de la variable global **ld**, no recibe parámetro de entrada.

****

Ilustración 7. Método retornaLd

* retornaAnterior**:** método que retorna el valor de la variable global **li**, no recibe parámetro de entrada.



Ilustración 8. Método retornaLi

### Clase Tripleta

Esta clase contiene todas las funcionalidades y métodos que hacen posible realizar las modificaciones requeridas por el programa, a continuación se describen las variables, métodos y demás.

#### Variables

* Variables de tipo Int**:** las variables de tipo int son: fila, columna, se usan para saber la fila y la columna de un trío de datos correspondientes a una celda de una matriz.
* Variables de tipo Object: la única variable de este tipo es la variable valor, la cual sirve para almacenar el apuntador a un nodo o un dato almacenado en una celda de la matriz ingresada por el usuario.

#### Métodos

* Métodos de asignación**:** sólo hay un método de asignación y es el método de asigna valor, el cual recibe un parámetro de entrada de tipo Object **v**  y los asigna a la variable global **valor**.
* Métodos de retorno**:** no reciben ningún parámetro y retornan el valor la fila y la columna de un objeto de la clase.

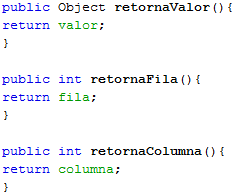


Ilustración 9. Métodos de retorno

* Constructor**:** recibe dos objetos de tipo Int para la fila y la columna y uno de tipo Object para el valor.

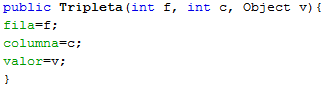


Ilustración 10. Constructor Tripleta.

### Clase MatrizEnForma1

Contiene los métodos principales de la aplicación, es aquí donde se realizan todas las operaciones sobre la matriz ingresada por el usuario, cabe anotar que no se trabaja sobre la original, sino sobre una copia que se crea con antelación para evitar la sobre escritura de los datos.

#### Variables globales

* Variables para almacenar datos:Se usan dos objetos de tipo Int que para almacenar el tamaño de la matriz y un objeto de tipo NodoDoble para identificar la cabeza de las listas.



Ilustración 11. Variables globales

#### Constructor:

Recibe dos objetos de tipo Int y crea un objeto de la clase tripleta en el que los almacena para luego asignarlos como valor al nodo global llamado **mat**

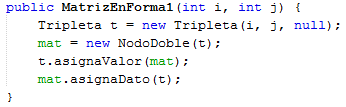
.

Ilustración 12. Constructor MatrizForma1

#### Métodos

* Método nodoCabeza**:** no tiene parámetro de entrada y retorna el nodo **mat** o nodo cabeza de todas las listas.



Ilustración 13. Método nodoCabeza

* Método primerNodo**:** No tiene parámetro de entrada y retorna el nodo siguiente al nodo cabeza, es decir el nodo cabeza de la primera fila y la primera columna..

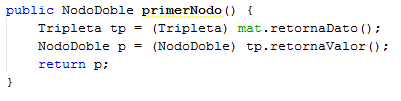


Ilustración 14. Método primerNodo

* Método esVacia**:** no recibe parámetro de entrada crea un NodoDoble p y lo hace igual al primero, luego retorna el nodo **mat**.

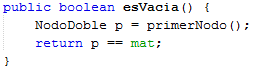


Ilustración . Método esVacia

* Método finDeRecorrido**:** recibe un NodoDoble de entrada y retorna el nodo **mat**.



Ilustración 16. Método finDeRecorrido

* Método mostrarMatriz**:** retorna un objeto de tipo String, no recibe ningún parámetro de entrada, básicamente hace uso de un nodo, una tripleta para retornar el valor de dicho nodo, y un string para almacenar el resultado de la operación la cual consiste en hacer un recorrido de toda la matriz y retornar los valores de cada nodo para irlos almacenando en el String y finalmente retornarlo.

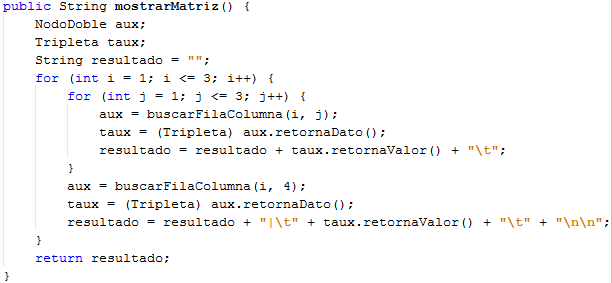


Ilustración 17. Método mostrarMatriz

* Método construyeNodosCabeza**:** no recibe parámetro de entrada y tampoco tiene ningún tipo de retorno, haciendo uso de dos variables de tipo NodoDoble se encarga de crear los nodos cabeza para cada una de las filas y columnas de la matriz, cada nodo cabeza es cabeza de su respectiva fila y columna y se verifica si hay más filas o columnas para poder generar los nodos necesarios. Cada que se crea un nodo cabeza se conecta con el siguiente por medio de su campo de dato.

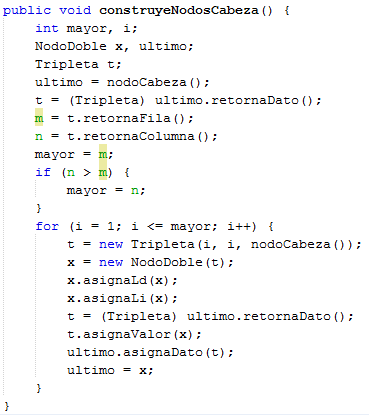


Ilustración 18. Método construyeNodosCabeza

* Método conectaPorFilas**:** recibe un objeto de tipo NodoDoble y no tiene retorno, se encarga de conectar de forma ordenada los nodos por medio de su campo de fila o liga derecha

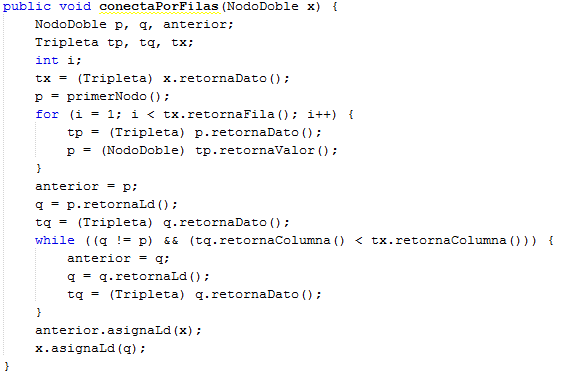


Ilustración 19. Método conectaPorFilas

* Método conectaPorColumnas**:** Hace lo mismo que el método anterior, sólo que conecta los nodos haciendo uso de su campo de liga izquierda.

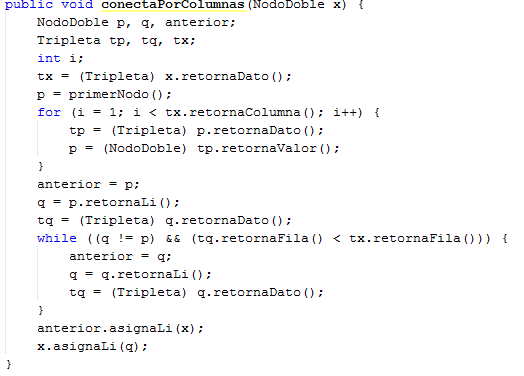
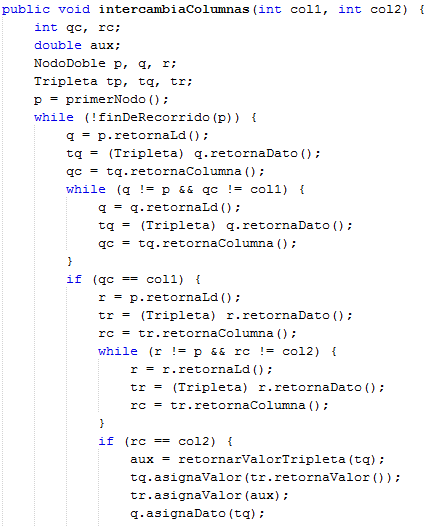


Ilustración 20. Método conectaPorColumnas

* Método intercambiaColumnas**:** recibe don enteros y no retorna nada, haciendo uso de tres nodos y las tripletas correspondientes recorre toda la lista primero comparando los valores del campo de columna usando las tripletas contenidas en cada nodo visitado y luego de encontrar las columnas buscadas se intercambian los datos entre los nodos correspondientes a cada una de estas.



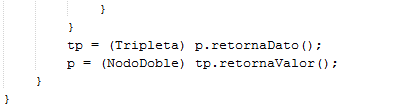


Ilustración 21. Método intercambiaColumnas

* Método intercambiaFilas**:** no retorna nada, recibe dos enteros que corresponden a las filas que se deben cambiar, y haciendo uso de 4 nodos primero busca las filas y luego las recorre por columnas intercambiando los valores de cada nodo recorrido, esto se hace de forma simultánea entre ambas filas hasta finalizar el recorrido.

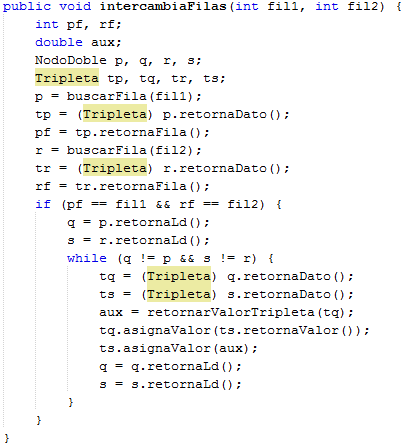


Ilustración 22. Método intercambiaFilas

* Método retornaValorTripleta**:** retorna un objeto de tipo double y recibe como parámetro de entrada una tripleta, lo que hace es convertir el valor almacenado en la tripleta de ingreso en un doublé para poder realizar operaciones con él.

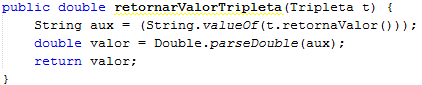


Ilustración 23. Método retornaValorTripleta

* Método buscarFila**:** retorna un objeto de tipo NodoDoble y recibe un entero como parámetro de entrada. Lo que hace es avanzar por la lista de nodos cabeza y retornar de cada nodo el valor de su fila haciendo uso como siempre de una tripleta y un entero, básicamente posiciona el nodo en el lugar correspondiente a la fila de ingreso.

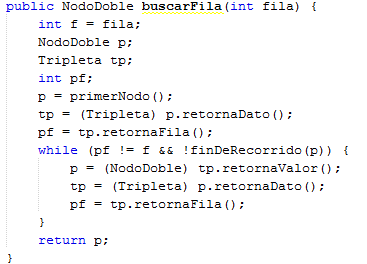


Ilustración 24. Método buscarFila

* Método buscarFilaColumna**:** retorna un objeto de la clase NodoDoble y recibe dos enteros que representan la fila y columna a buscar respectivamente, es uno de los métodos más utilizados en la aplicación, lo que retorna es la posición de un nodo en concreto dentro de la matriz en forma 1, esto lo hace haciendo uso del método buscarFila y cuando encuentra la fila simplemente recorre todas las columnas retornando de cada nodo dicho valor hasta encontrarlo.

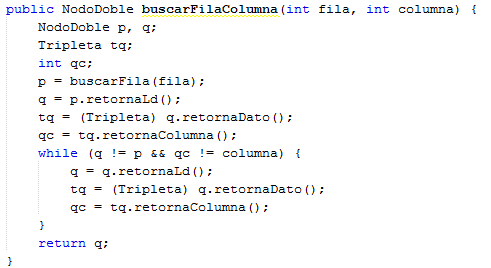
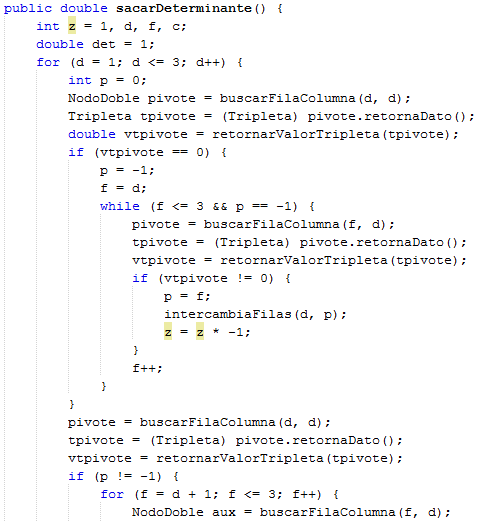


Ilustración 25. Método buscarFilaColumna

* Método sacarDeterminante**:** retorna un objeto de tipo double y no tiene parámetro de entrada. Este método es la base del trabajo con cramer dado que resuelve las determinantes de cada matriz, en primera instancia selecciona los pivotes y verifica que no sean cero, en caso de serlo hace un intercambio de filas, luego por medio de operaciones elementales hace una reducción en la parte inferior de la matriz convirtiéndola en una diagonal superior y finalmente multiplica todos los valores de la diagonal principal para obtener el determinante.



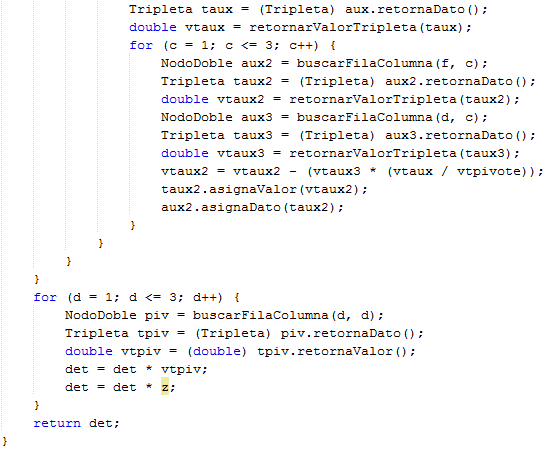


Ilustración 26. Método sacarDeterminante

* Método copiarMatriz**:** retorna una MatrizEnForma1 y no recibe nada, como su nombre lo indica crea una nueva MatrizEnForma1 y por medio del recorrido de la matriz de base simplemente lleva la info de cada nodo visitado a un nuevo nodo que se conecta por filas y columnas a la nueva matriz.

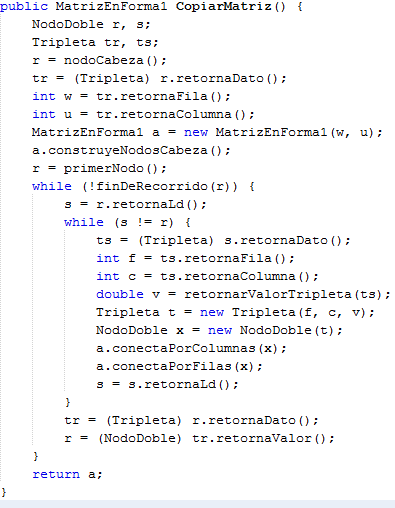


Ilustración 27. Método copiarMatriz

* Método cambiaFIlasPivoteCero**:** retorna un String con el resultado de cambiar las filas que tienen pivote cero para así evitar errores a la hora de realizar las operaciones elementales sobre la matriz.

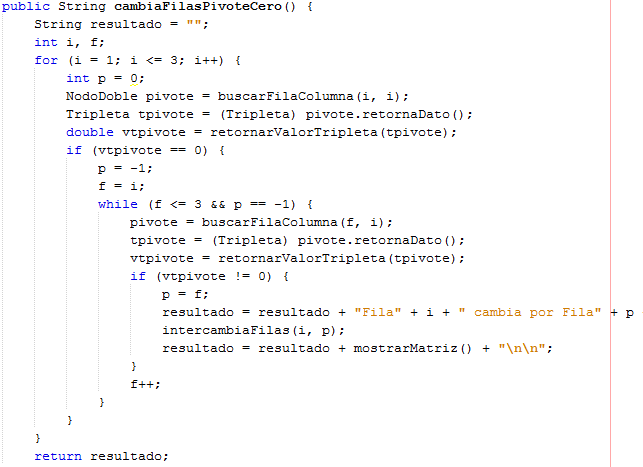


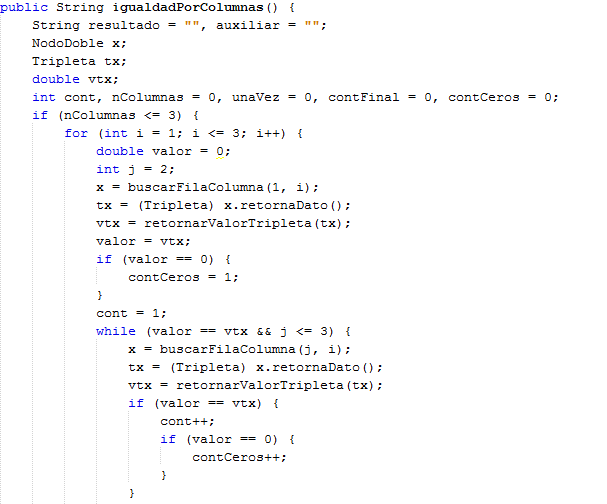
Ilustración 28. Método cambiaFilasPivoteCero

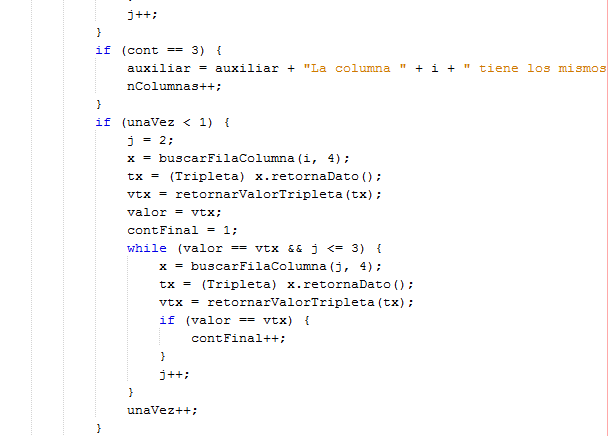
* Método igualdadPorColumnas**:** no recibe nada, retorna un String con el resultado de verificar si hay columnas que en todas sus celdas tengan el mismo valor, puede retornar tres valores distintos dependiendo del resultado obtenido se usan varios contadores:
* nColumnas: cuenta el número de columnas que tienen el mismo valor en todas sus celdas
* contFinal: aumenta cada vez que se encuentra repetido el valor de la primera celda de la columna de resultado.
* contCeros: aumenta cada vez que se encuentra un cero en una columna x que se esté verificando.

Si cuando se acaba de hacer los recorridos la columna del resultado tiene el mismo valor en todas sus celdas y además hay dos o más columnas dentro de la matriz de coeficientes que tienen el mismo dato en todas sus celdas, entonces el método retorna soluciones infinitas.

Si cuando se acaba de hacer los recorridos la columna del resultado tiene diferentes valores en todas sus celdas y además hay dos o más columnas dentro de la matriz de coeficientes que tienen el mismo dato en todas sus celdas, entonces el método retorna soluciones inconsistencia.

Si cuando se acaba de hacer los recorridos hay una columna con puros ceros y no hay más columnas con valores iguales en todas sus celdas, así la del resultado tenga los mismos valores retornará inconsistencia.





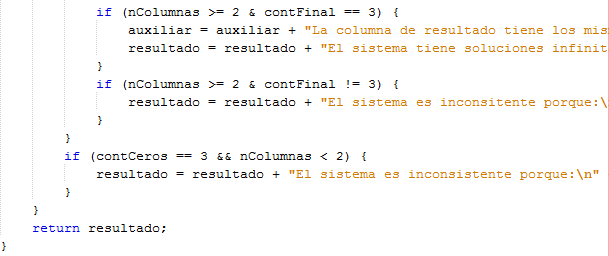
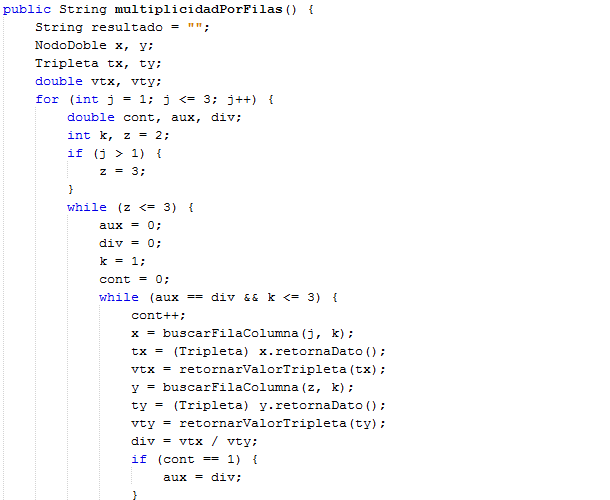


Ilustración 29. Método igualdadPorColumnas

* Método multiplicidadPorFilas**:** retorna un String con el resultado de verificar si hay filas que son múltiplos de otras, no recibe nada. Simplemente hace un recorrido por filas y dentro de filas por columnas, retorna los valores almacenados en cada nodo y los divide con el correspondiente de su misma columna de la otra fila, si al final todos son iguales incluso el de la columna de resultado, entonces retorna soluciones infinitas, pero si el de la columna de resultado no es igual retorna inconsistencia.



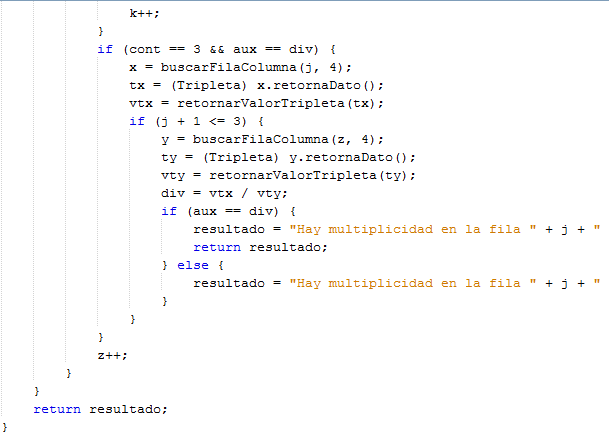


Ilustración 30. Método multiplicidadPorFilas

* Método casosEspeciales**:** no recibe nada, retorna un String con el resultado de verificar primeramente si hay filas que sólo tienen ceros, si toda la fila tiene ceros retorna soluciones infinitas, si todas menos la última columna de la fila tienen ceros, entonces retorna inconsistencia. Luego de hacer esta verificación hace un llamado a los métodos multiplicidadPorFIla e igualdadPorColumnas y finalmente retorna el resultado de las operaciones.

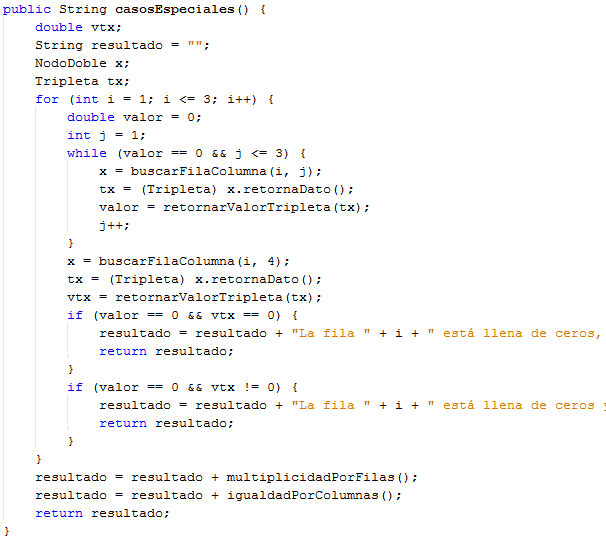
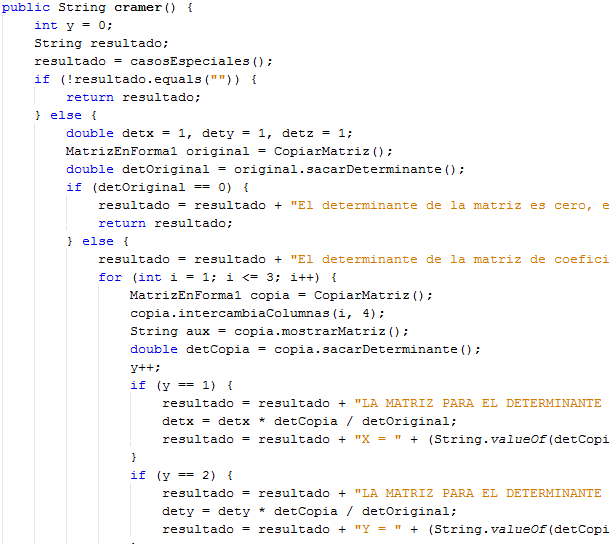


Ilustración 31. Método casosEspeciales

* Método cramer**:** retorna un String con el resultado y no recibe nada, primeramente verifica los casos especiales, si se cumple alguno simplemente retorna, sino hace el determinante de la matriz original y luego intercambia en una matriz copia la columna 4 con cada una de las otras columnas y le aplica el método para sacar el determinante, luego divide el determinante de cada matriz por el determinante de la original, agrega el resultado al string y finalmente retorna.



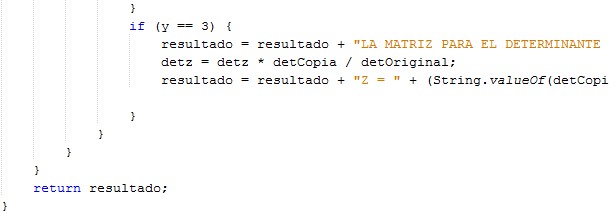
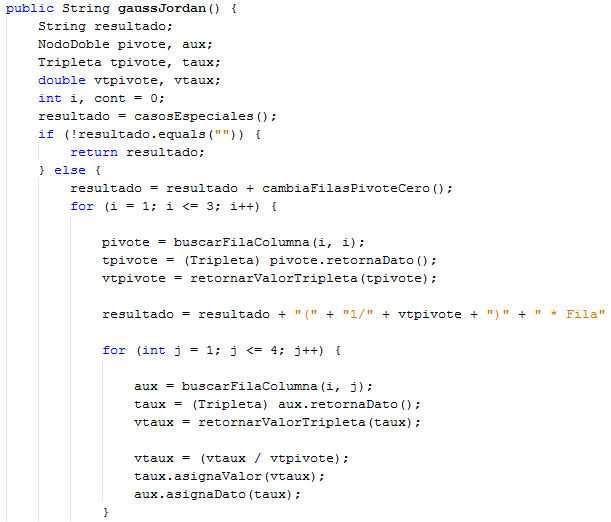
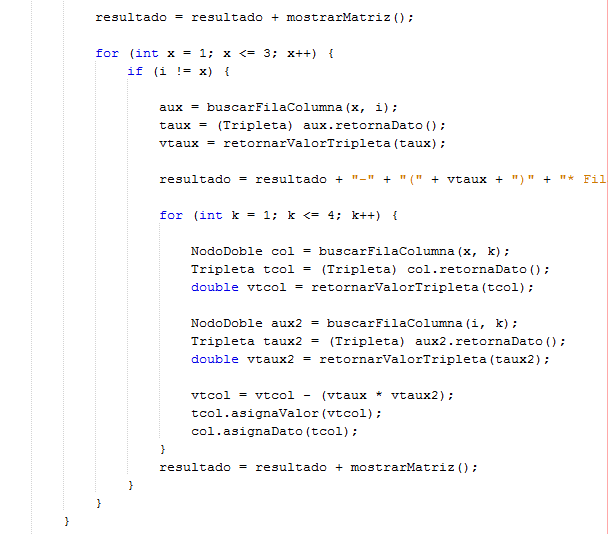


Ilustración 32. Método cramer

* Método gaussJordan**:** no recibe nada y retorna un String con el resultado, lo primero que hace es verificar los casos especiales y si no se cumplen comienza por escoger el primer pivote y multiplica toda la fila por su inverso aditivo, luego hace ceros todos los valores de esa columna multiplicando el pivote de ella por el negativo de la fila de la cual se quiere eliminar dicho valor y se suma luego a dicha fila y así sucesivamente hasta realizar toda la reducción. Finalmente, termina por asignar los valores respectivos a las variables y luego retorna.





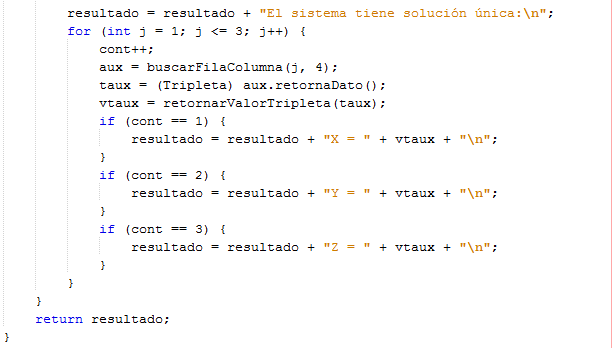


Ilustración 33. Método gaussJordan

### Clase Ventana

Contiene todos los métodos que relacionan la interfaz de usuario o formulario con las operaciones que se deben realizar sobre el sistema de ecuaciones.

#### Variables globales

* Variables para almacenar datos:Se usan dos objetos de tipo MatrizEnForma1 para almacenar los datos ingresados por el usuario.

#### Constructor:

Inicializa las componentes de la ventana, la centra en la pantalla, le da el título y deshabilita el redimensionamiento de la misma.

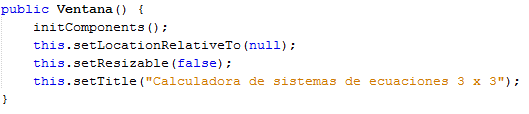


Ilustración 34. Constructor clase Ventana

#### Métodos

* Método botonIngresarMatrizActionPerformed:este método crea la matriz en lista ligada forma uno y la muestra en el área de resultado. Además muestra un mensaje de error cuando el usuario ingresa mal los datos. No retorna nada, recibe una variable de tipo java.awt.event.ActionEvent.

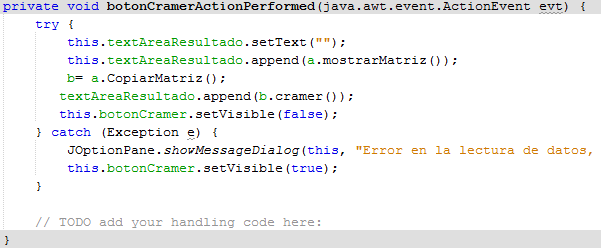


Ilustración 35. Método botonIngresar

* Método botonGaussActionPerformed:este método limpia el área de resultado y muestra en ella la operación gaussJordan ejecutada sobre una matriz copia de la matriz ingresada por el usuario. No retorna nada, recibe una variable de tipo java.awt.event.ActionEvent.

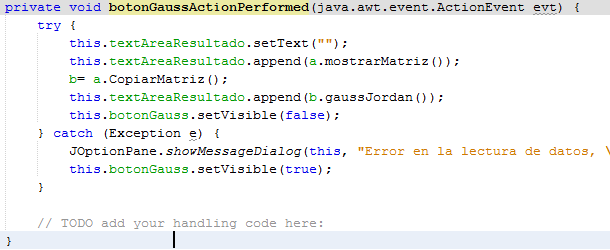


Ilustración 36. Método botonGauss

* Método botonCramerActionPerformed:este método limpia el área de resultado y muestra en ella la operación cramer ejecutada sobre una matriz copia de la matriz ingresada por el usuario. No retorna nada, recibe una variable de tipo java.awt.event.ActionEvent.

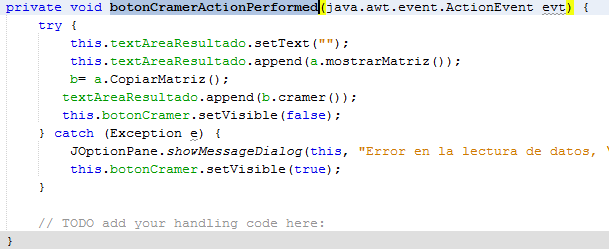


Ilustración 37. Método botonCramer

* Método botonLimpiarActionPerformed:este método limpia el área de resultado, todas las celdas de la tabla de ingreso de datos y activa nuevamente los botones de operaciones. No retorna nada, recibe una variable de tipo java.awt.event.ActionEvent.

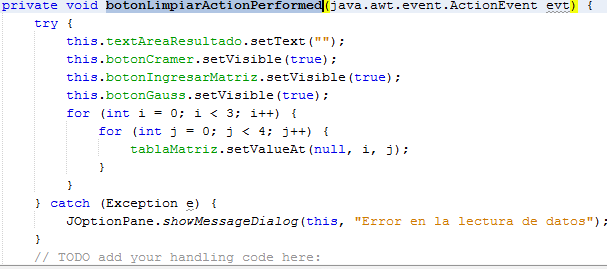


Ilustración 38. Método botonLimpiar

# CONCLUSIONES

A la hora de hacer uso de información ingresada por el usuario es muy importante hacer uso del try catch para evitar las excepciones, por otra parte el trato de los datos de entrada es de suma importancia, se debe tener en cuenta que tipo de datos se le permite ingresar al usuario, si los métodos son muy largos se pueden dividir en varios e incluso hacer su llamado dentro de varios métodos que así lo puedan requerir.

# LISTA DE ILUSTRACIONES

[Ilustración 1. Clase CalculadoraMatrices .java 8](#_Toc411131893)

[Ilustración 2. Constructor con dato 9](#_Toc411131894)

[Ilustración 3. Método asignaDato 9](#_Toc411131895)

[Ilustración 4. Método retornaDato 9](#_Toc411131896)

[Ilustración 5. Método asignaLd 9](#_Toc411131897)

[Ilustración 6. Método asignaLi 9](#_Toc411131898)

[Ilustración 7. Método retornaLd 10](#_Toc411131899)

[Ilustración 8. Método retornaLi 10](#_Toc411131900)

[Ilustración 9. Métodos de retorno 11](#_Toc411131901)

[Ilustración 10. Constructor Tripleta. 11](#_Toc411131902)

[Ilustración 11. Variables globales 11](#_Toc411131903)

[Ilustración 12. Constructor MatrizForma1 12](#_Toc411131904)

[Ilustración 13. Método nodoCabeza 12](#_Toc411131905)

[Ilustración 14. Método primerNodo 12](#_Toc411131906)

[Ilustración 15. Método esVacia 12](#_Toc411131907)

[Ilustración 16. Método finDeRecorrido 13](#_Toc411131908)

[Ilustración 17. Método mostrarMatriz 13](#_Toc411131909)

[Ilustración 18. Método construyeNodosCabeza 14](#_Toc411131910)

[Ilustración 19. Método conectaPorFilas 15](#_Toc411131911)

[Ilustración 20. Método conectaPorColumnas 16](#_Toc411131912)

[Ilustración 21. Método intercambiaColumnas 17](#_Toc411131913)

[Ilustración 22. Método intercambiaFilas 18](#_Toc411131914)

[Ilustración 23. Método retornaValorTripleta 18](#_Toc411131915)

[Ilustración 24. Método buscarFila 19](#_Toc411131916)

[Ilustración 25. Método buscarFilaColumna 20](#_Toc411131917)

[Ilustración 26. Método sacarDeterminante 21](#_Toc411131918)

[Ilustración 27. Método copiarMatriz 22](#_Toc411131919)

[Ilustración 28. Método cambiaFilasPivoteCero 23](#_Toc411131920)

[Ilustración 29. Método igualdadPorColumnas 25](#_Toc411131921)

[Ilustración 30. Método multiplicidadPorFilas 27](#_Toc411131922)

[Ilustración 31. Método casosEspeciales 28](#_Toc411131923)

[Ilustración 32. Método cramer 29](#_Toc411131924)

[Ilustración 33. Método gaussJordan 32](#_Toc411131925)

[Ilustración 34. Constructor clase Ventana 32](#_Toc411131926)

[Ilustración 35. Método botonIngresar 33](#_Toc411131927)

[Ilustración 36. Método botonGauss 33](#_Toc411131928)

[Ilustración 37. Método botonCramer 34](#_Toc411131929)

[Ilustración 38. Método botonLimpiar 34](#_Toc411131930)