Manual de Usuario

Almacenamiento y operaciones de matrices

Juan David Monroy

CONTENIDO

1. **Introducción………………………………………………….….3** 1. Objetivo………………………………………………….....3

2. Requerimientos……………………………………………..3

1. **Funcionamiento del aplicativo**
2. Ingreso de Información………………………………………(4-8)
3. Funcionalidades……………………………………………(6-7)
4. Clase Matriz……………………………………………….(8-10)
5. **INTRODUCCIÓN**
6. **Objetivo**

Crear una aplicación que permita realizar los siguientes ejercicios con matrices:

* 1. A + B
  2. A − B
  3. A · B
  4. A X B
  5. A Transpuesta
  6. (A)³

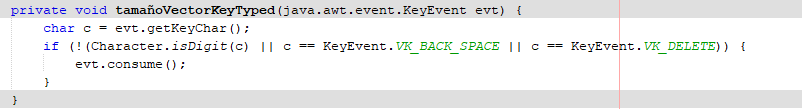
El sistema debe permitir al usuario escoger el ejercicio que desea resolver, definir los tamaños de las matrices necesarias y decidir si las matrices las llena el usuario o el sistema - se debe mostrar siempre las matrices iniciales y la matriz resultado.

1. **Requerimientos**

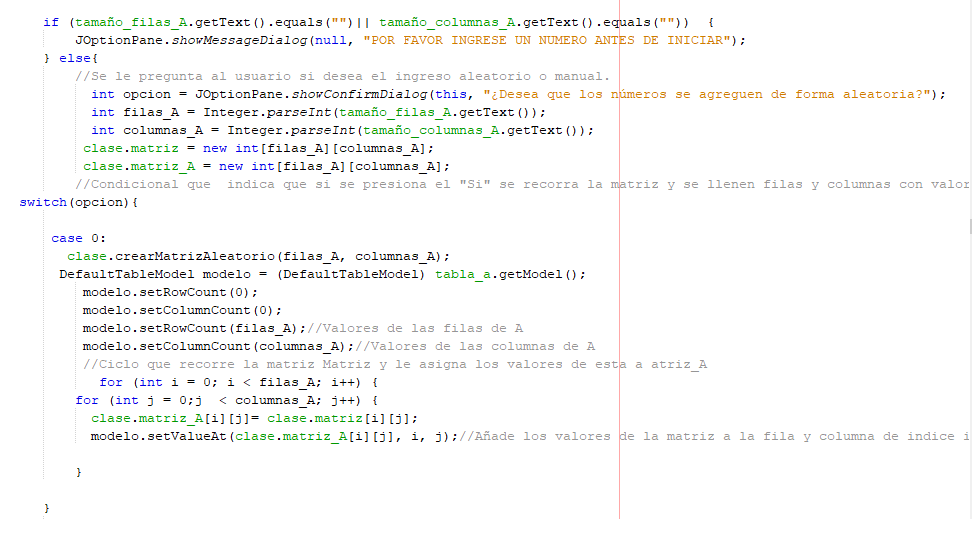
* Netbeans IDE 8.2
* Java.

1. **Funcionamiento del aplicativo**
2. **Ingreso de Información**

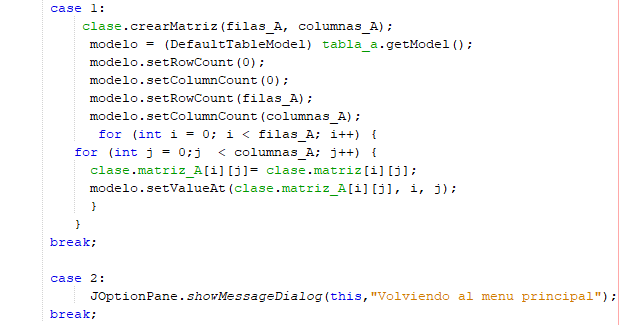
El ingreso de la información del usuario parte de que él ingrese el tamaño del arreglo por medio de un textfield, el dato ingresado es validado por medio de un condicional primero verificando que el campo de texto no esté vacío. El usuario no tiene posibilidad de ingresar caracteres no numéricos en este campo debido a la definición del siguiente método:



El cual apenas detecta el evento de al ser digitado un carácter e inmediatamente lo borra.

Después de validar que ingresó un número, por medio de un JOptionPane se le pregunta al usuario si desea el ingreso de valores aleatorios para llenar la matriz y por medio de un condicional se valida que si presionó que si (internamente es el valor numérico 0) se genere un nuevo objeto de la clase Random, se le solicite el ingreso del rango de los valores y se cree la matriz por medio del método crearMatrizAleatorio(),recibiendo como parámetros las filas y las columnas ingresadas por el usuario, posteriormente, medio de un ciclo for , se le asignan los valores de la matriz de la clase a una matriz local y se agregan la tabla de la GUI, por medio de un setValueAt().

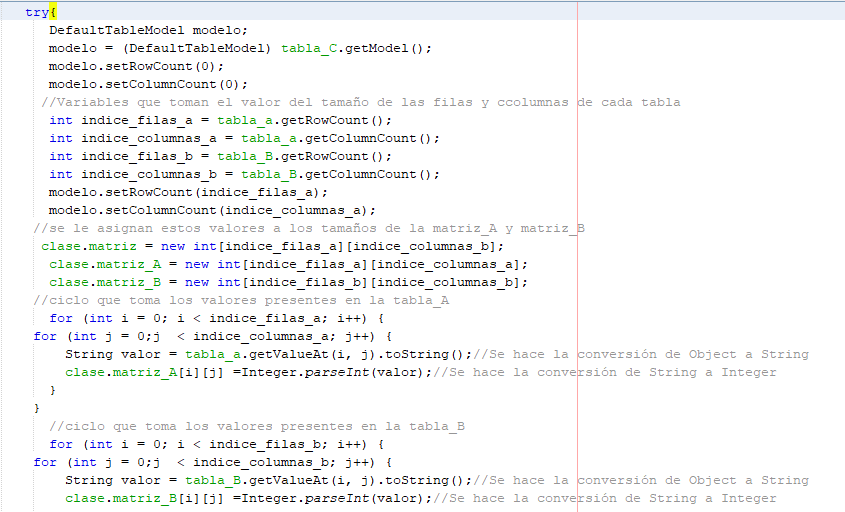
Al usuario presionar el botón no (internamente con el valor numérico 1) se ejecuta el método crearMatriz() que también recibe como parámetros las filas y las columnas, y se le procederá a pedir al usuario el ingreso de los datos. En el caso que el campo de texto quede vacío se le preguntará al usuario si desea que se le agregue el valor por defecto (valor 0) , y con el uso de un condicional de función parecida al anterior para decidir si el ingreso es aleatorio o no, se procede a llenar la posición con 0 o se vuelve a abrir el menú para que el usuario ingrese el número.

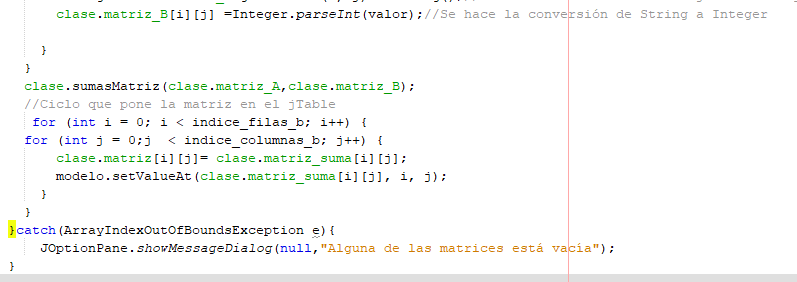


1. **Funcionalidades**

**Suma, Resta, Producto punto y Matriz Cubo de matrices:**

Para estos tres métodos se tomó el tamaño de cada fila y columna de las tablas donde se habían ingresado, se inicializaban las matrices públicas de la clase matriz con estas dimensiones. A continuación por medio de un GetValueAt() se tomaba cada objeto de la tabla, por medio de un ToString() se realizaba su conversión de objeto a texto y posteriormente con un Integer.Parseint() se realiza su conversión a entero, para añadirlos a la matriz y realizar el método pertinente dado el caso (sumaMatriz(), restasMatriz(), productoPuntoMatriz(), cuboMatriz()), tomando como parametros para . Finalmente se asignan los valores de esta matriz a otra y por un ciclo se van mostrando en la tabla por medio de un SetValueAt().

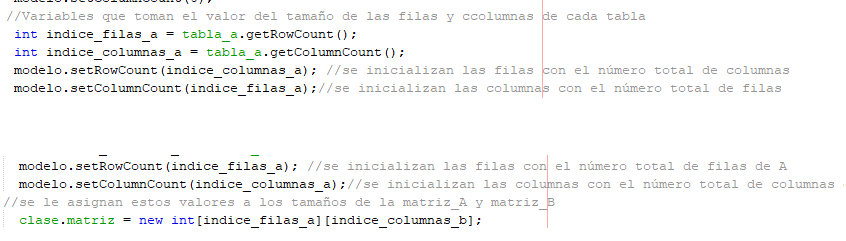




**Multiplicación de matriz y Transpuesta Matriz:**

Para este par de operaciones en la parte de la GUI se realiza una leve variación respecto a los métodos anteriores, en el caso de la transpuesta al iniciarse el proceso el valor dados para las filas y las columnas se invierten en el objeto tabla para evitar algún error de desbordamiento en este objeto. En el caso de la multiplicación de Matriz, el valor de las filas y columnas de la matriz resultado asignada será equivalente a las Filas de la Matriz A y las columnas de la Matriz B.

De forma similar a los métodos anteriores, se toman los valores de las tablas con un GetValueAt() y por medio de un ciclo for se llena la matriz para proceder a usar los métodos de la clase Matriz transpuestaMatriz() y multiplicacionMatriz().

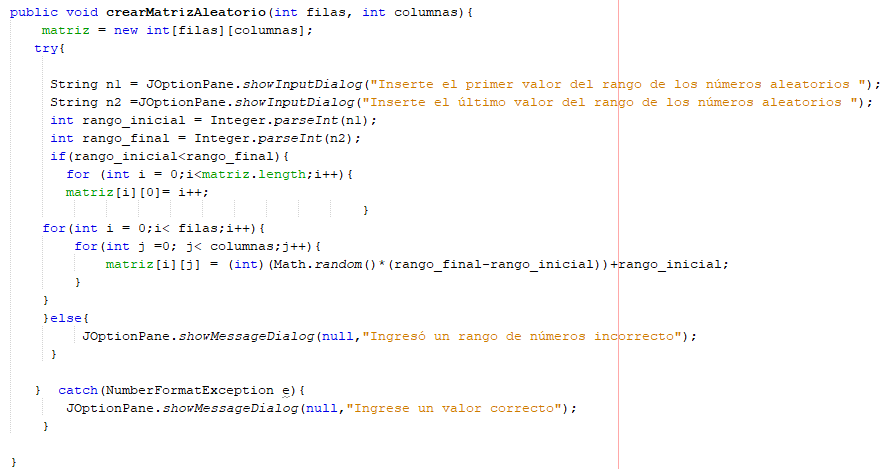


1. **Clase Matriz**

**Crear Matriz y Crear Matriz Aleatorio:**

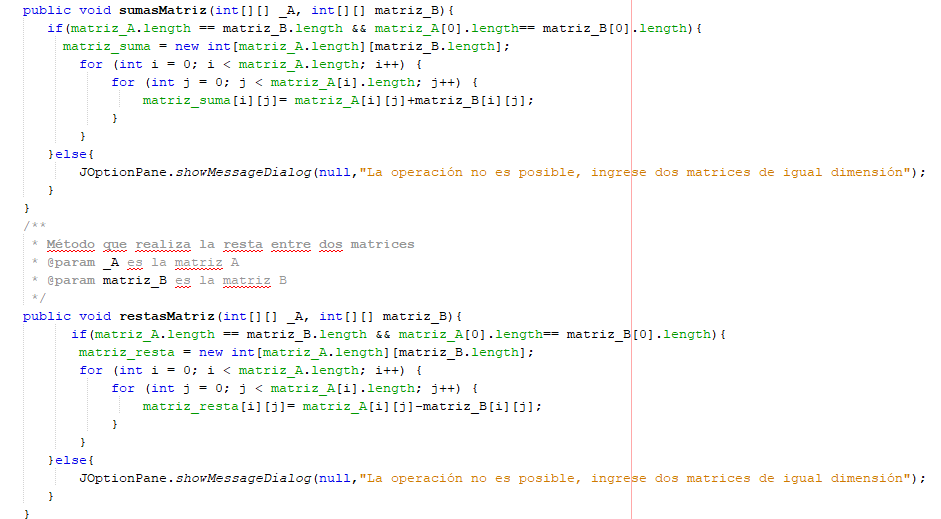
En estos métodos se utilizan como parámetros de entrada los valores enteros de filas y de columnas dados por el usuario , se le solicita el ingresos de los datos a llenar en la matriz desde aquí por medio de un JOptionPane, que al mismo tiempo valida en el caso de que se deje el espacio vacío al momento de ingresar el número, pregunte y valide con un switch si se llena este dato con un 0 o se regrese al menú principal. En el caso de crearMatrizAletorio() con la ayuda de la librería de Java Math y su función Random se asignan los números a la posición recorrida por la matriz, después de previamente haberle pedido al usuario el ingreso de los rangos, validando que el dato del primer valor del rango sea menor que el del segundo para continuar con la operación. En ambos métodos se inicializa y se recorre la matriz para llenarla por medio de un ciclo for.

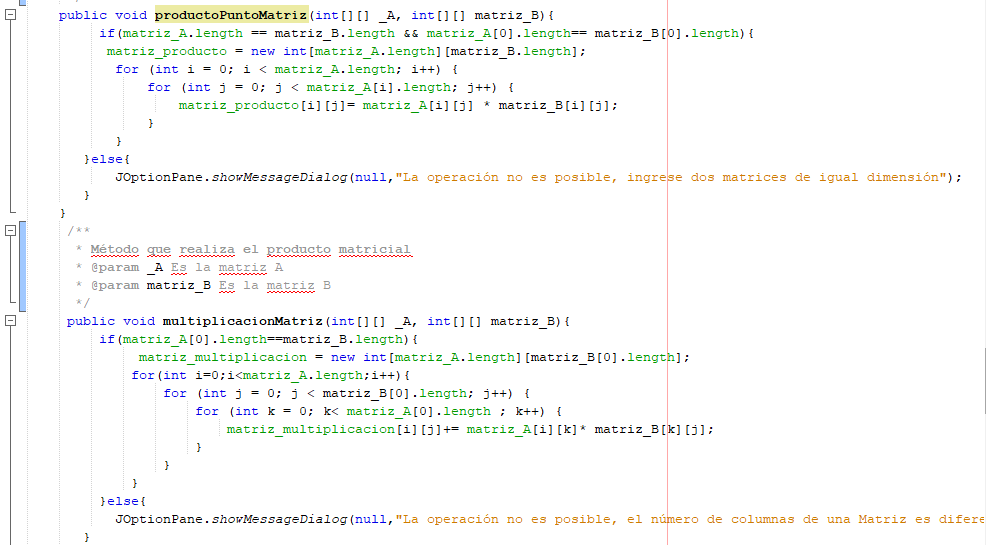




**Métodos Suma, Resta, Producto Punto y Multiplicación de matrices:**

Para estos 4 métodos se reciben como parámetros de entrada dos matrices , se inicializa una tercera matriz que será la que almacene los datos almacenados al recorrer cada elemento de las matrices e irlos operando. En el caso del método suma, resta y producto punto, se valida que el tamaño tanto de la matriz A como de la matriz B sean iguales para poderse realizar la operación, y en caso de la multiplicación de matrices se valida que el rango de las filas de la matriz A sea igual al rango del número de columnas de la matriz B.





**Métodos Cubo Matriz y Matriz Transpuesta:**

En estos dos métodos se recibe como parámetro de entrada la matriz, junto a su número de filas y columnas.

