|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Ing. Guadalupe Lizeth Parrales Romay |
| *Asignatura:* | Programación Orientada a Objetos |
| *Grupo:* | 04 |
| *No de Práctica(s):* | 3 |
| *Integrante(s):* | Dávila Ortega Jesús Eduardo - No. Cuenta: 317199860  Díaz Hernández Marcos Bryan - No. Cuenta: 317027253  Pareja Avila Emiliano - No. Cuenta: 317081345  Vazquez Zavala Oliver Alexis - No. Cuenta: 317202263 |
| *No. de Equipo de cómputo empleado:* |  |
| *No. de Lista o Brigada:* | 12,14, 30, 37 |
| *Semestre:* | 2021-1 |
| *Fecha de entrega:* | 24 de octubre del 2020 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_

**Objetivo de la práctica.**

Utilizar bibliotecas propias del lenguaje para realizar algunas tareas comunes y

recurrentes.

**Introducción.**

Para la realización de este reporte y de la práctica fue necesario el implementar una clase que nos pudiera dar las propiedades del sistema en el que estábamos trabajando, esta clase es Properties y se encuentra dentro de la clase util de Java. Esta clase tiene una estructura especial debido a que es una colección de tipo HashTable, que es especializada para las Strings, es decir que todos los valores de esa clase son cadenas y esto es debido a que las cadenas representan las distintas propiedades, además al ser un HashTable, se almacenan las propiedades mediante una llave y un valor, en este caso las llaves y los valores son Strings.

La clase permite el análisis del sistema, esto puede servir en caso de que sea un requisito el tener cierta propiedad en el sistema para que se pueda implementar cierta clase, y entonces Properties puede mostrar si el sistema cuenta con esa propiedad.

Como se mencionaba es una HashTable especializada que contiene las propiedades pero al ser un tipo de colección es posible realizar las operaciones correspondientes a una Hash, en este caso es necesario crear una instancia de la clase para que sea posible obtener las propiedades que asignamos y las que tiene nuestro sistema de Java por defecto, la declaración general es la que sigue:

* Properties props = new Properties();

Los métodos más utilizados de la clase son:

* getProperty(String llave): Devuelve el valor que indica la llave, sino existe retorna null
* propertyNames(): Regresa la enumeración de todas las llaves, incluyendo las llaves del default
* setProperty(String llave, String valor): Asignación de elemento como propiedad
* System.getProperties(): Regresa una tabla de propiedades que contiene las propiedades del sistema.

Como se menciona existe una HashTable para los defaults, esto es en caso de querer obtener una propiedad por medio de la llave y no encontrarla, se utiliza en caso de que se quiera asignar una propiedad a la HashTable que nosotros creamos y se tenga una segunda tabla que contenga a propiedades secundaria o no primordiales para lo que se desea implementar nuestra colección Properties.

Existen dos métodos importantes que son el System.getProperties(), y propertyNames(), ya que con estos se puede resolver el último ejercicio de la práctica.

System.getProperties se utiliza para poder obtener la tabla de las propiedades del sistema que se esté utilizando, esto es para tener la referencia de la tabla que contiene las propiedades, pero hay que recordar que es una HashTable y por eso es necesario utilizar el propertyNames(), ya que este utiliza el Enumeration que es una interfaz que permite iterar en las colecciones por medio de las llaves, entonces este nos ayuda a obtener todas las llaves y posteriormente, se envían una a una al getProperty(llave), para poder visualizar el valor que está asociado a la llave.

En cuanto a la interfaz Enumeration, cuenta con dos métodos importantes que son el hasMoreElements(), que devuelve un booleano y nextElement() que devuelve la siguiente llave de la colección. Con estas dos es posible entrar a cada elemento de la colección, ya que se coloca el hasMoreElements() dentro de un while como condición y el nextElement() regresa la llave siguiente y se va buscando cada elemento de la colección por medio de su llave.

De esa forma se puede acceder a cada uno de los elementos de la tabla y poder imprimirlos para el caso del ejercicio 3, ya que el ejercicio pide la impresión, para que se puedan visualizar cada uno de ellos.

* **Análisis de los ejercicios.**

**Ejercicio 1**

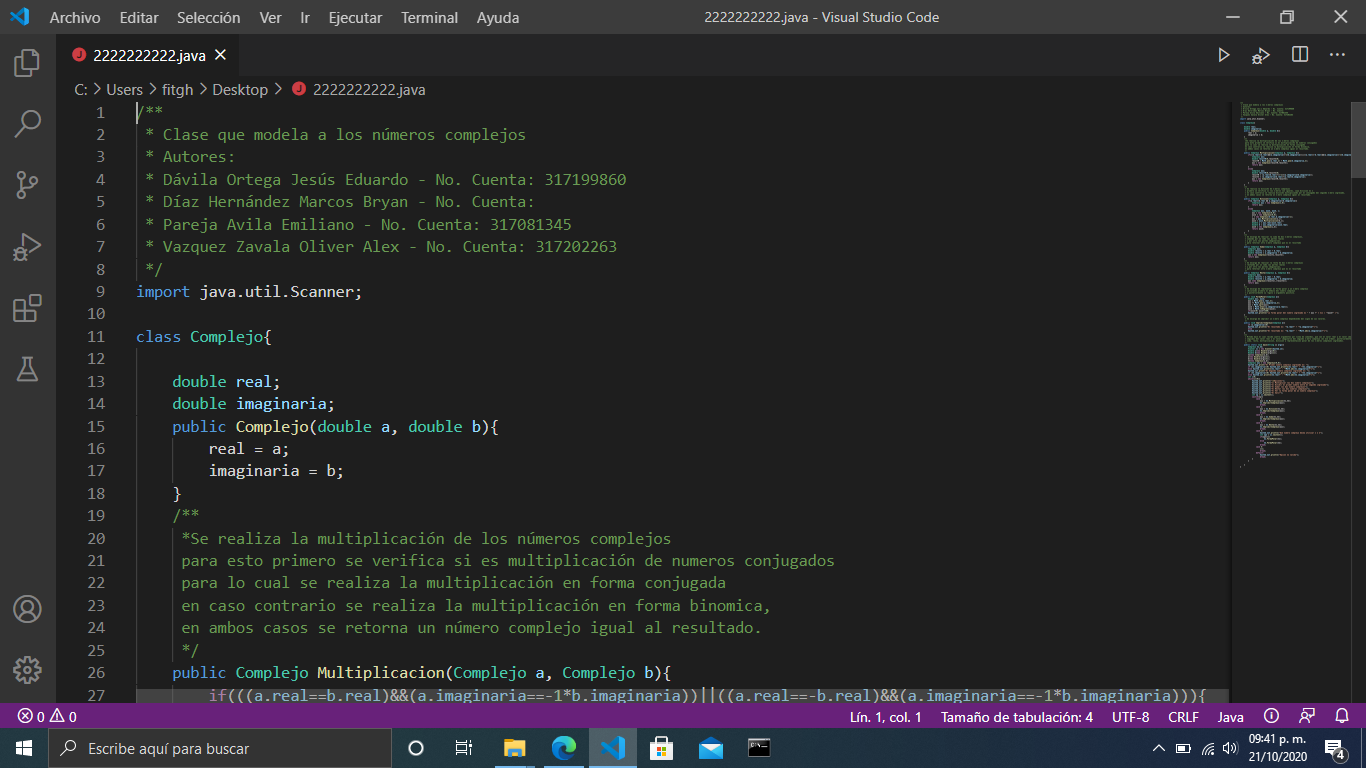
* **Análisis previo**

Para este ejercicio empezamos a pensar en cuáles son los atributos que hacen que un número sea un número complejo y lo primero que se nos vino a la mente fue que este posee una parte real y una imaginaria por lo cual se definió como atributos de la clase complejos, dos variables de tipo double las cuales almacenan el valor numérico de cada parte, posteriormente de haber definido los atributos se decidió crear un constructor, el cual inicializa ambas partes con los valores ingresados por el usuario, posterior al constructor se decidieron crear los métodos de la clase y en estos métodos se incluyen todas las operaciones básicas con los números complejos más tres métodos extras uno que es para crear el conjugado de un número complejo, este método solo se va a usar en la división por la manera de resolver la misma, otro método nos permite obtener la representación polar del número complejo, y por último un método que me permite imprimir los resultados o como imprimir un número complejo.

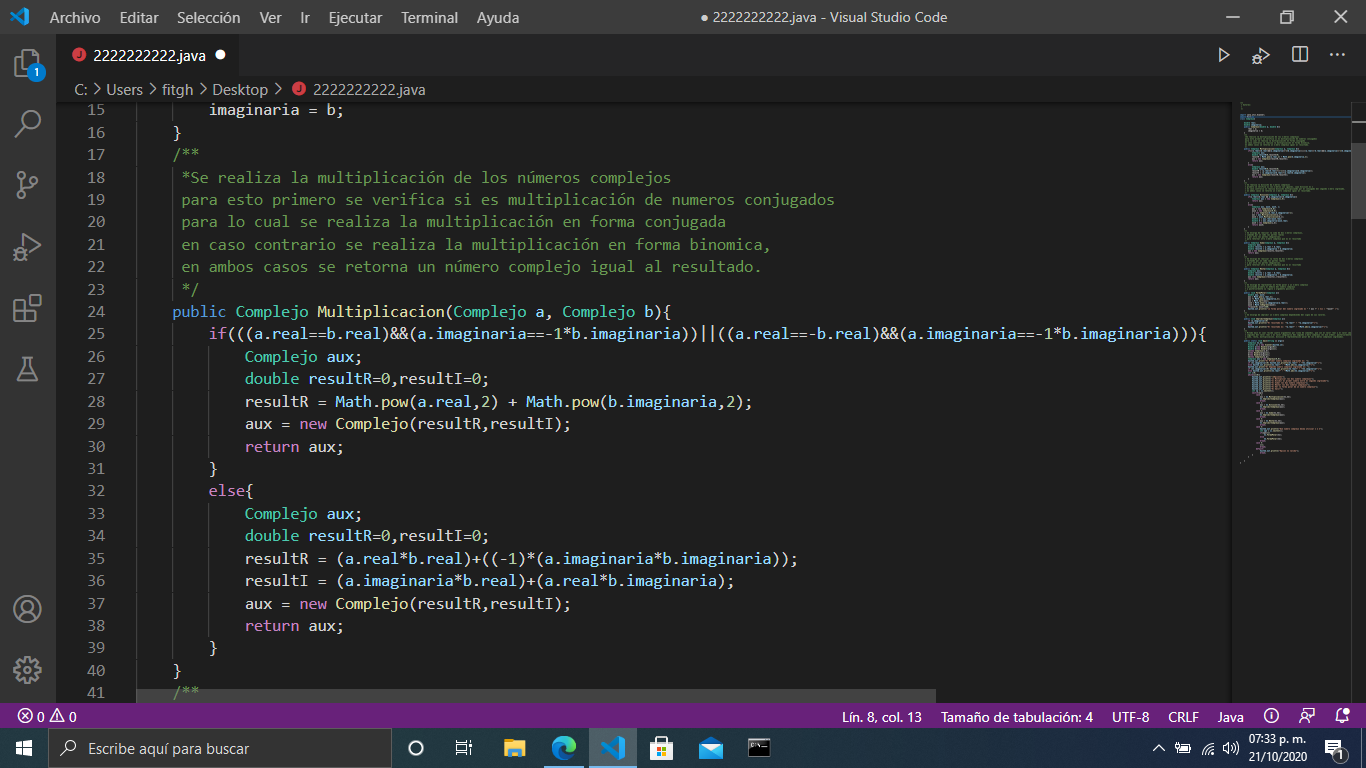
Al momento de implementar las ideas en el código se decidió intentar hacer el código lo más entendible posible para que al momento de que lo vieran más personas pudieran seguir fácilmente el flujo de cómo funciona el código sin mayor complicación, lo que más tiempo y en donde más errores se cometieron fue al momento de empezar a hacer el método de división, esto se debió a que al momento de dividir dos complejos estos tienen una metodología interesante para resolverse y por esta metodología también se creó un método llamado Conjugado el cual ayuda al momento de hacer la división

* **Desarrollo**

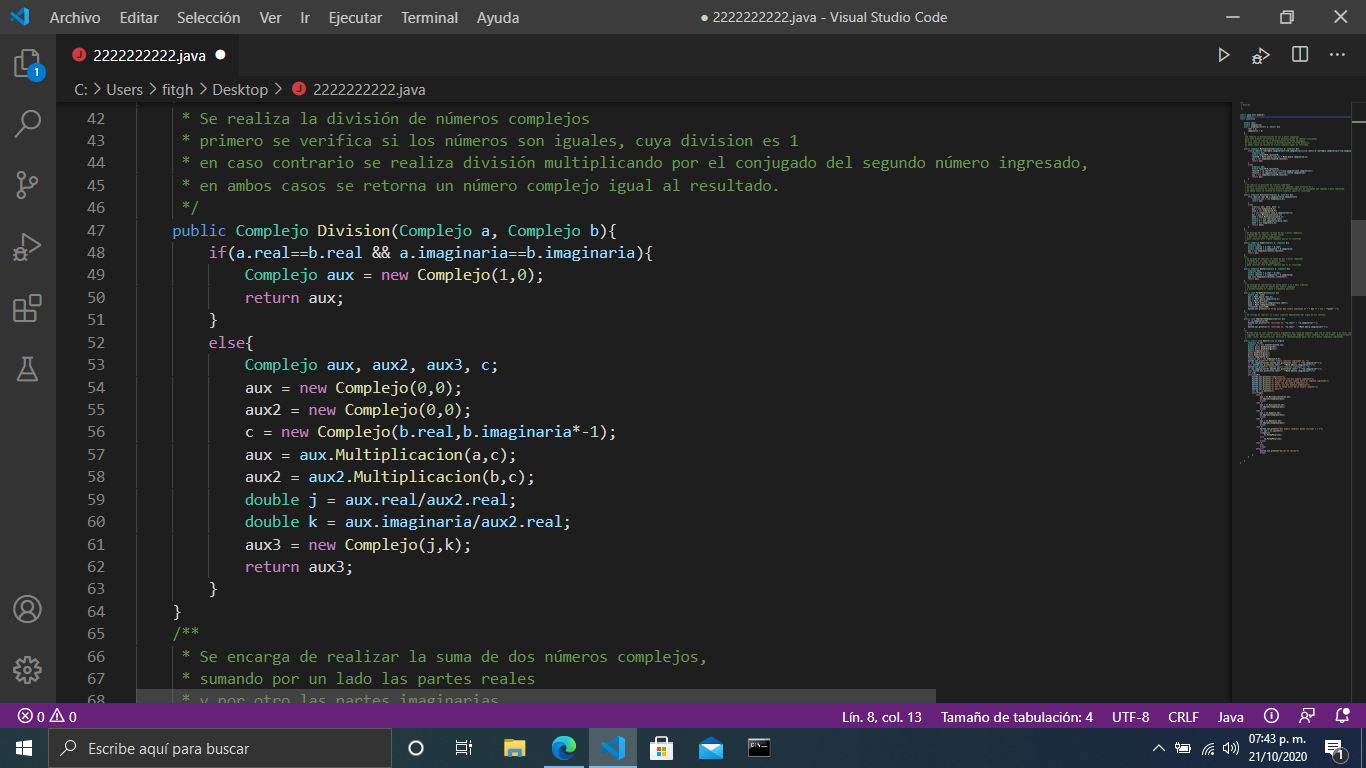
Inicialmente se importó la clase Scanner, debido a que se requiere información sobre qué operación se realizará con los dos números complejos, posteriormente se crea la clase para modelar a los números complejos, para ello dicha clase cuenta con dos variables de instancia del tipo flotante, donde se almacenará el valor de la parte imaginaria y real de los objetos encargados de modelar a los números complejos, apoyándose de un método constructor.



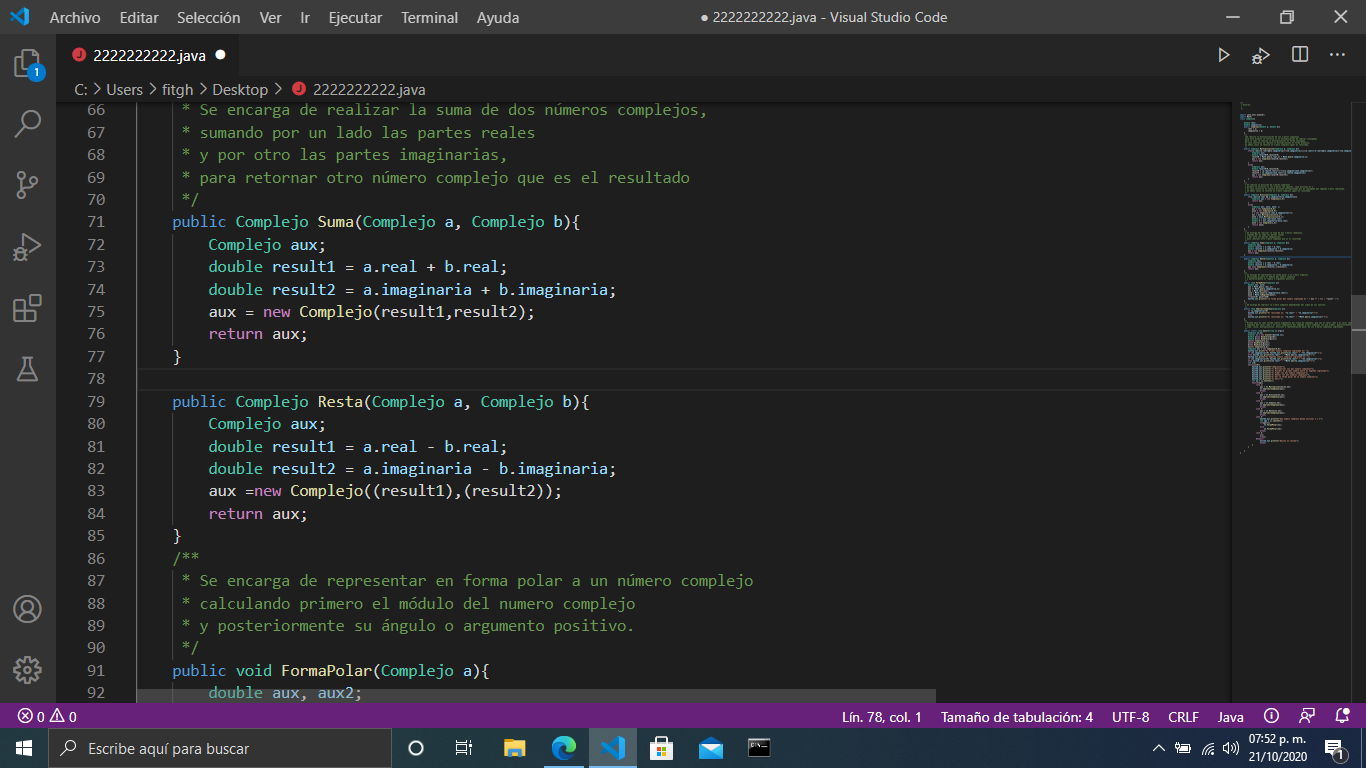
Posteriormente se crearon los métodos para realizar las operaciones, el primer método es para la multiplicación de los números complejos, para esto se verifica si los números a multiplicar son conjugados y así realizar la multiplicación equivalente a la suma de los cuadrados de la parte real e imaginaria, en otro caso se realiza la multiplicación en forma binómica, y se retorna el resultado como se muestra:



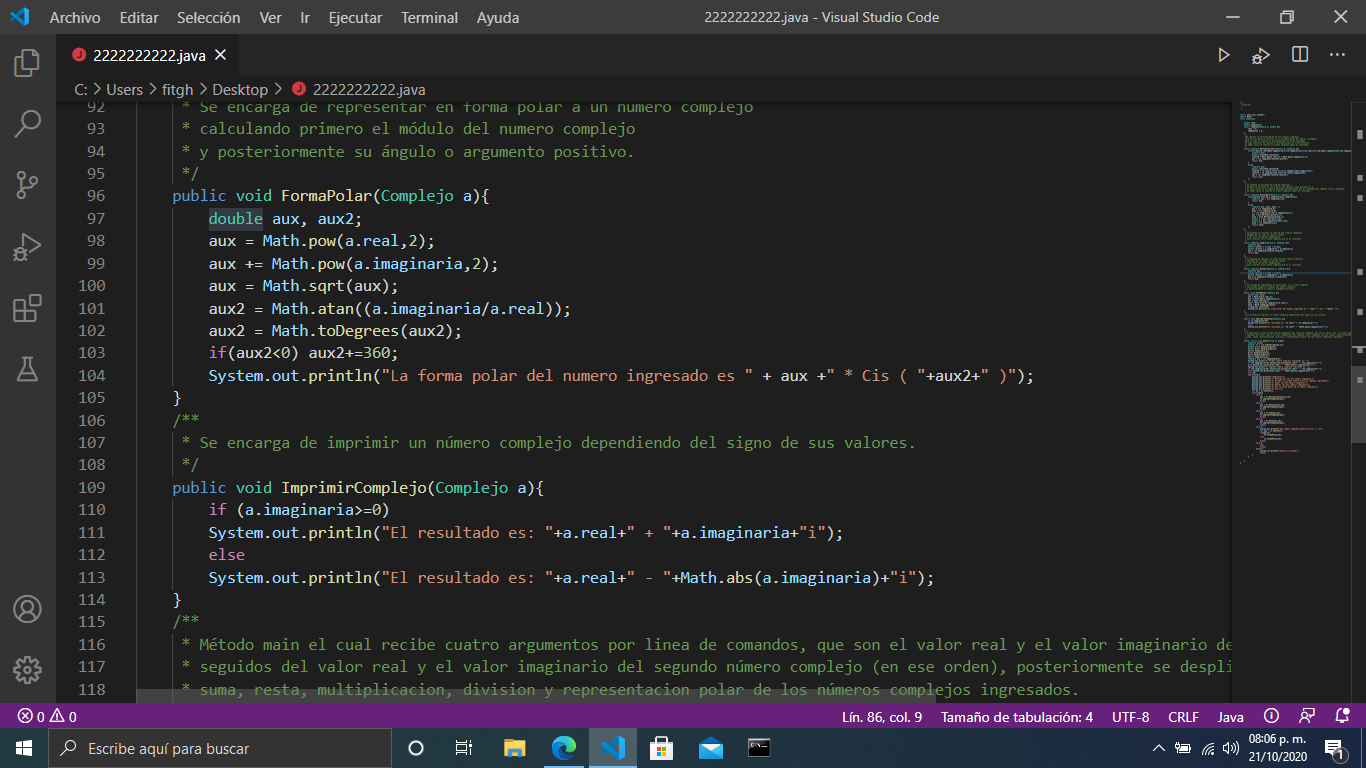
Para la división se verifica primero si se quiere dividir a un número por su igual, cuyo resultado retornará la unidad, en caso contrario se multiplicará a ambos resultados por el conjugado del segundo número ingresado (divisor), multiplicando por separado al primer número y al segundo número, esto apoyándose del método definido anteriormente para realizar dicha multiplicación, al resultado de la división del primer número, le será dividida tanto la parte real, como la imaginada, por el número real obtenido de la multiplicación del segundo número por su conjugado, retornado el resultado:



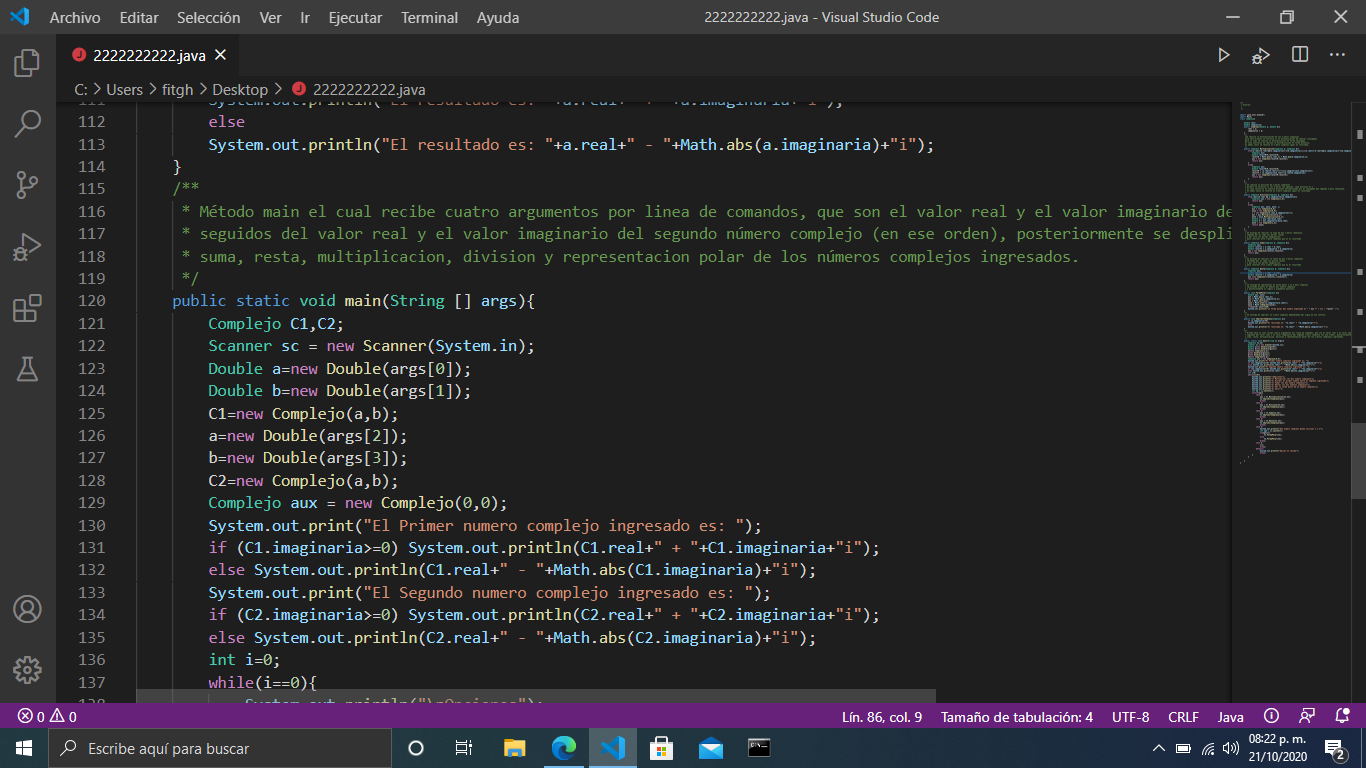
Para la suma y resta se realiza la respectiva operación entre las partes reales por un lado y por otro entre las partes imaginarias, para retornar un número complejo cuya parte real es resultado de operar las partes reales de los dos números y de igual manera la parte imaginaria es el resultado de operar las partes imaginarias de los dos números.



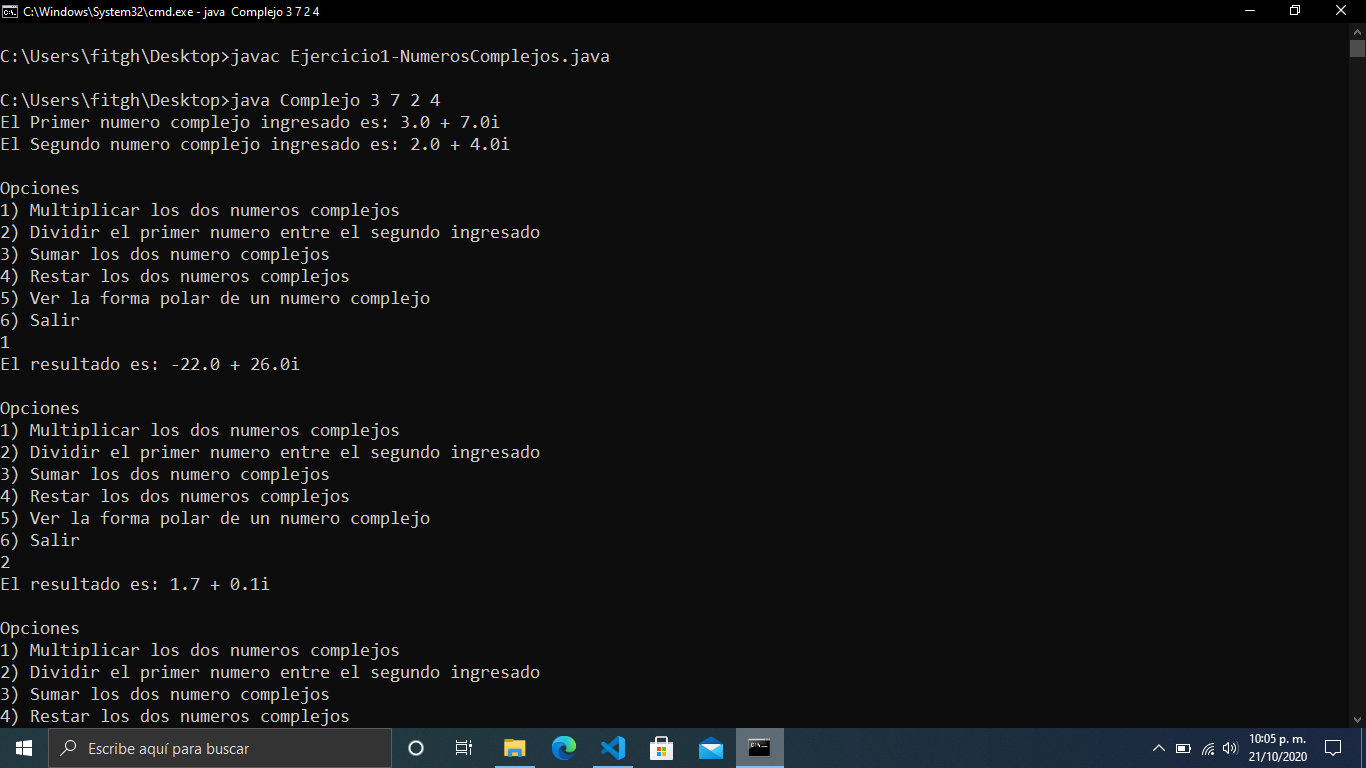
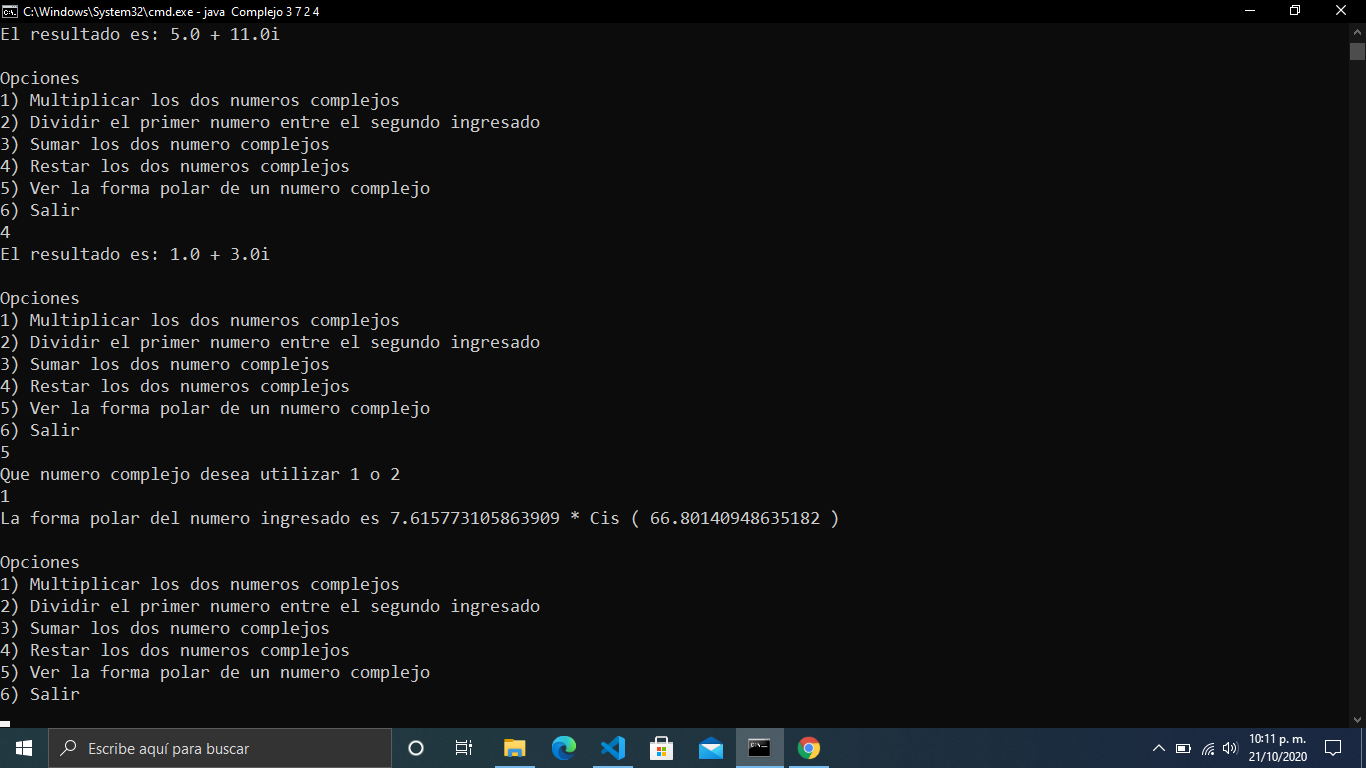
Para poder representar a uno de los números en su forma polar primero se calcula su módulo que es igual a la raíz de la suma de los cuadrados de la parte real e imaginaria, posteriormente su ángulo será igual a la tangente inversa de la parte imaginaria dividida entre la parte real, para esto se hace uso de math y en caso de que el ángulo resulte negativo bastará con sumar 360, para obtener su correspondiente ángulo positivo, el método desarrollado fue el siguiente:



El método main recibirá argumentos desde la línea de comandos al momento de ejecutar el código, en este caso se solicitará al usuario la parte real y la parte imaginaria del primer número complejo y la parte real e imaginaria del segundo número, en ese orden los cuatro argumentos, debido a que el método main solamente recibe argumentos de tipo cadena se requiere de un casteo a double para poder definir a los objetos y poder trabajar con las operaciones.



Una vez definidos los dos números complejos se desplegará un menú con las opciones de multiplicación, división, suma, resta, forma polar y salir, donde el usuario puede seleccionar las opciones que desee realizar y se imprimirán en pantalla los resultados de las operaciones. **Ejecución del programa**

** **

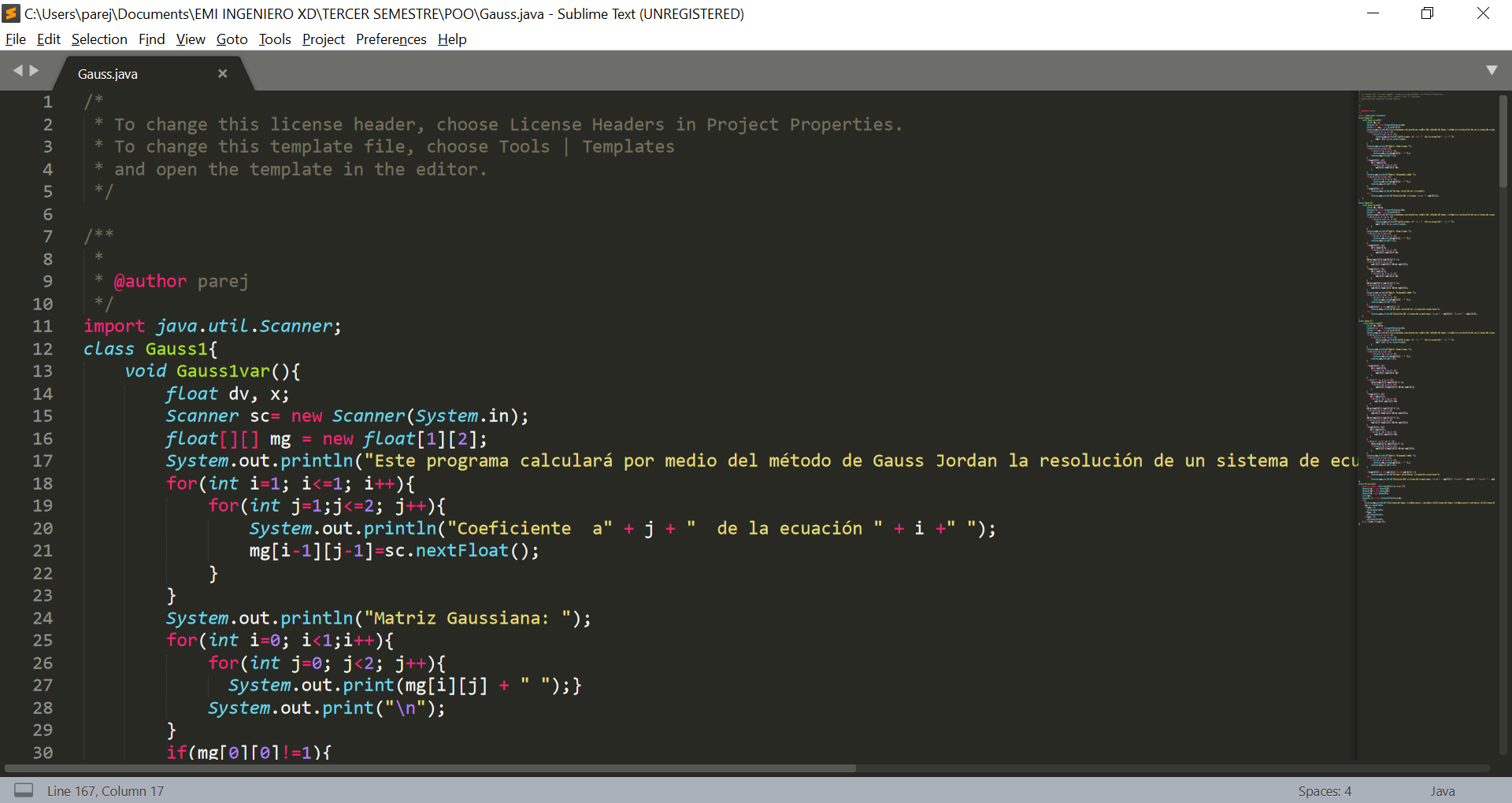
**Ejercicio 2**

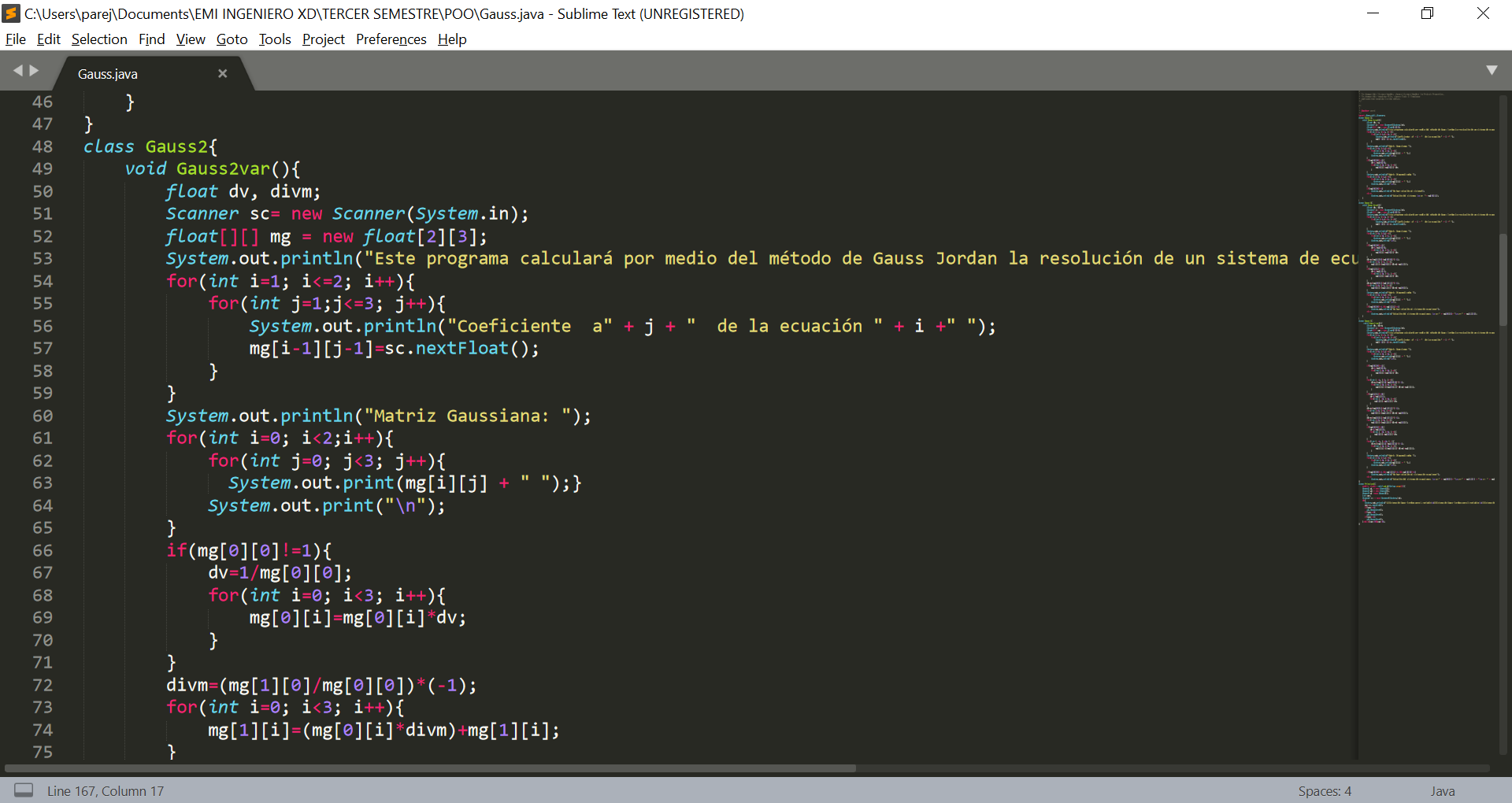
* **Análisis previo.**

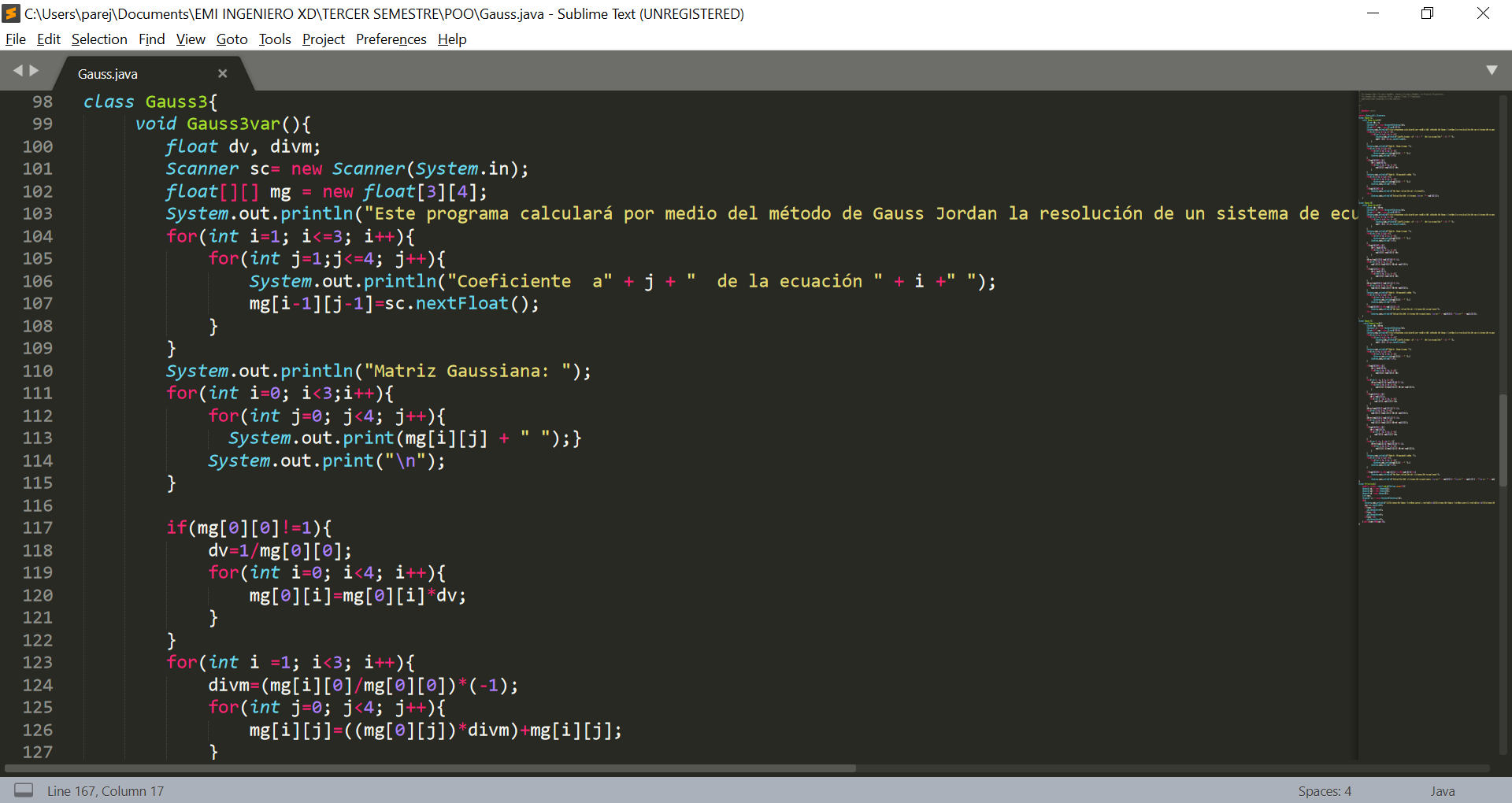
Para este ejercicio, primero investigamos la resolución de un sistema de ecuaciones por Gauss Jordan para 1, 2 y 3 variables, en todas se tiene que diagonalizar la matriz y hacer que esta sea igual a la matriz identidad, en el caso del sistema con solo una variable, vimos que no habría demasiado problema, ya que solo era dividir el coeficiente de la variable y el resultado entre el coeficiente de la variable, y así nos daría el resultado de la variable. Para el sistema de 2 incógnitas, lo primero que establecimos que se debía hacer es convertir el primer coeficiente del primer renglón de la matriz en un 1, para que al momento de hacer la eliminación no tuviéramos problemas, entonces al igual que el ejercicio anterior, dividimos todo el primer renglón de la matriz entre el coeficiente del primer elemento del primer renglón. Una vez que el coeficiente fuera un 1, para diagonalizar la matriz y hacer el primer elemento del segundo reglón un 0, todo el primer renglón de la matriz, lo multiplicamos por el coeficiente del primer elemento del segundo renglón de la matriz, por el coeficiente del primer elemento del primer renglón de la matriz por -1, y el resultado que saliera lo sumamos a los elementos del segundo renglón, una vez que el primer elemento del segundo renglón fuera un cero, ahora haríamos el segundo elemento del segundo renglón de la matriz un 1, para ello dividimos todo el segundo renglón entre el coeficiente del segundo elemento del segundo renglón de la matriz, y para terminar de diagonalizar la matriz, todo el segundo renglón lo multiplicamos por el coeficiente del segundo elemento del primer renglón entre el coeficiente del segundo elemento del segundo renglón por -1 y sumamos todos los resultados obtenidos al segundo renglón, así la matriz ya era igual a la identidad y los terceros elementos del primer y segundo renglón eran los valores de las variables del sistema de ecuaciones. Y para el sistema de 3 ecuaciones, al tener ya la matriz construida, lo primero que haríamos es hacer el primer elemento del primer renglón de la matriz un 1, por tanto dividimos todo el primer renglón de la matriz, entre el coeficiente del primer elemento del primer renglón de la matriz. Después para hacer los primeros elementos del segundo y tercer renglón de la matriz 0s, multiplicamos todo el primer renglón de la matriz por el coeficiente de los primeros elementos del segundo y tercer renglón respectivamente entre el coeficiente del primer elemento del primer renglón de la matriz, y los resultados que fuéramos obteniendo, se los sumamos al segundo y tercer renglón. Después para seguir con la diagonalización de la matriz, ahora hicimos el segundo elemento del segundo renglón de la matriz un 1, y para lograrlo, dividimos todos los elementos del segundo renglón de la matriz, entre el coeficiente del segundo elemento del segundo renglón de la matriz, y para hacer 0s los segundos elementos del primer y tercer renglón multiplicamos los valores del segundo renglón por los coeficientes de los segundos elementos del primer y tercer renglón respectivamente entre el coeficiente del segundo elemento del segundo renglón por -1, y todos los resultados obtenidos los sumamos al primer y tercer renglón. Y para finalizar la diagonalización y hacer la matriz igual a la matriz identidad, primero hicimos al tercer elemento del tercer renglón de la matriz un 1, y para ello, dividimos todo el tercer renglón entre el coeficiente del tercer elemento del tercer renglón. Y para ahora hacer 0s los terceros elementos del primer y segundo renglón de la matriz, multiplicamos todos los elementos del tercer renglón por los coeficientes de los terceros elementos del primer y segundo reglón entre el coeficiente del tercer elemento del tercer renglón por -1 y los resultados obtenidos los sumamos al primer y segundo renglón y así concluía la diagonalización de la matriz, y los cuartos elementos del primer, segundo y tercer renglón, serían los valores de las variables de nuestro sistema de ecuaciones.

* **Desarrollo**

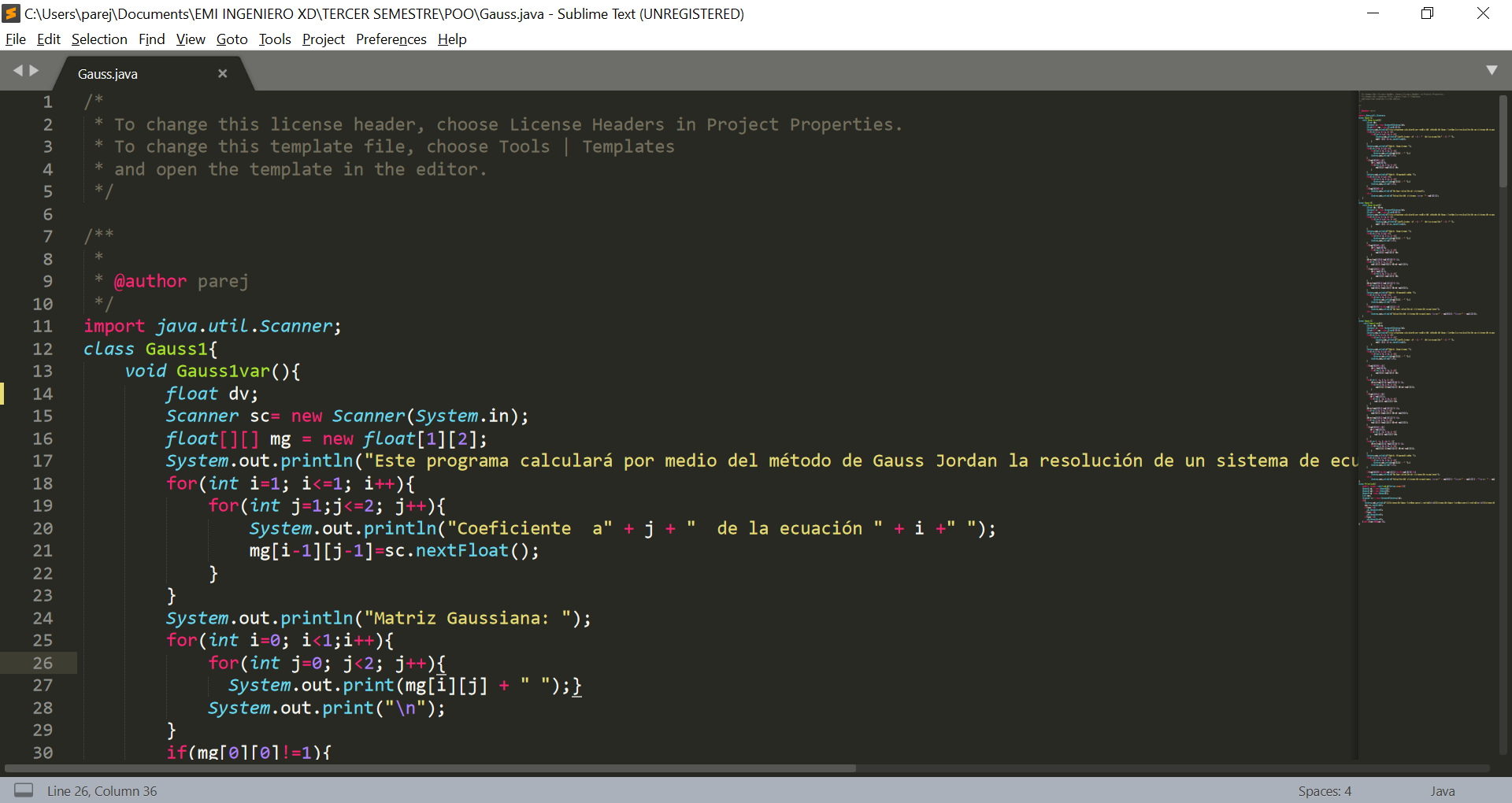
Primero decidimos crear una clase para cada resolución de cada sistema de ecuaciones, así se nos facilitaría más al momento de llamar los métodos de dichas clases desde la clase principal en su método main



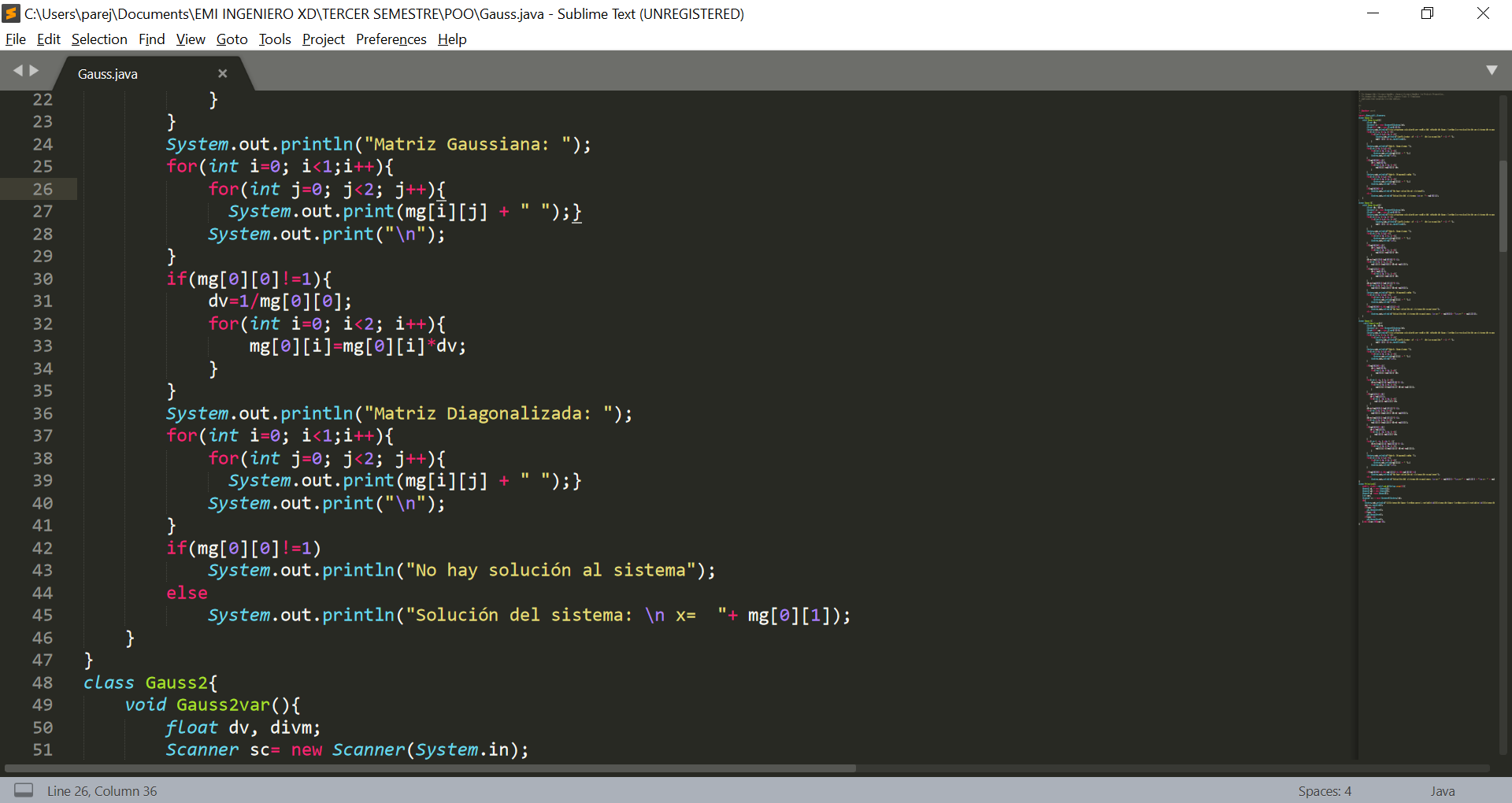




* **En la resolución de un sistema de 1 variable:**

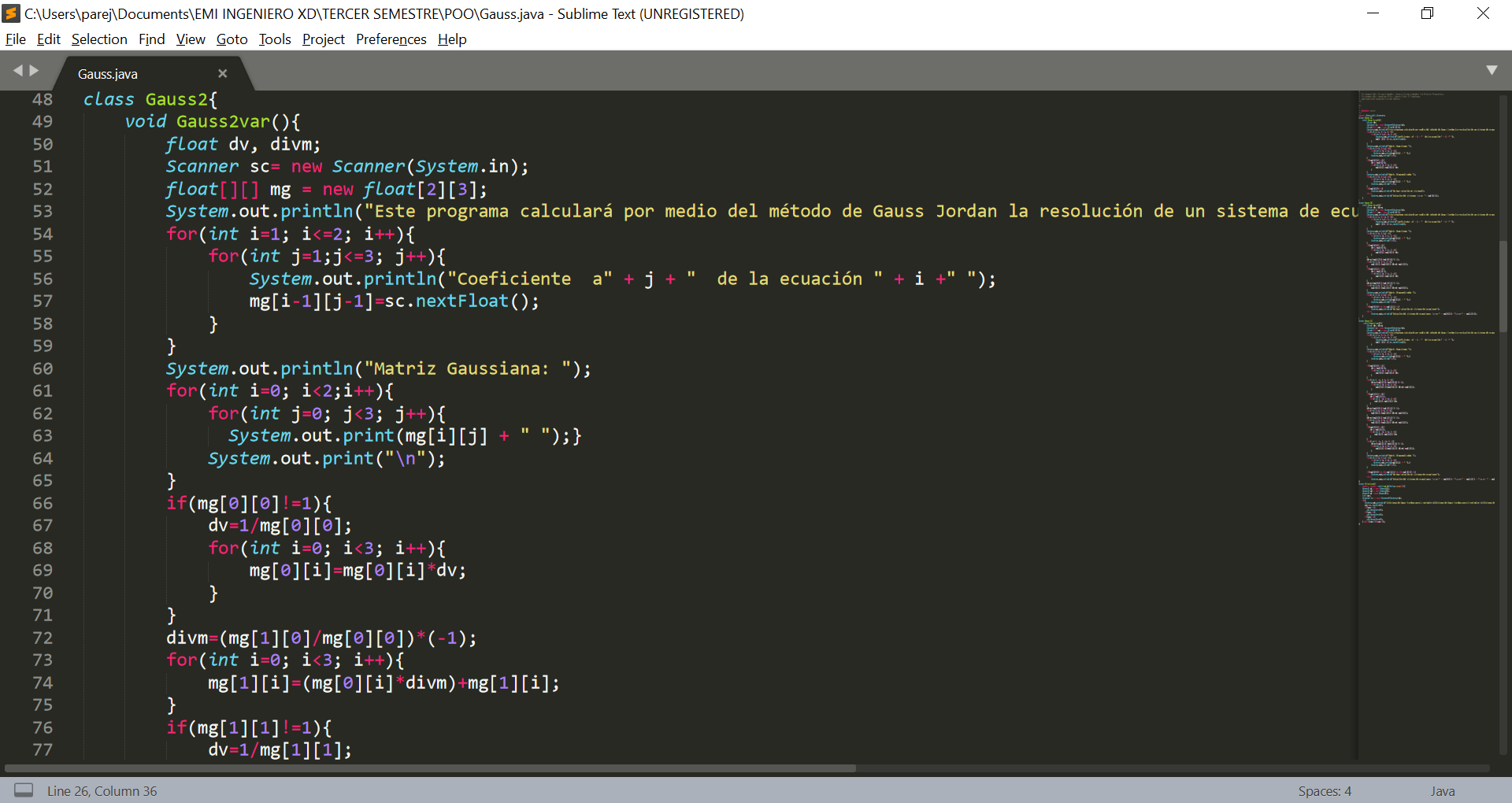


En la clase Gauss1 definimos un método de tipo void llamado Gauss1var y dentro de este método, primero creamos un objeto de la clase Scanner que nos ayudaría a leer los valores ingresados por el usuario, después declaramos 1 variable de tipo float llamada div, que nos ayudaría al momento de diagonalizar la matriz, y después declaramos un arreglo multidimensional de floats de 1x2 aunque esto era innecesario, ya que con solo declarar un arreglo de 2 elementos era más que suficiente, pero decidimos ponerlo así, puesto que debe seguir el método de Gauss-Jordan, después indicamos al usuario lo que hace el programa y empezamos a leer los coeficientes del sistema de ecuaciones, gracias a 2 ciclos for anidados con el método nextFloat() leímos los valores de la matriz. Y justo después de terminar la lectura de datos, imprimimos a pantalla la matriz, que solo imprimiría una línea ya que solo era un sistema de una sola variable.

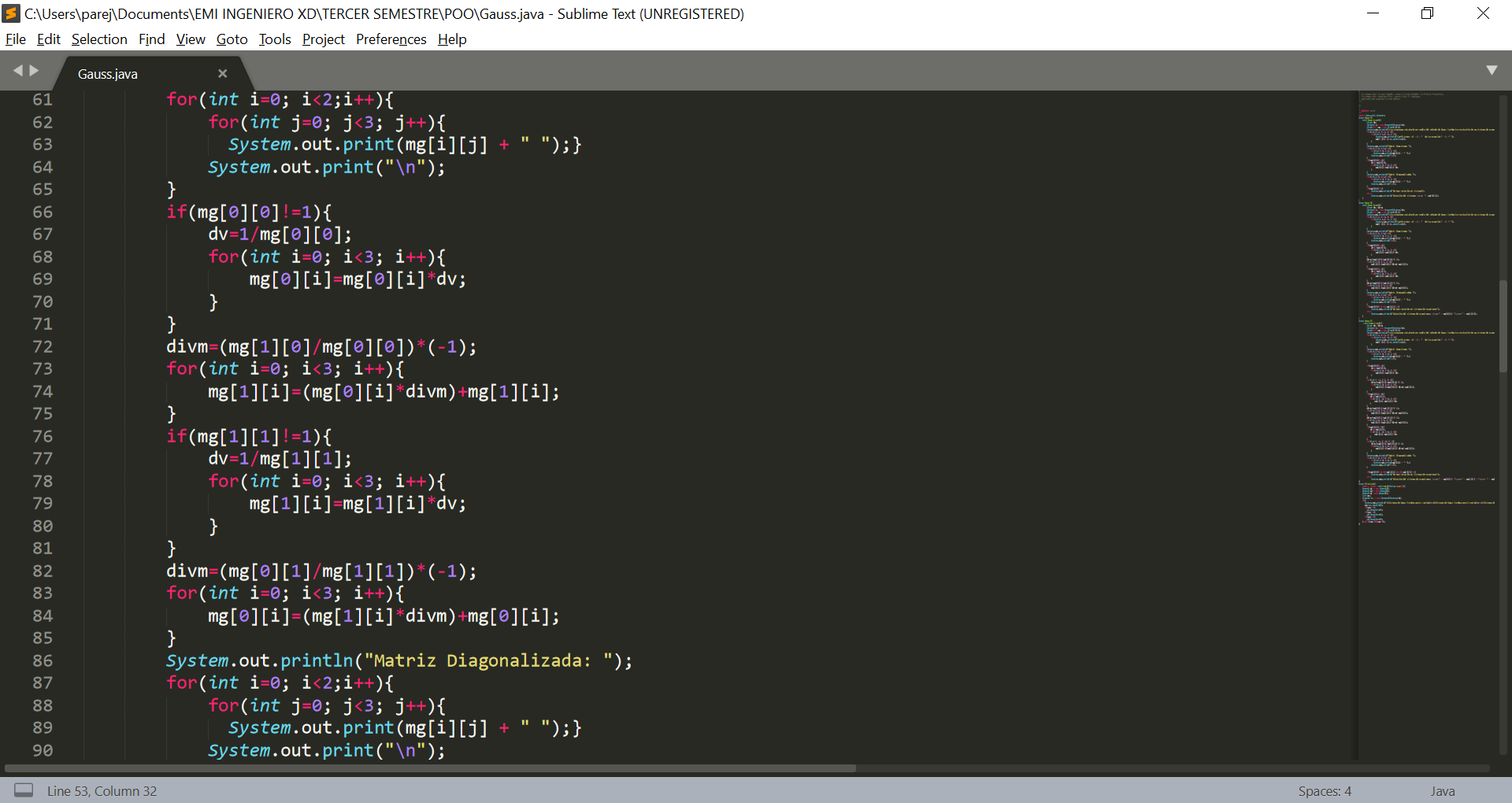


Después para diagonalizar la matriz, verificamos que el primer elemento del primer renglón fuera un 1, si no era un 1 significa que el sistema ya estaba resuelto, de lo contrario, la variable dv tomaba el valor de 1 entre el primer elemento del primer renglón y después gracias a un ciclo for, cambiamos el valor de todos los elementos del primer renglón, multiplicando todos elementos del primer renglón por el valor de div. Así logramos que nuestra matriz fuera diagonalizada en su totalidad. Después imprimimos a pantalla la matriz gracias a 2 ciclos for anidados, que en realidad solo imprimirían una línea. Y después para evitar cualquier tipo de errores verificamos con una estructura de control de flujo if que el primer elemento del primer renglón de la matriz fuera un 1, ya que en caso contrario significa que el sistema no tiene solución, y si este elemento fuera un 1 significa que nuestra matriz fue diagonalizada correctamente y se imprime a pantalla el valor de la incógnita del sistema de ecuaciones que es el segundo elemento del primer renglón de la matriz.

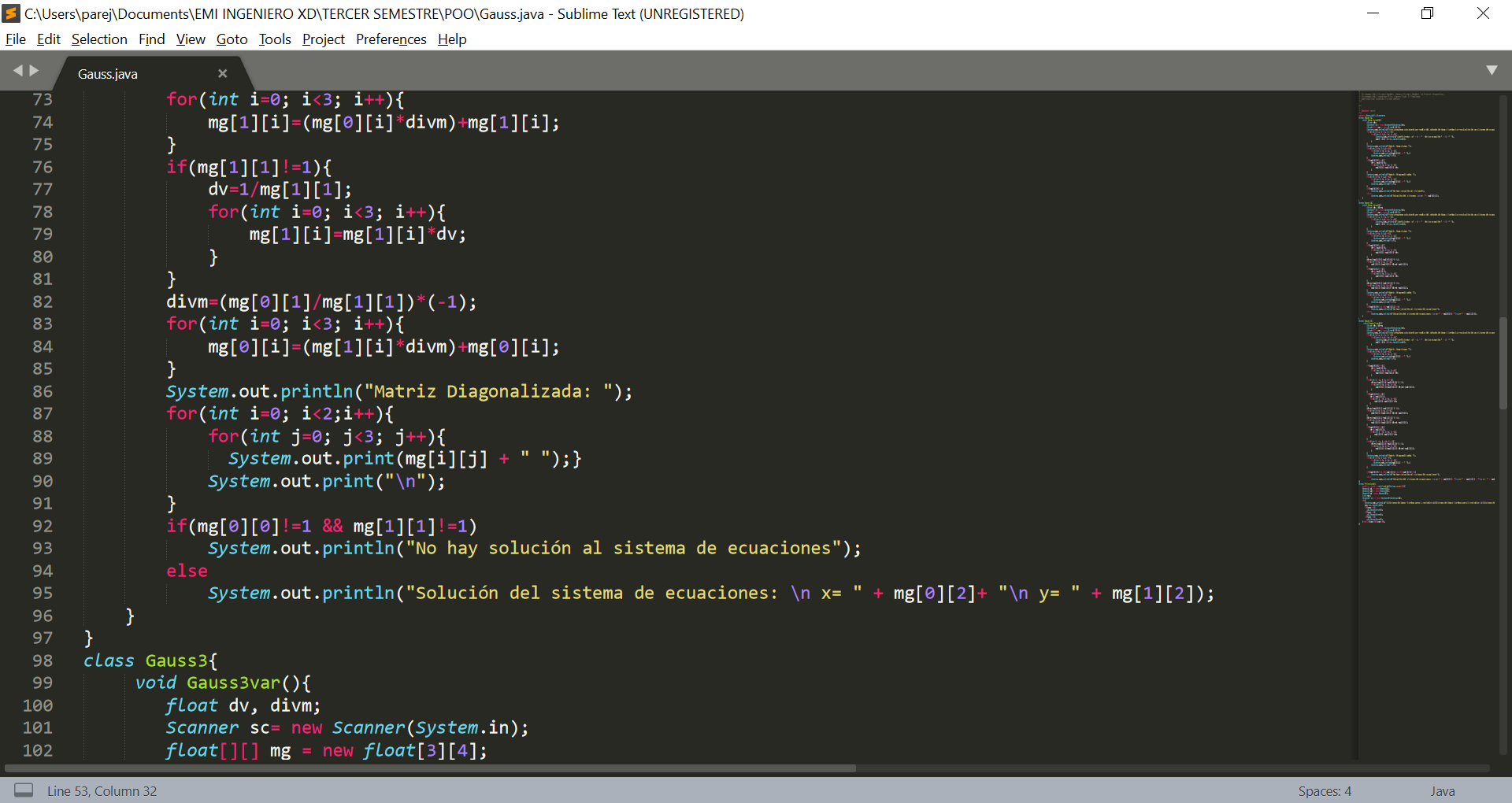
* **En la resolución de un sistema de 2 variables:**



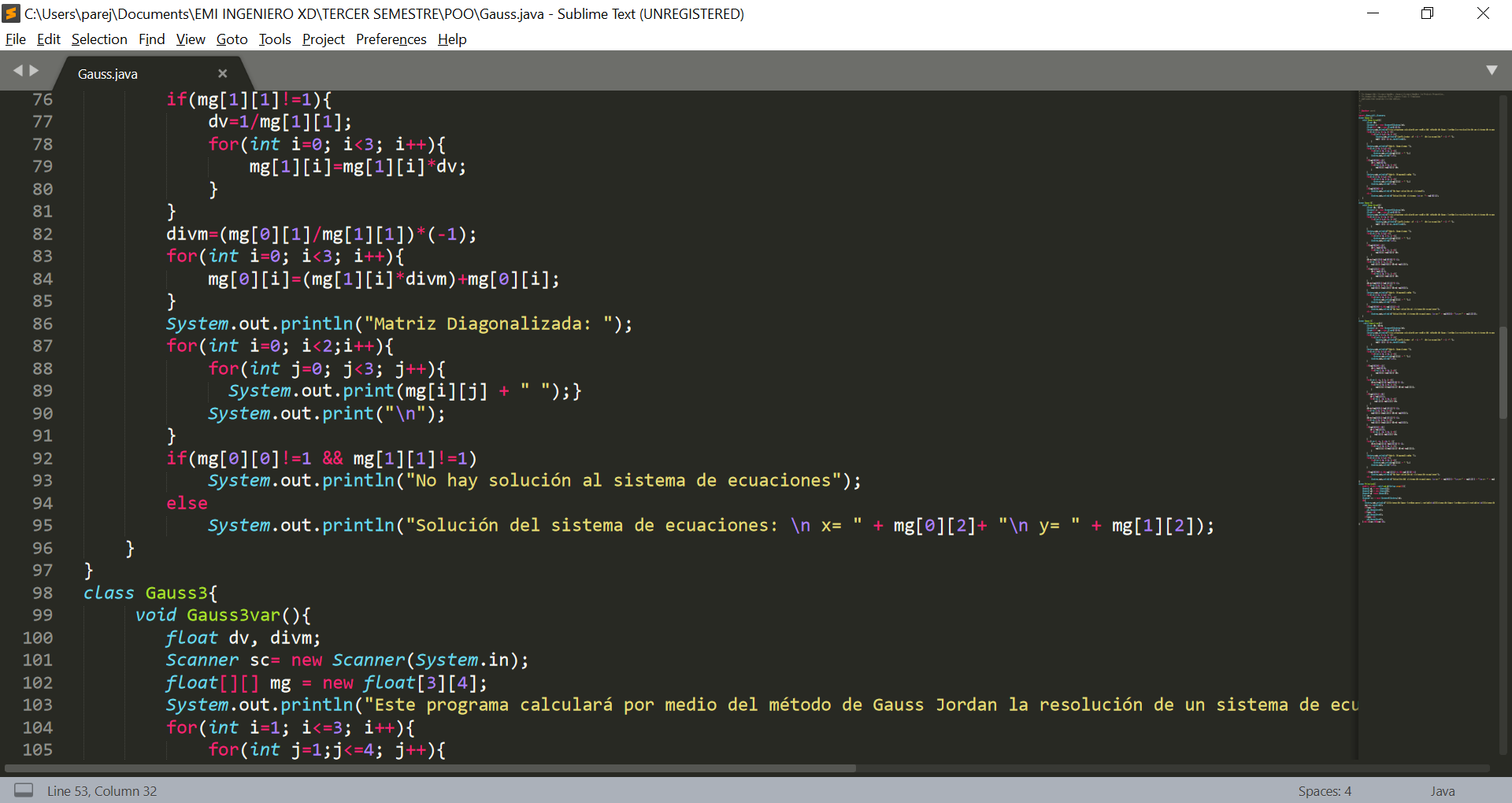
Creamos una clase llamada Gaus2 y también su método tipo void llamado Gauss2var, en él primero creamos 2 variables tipo float dv y divm, una nos ayudará a hacer 1 cierto elemento de la matriz que queramos y el otro nos ayudará al momento de diagonalizar la matriz. Después creamos un objeto de la clase Scanner llamado sc que con su método nextFloat() nos ayudará a leer los datos de los coeficientes de la matriz. Y creamos nuestro arreglo multidimensional de floats para la matriz, de 2x3, después indicamos al usuario lo que hace el programa y el orden de entrada de los datos, luego con 2 ciclos for anidados hacemos la lectura de datos, el primer ciclo for hace la lectura de datos por renglón y el segundo por columnas. E inmediatamente después de finalizar la lectura de datos de la matriz, imprimimos los datos de la propia, igualmente gracias a 2 ciclos for anidados, donde el primero imprime los datos del renglón, y el segundo de la columna.

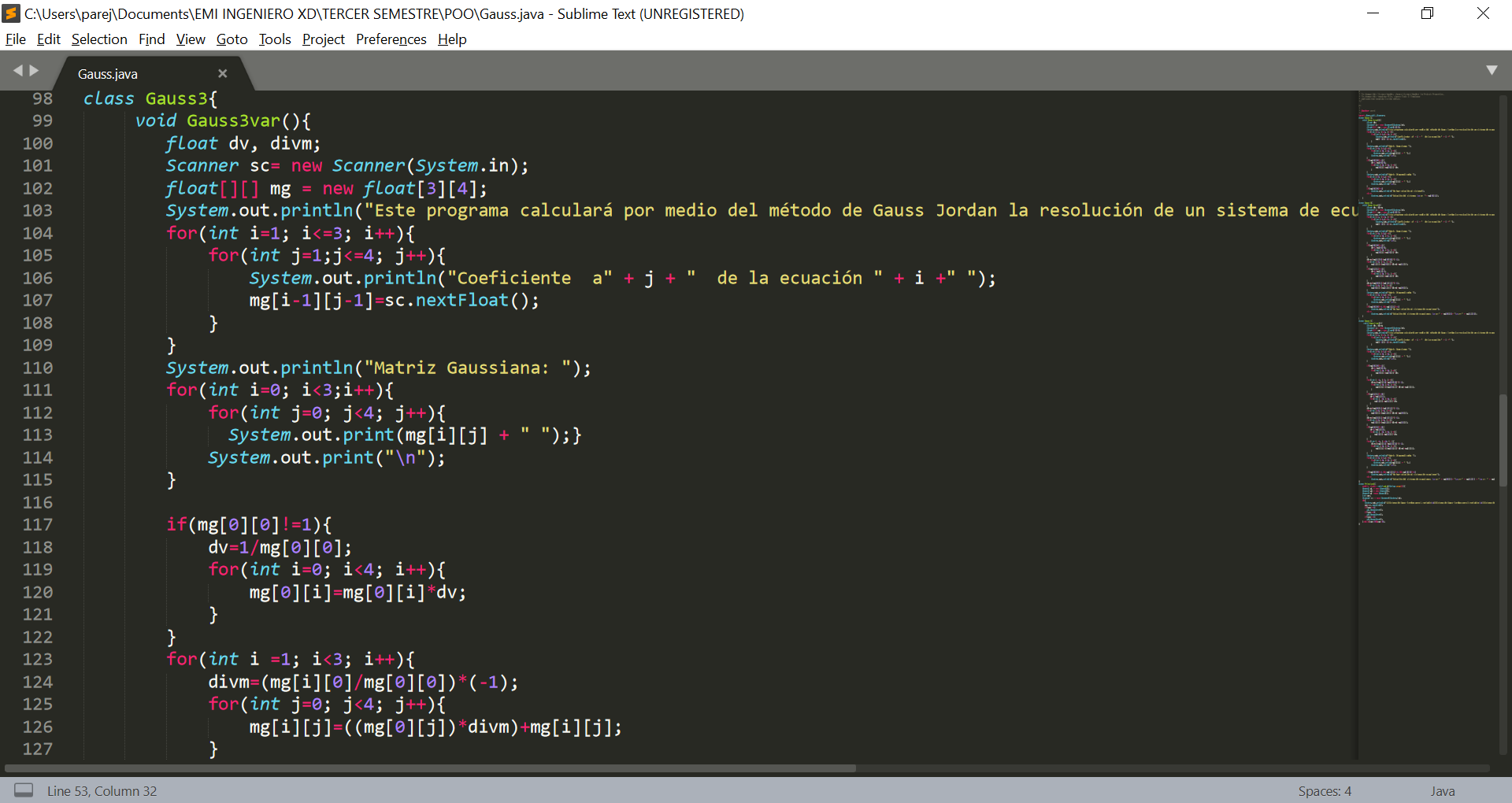


Después con una estructura de control de flujo if verificamos que el primer elemento del primer renglón sea igual a 1, si la condición es falsa, entonces la variable flotante dv toma el valor de 1 entre el valor del primer elemento del primer renglón de la matriz, y después con un ciclo for cambiamos todos los elementos del primer renglón, multiplicando cada uno de ellos por div. Al finalizar este proceso el primer elemento del primer renglón de la matriz tomaría el valor de 1. Después para hacer la eliminación Gaussiana, la variable divm toma el valor del primer primer elemento del segundo renglón entre el primer elemento del primer renglón y lo multiplica por -1. Luego se hace la eliminación con un ciclo for donde uno por uno, todos los elementos del primer renglón se multiplican por divm y el resultado se suma al elemento que se encuentre en la misma posición pero en el renglón 2, y el valor resultante se almacenará en los elementos del segundo renglón, al finalizar este ciclo for el primer elemento del segundo renglón será igual a 0.

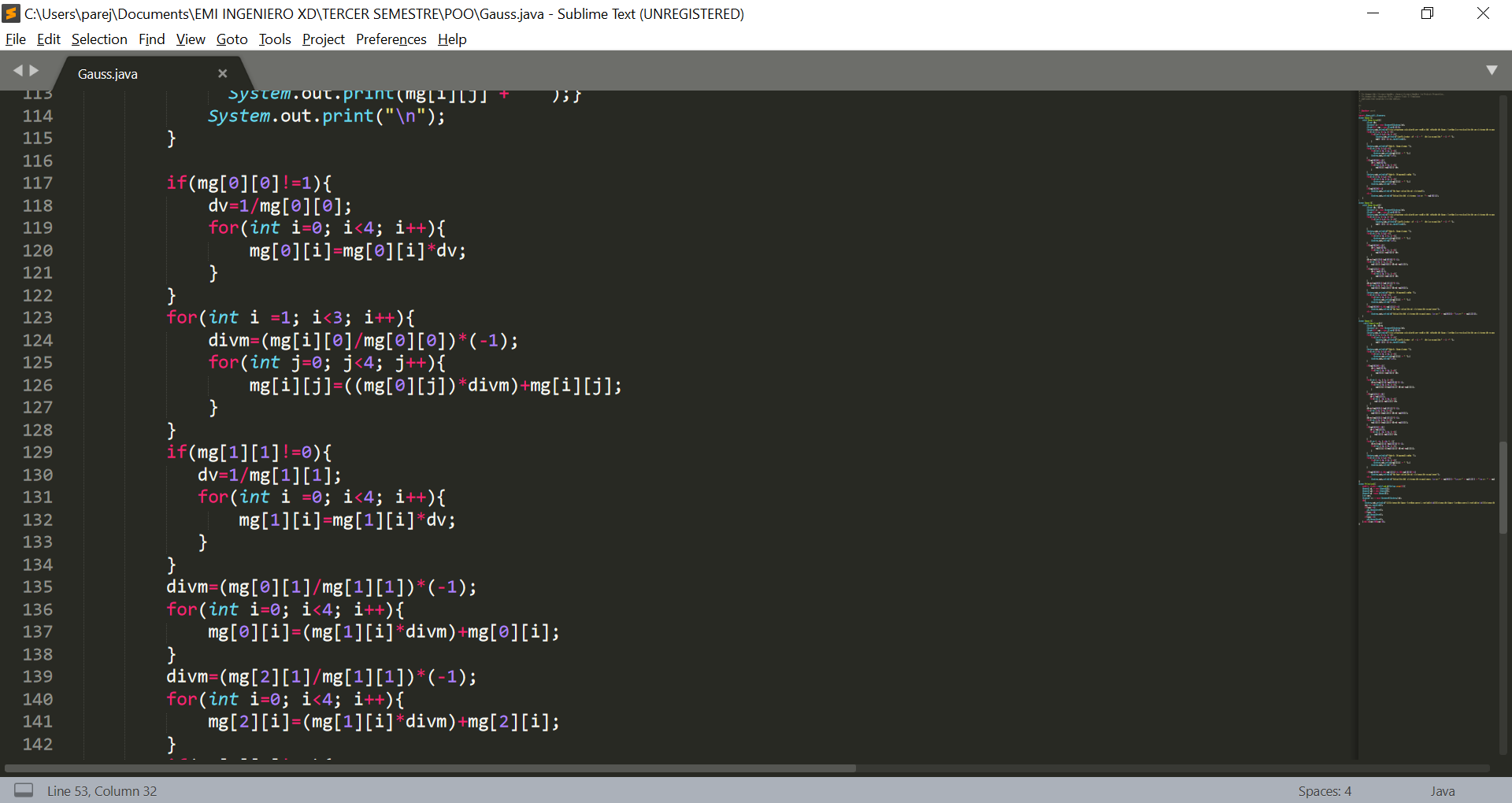


Luego, en una estructura de control de flujo if verificamos que el segundo elemento del segundo renglón sea igual a 1, si la condición es falsa, entonces tendremos que hacer a ese elemento un 1, esto se logra dándole a dv el valor de 1 entre el valor del segundo elemento del segundo renglón de la matriz, y después, mediante un ciclo for cambiamos el valor de todos los elementos del renglón multiplicando a cada uno de ellos por el valor de dv. Una vez que el segundo elemento del segundo renglón es igual a 1, ahora divm tomará el valor del segundo elemento del primer renglón, entre el segundo elemento del segundo renglón por -1, Y se hace la eliminación, cambiando todos los valores del primer renglón, esto se logra con un ciclo for en donde en cada iteración de este el elemento del primer renglón que en la posición que corresponda al número de iteración será igual al elemento del segundo renglón en la posición que corresponda al número de iteración por divm y a este resultado se le suma el valor del elemento primer renglón que corresponda al número de iteración. Después de que concluya la diagonalización de la matriz ahora la matriz ya diagonalizada se imprime a pantalla gracias a 2 ciclos for anidados.

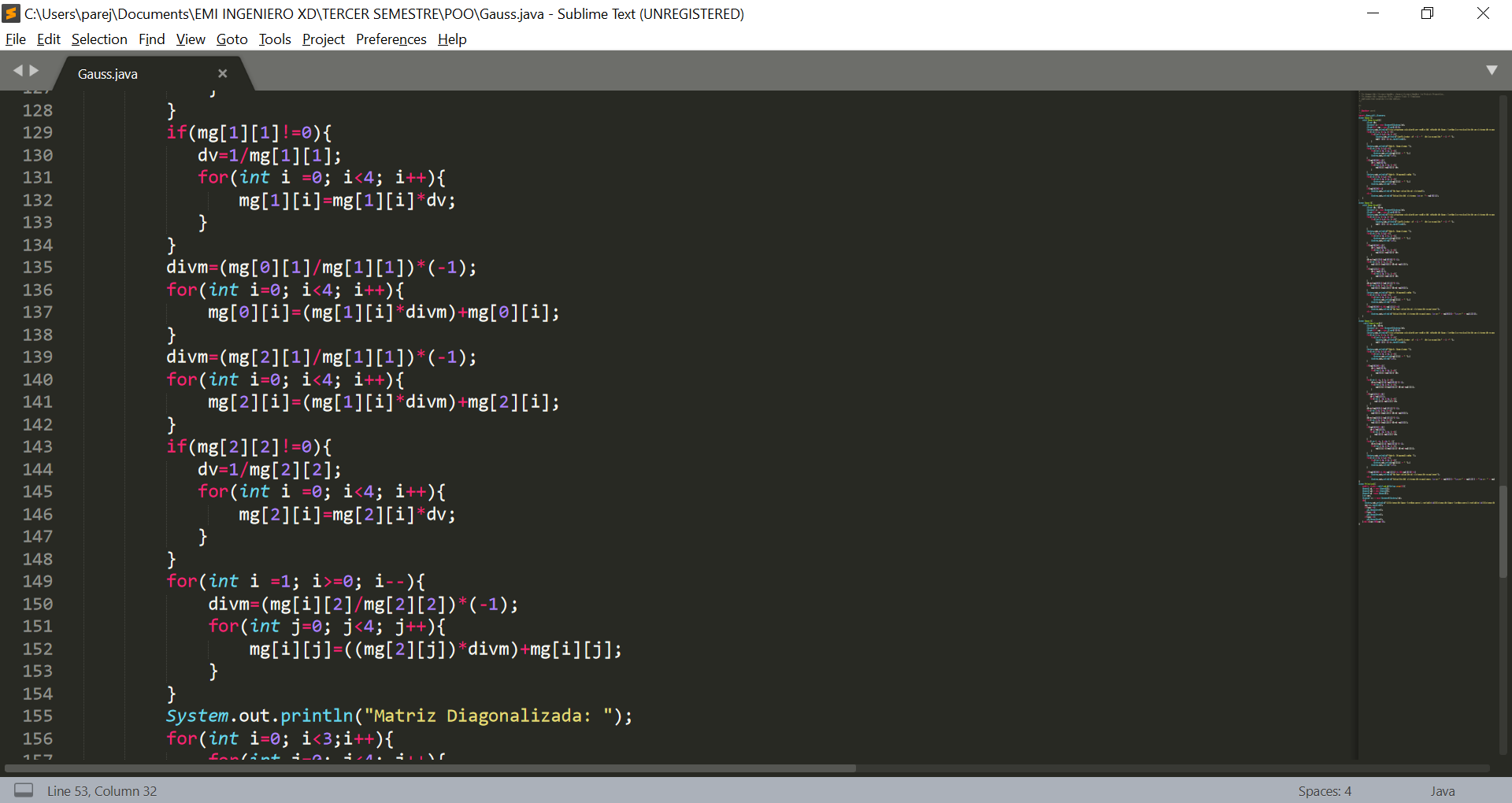
Para finalizar se verifica que el primer elemento del primer renglón y el segundo elemento del segundo renglón de la matriz, sean iguales a 1, si no son iguales, significa que el sistema no tiene solución y se indica haciendo una impresión a pantalla, y si ambos elementos son iguales a 1, la primer incógnita del sistema de ecuaciones tomará el valor del tercer elemento del primer renglón de la matriz y la segunda incógnita tomará el valor del tercer elemento del segundo renglón y se imprimen a pantalla sus valores.

* **En la resolución de un sistema de 3 variables:**

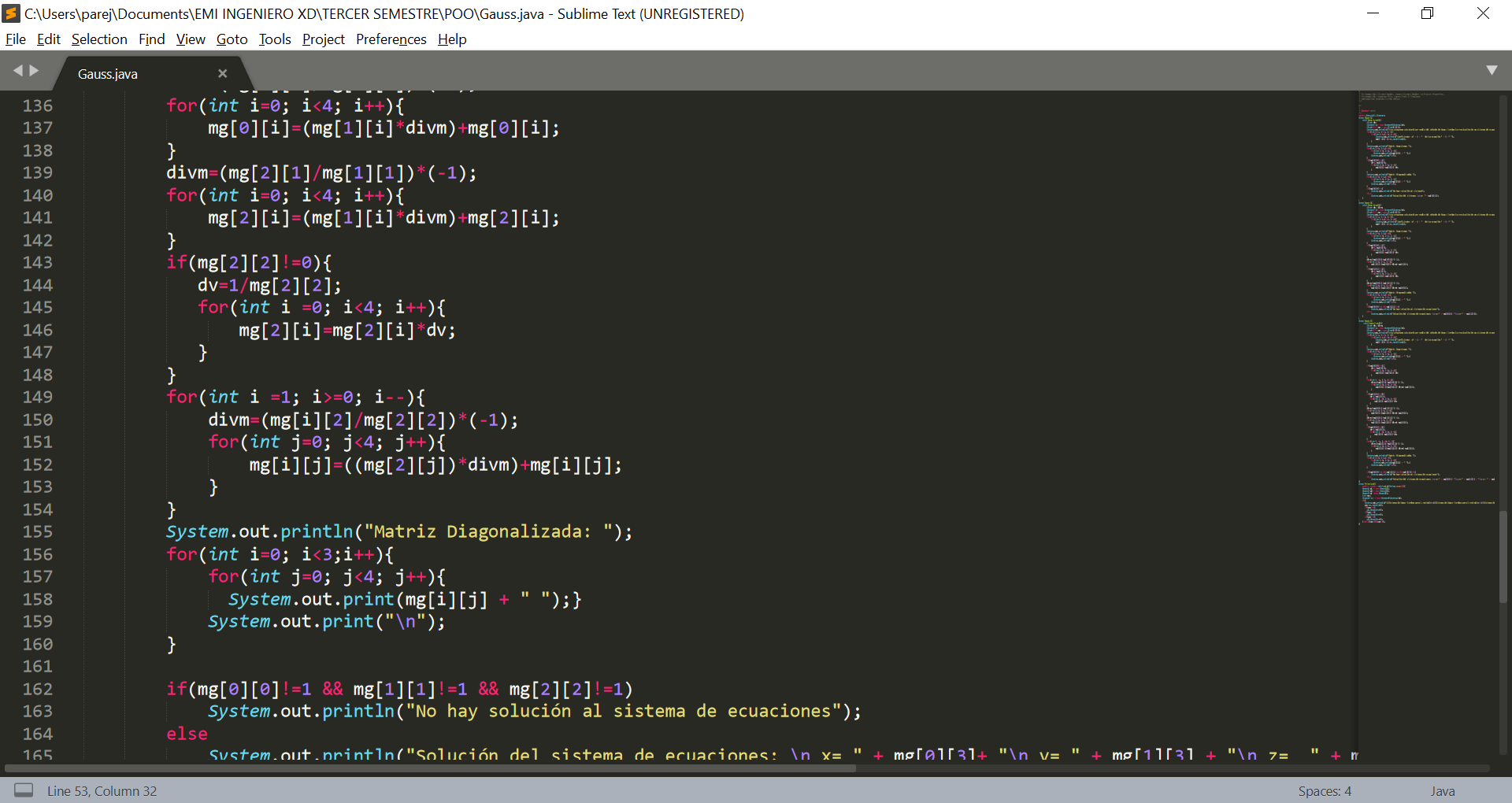
En la clase Gauss3 definimos su método de tipo void llamado Gauss3var, dentro de este método declaramos 2 variables de tipo float, dv y divm, la primera nos ayudará a hacer 1 cierto elemento de la matriz que requeriremos y la segunda nos ayudará a hacer la eliminación Gaussiana. Después creamos un objeto de la clase Scanner llamado sc que nos ayudaría a leer los valores de la matriz con su método nextFloat(), después creamos un arreglo multidimensional de floats de 3x4 y especificamos al usuario lo que hace el programa y el orden de los datos a ingresar. Después gracias a 2 ciclos for anidados se hace la lectura de datos que ingrese el usuario, el primer ciclo for itera los renglones y el segundo las columnas. Y para comprobar que los datos se anexaron correctamente se imprime a pantalla igualmente mediante 2 ciclos for, los valores de la matriz, donde el primer ciclo for itera los renglones y el segundo las columnas.



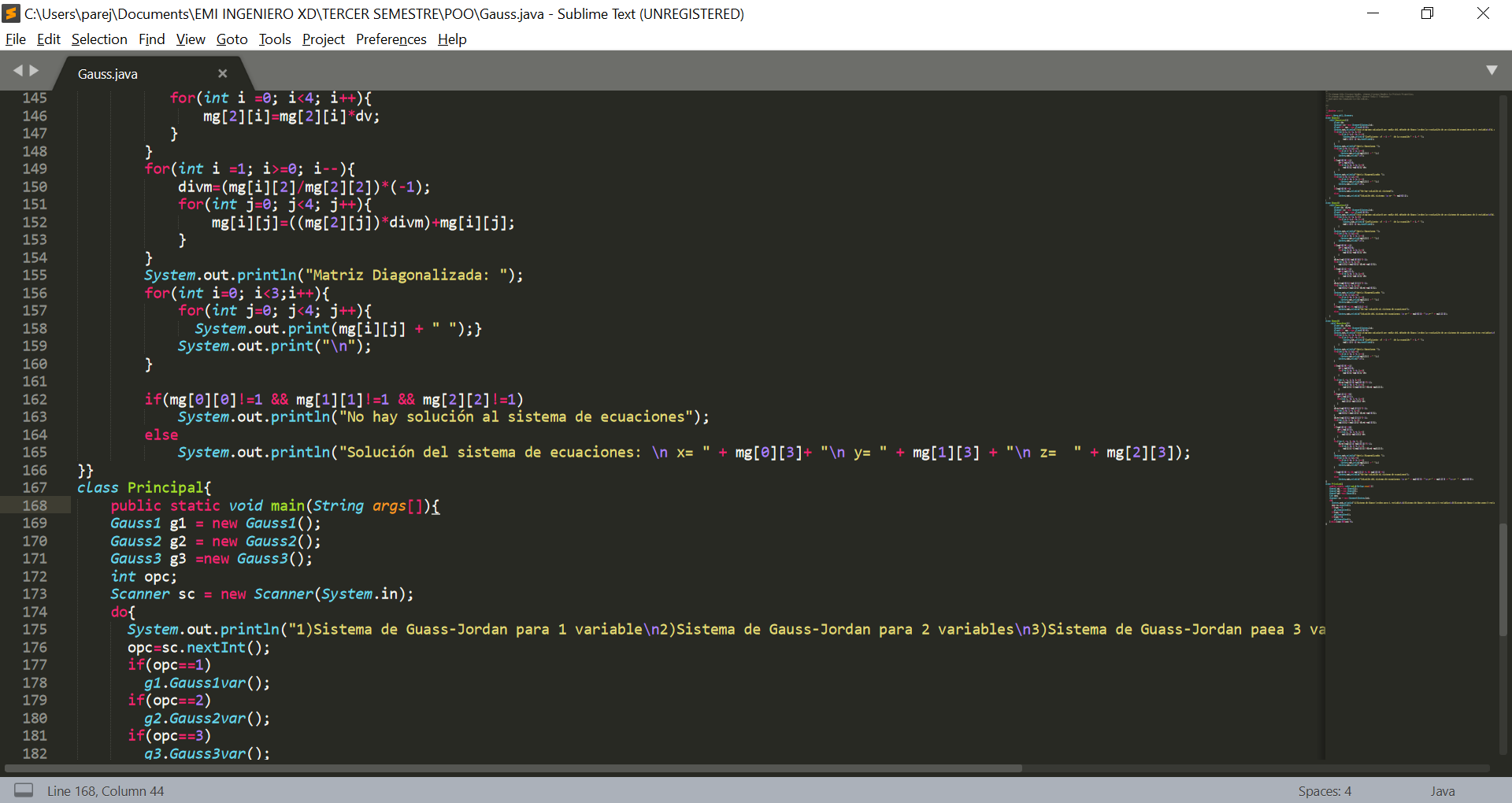
Una vez que ya se inicializa la matriz, lo primero que se se realiza en una estructura de control de flujo if, es verificar que el primer elemento del primer renglón de la matriz, sea igual a 1, si no es igual, entonces la variable flotante dv tomará el valor de 1 entre el valor del primer elemento del primer renglón de la matriz, y después mediante un ciclo for multiplicamos todos los elementos del primer renglón por dv, así ahora el coeficiente del primer elemento del primer renglón de la matriz ya es igual a 1. Después para hacer la primera eliminación Gaussiana, mediante un ciclo for que empieza en 2, a la variable divm se le asigna el valor resultante multiplicar el primer elemento del renglón que corresponda al número de iteración entre el valor del primer elemento del primer renglón todo por -1. Y en otro ciclo for que empieza en 1, se cambian los valores del segundo y tercer renglón, multiplicando los elementos del primer renglón por divm y lo que resulte se lo suman a los valores del renglón correspondiente a la iteración del primer ciclo for en los valores correspondientes a la iteración del segundo ciclo for. Finalizando este ciclo, tanto el primer valor del segundo y tercer renglón serán iguales a 0.



Ahora para realizar la segunda eliminación Gaussiana, lo primero que se hace es verificar si el segundo elemento del segundo renglón de la matriz sea 1, si no es igual, dv toma el valor de 1 entre el valor del segundo elemento del segundo renglón de la matriz, y para hacer el renglón equivalente, con un ciclo for se multiplica cada uno de los elementos del segundo renglón por dv. Después de que acaba este conjunto de instrucciones el segundo elemento del segundo renglón será igual a 1 y ahora para hacer la eliminación Guassiana sobre el primer renglón, divm toma el valor del segundo elemento del primer renglón entre el segundo elemento del segundo renglón por -1. Y mediante un ciclo for que empieza en 1, se cambiarán los valores del primer renglón, multiplicando los elementos que correspondan al número de iteración del ciclo for del segundo renglón por divm más los elementos que correspondan al número de iteración del ciclo for del primer renglón. Después para hacer la eliminación Gaussiana sobre el tercer renglón, divm toma el valor del segundo elemento del tercer renglón entre el segundo elemento del segundo renglón por -1. E igualmente que con el renglón anterior, con un ciclo for que itera de 1 a 4, se cambian los valores del tercer renglón, multiplicando el elemento que corresponde al número de iteración del ciclo for del segundo renglón por divm a lo que se le suma el elemento que corresponda al número de iteración del ciclo for del tercer renglón.



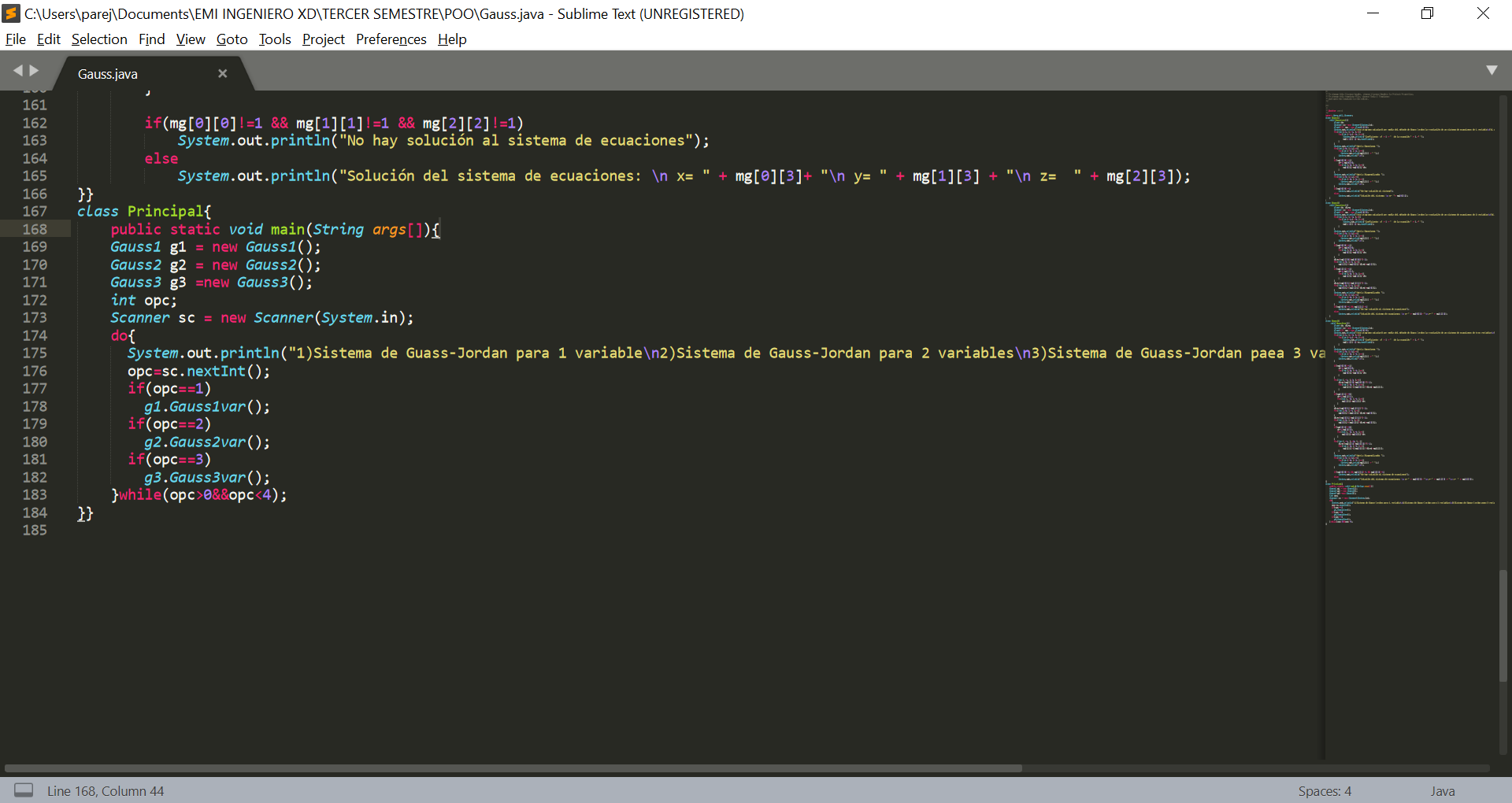
Y para terminar la diagonalización de la matriz, primero se verifica que el tercer elemento del tercer renglón de la matriz sea igual a 1, si no es igual, dv toma el valor de 1 entre el valor del tercer elemento del tercer renglón, y después con un ciclo for que itera de 1 hasta 4, multiplicamos cada uno de los elementos del tercer renglón por dv. Al finalizar este conjunto de instrucciones el tercer elemento del tercer renglón de la matriz será igual a 1, y luego en un ciclo for que va desde 2 hasta 1 divm toma el valor del tercer elemento del renglón que corresponda al número de iteración del ciclo for entre el valor del tercer elemento del tercer renglón de la matriz por -1, y en otro ciclo for anidado que va desde 1 hasta 4, se hacen los cambios en el primer y segundo renglón, donde el valor del elemento que corresponde a la iteración del primer ciclo for en el renglón que corresponde a la iteración del segundo ciclo for toma el valor del elemento que corresponde a la iteración del segundo ciclo for del tercer renglón por divm y a lo obtenido se le suma el valor del elemento que corresponde a la iteración del primer ciclo for en el renglón que corresponde a la iteración del segundo ciclo for. Después de que acaban estas instrucciones, la matriz ya se encuentra diagonalizada. Y se imprime a pantalla con 2 ciclos for anidados, donde el primero itera los renglones y el segundo las columnas.

Y para comprobar que el sistema tenga solución, verificamos que el primer elemento del primer renglón, el segundo elemento del segundo renglón y el tercer elemento del tercer renglón de la matriz sean iguales a 1, si esto no se cumple, significa que el sistema no tiene solución, si todos estos son iguales a 1, el cuarto elemento del primer renglón toma el valor de la primera incógnita, el cuarto elemento del segundo renglón es el valor de la segunda incógnita y el cuarto valor del tercer renglón es el valor de la tercera incógnita del sistema de ecuaciones, y todos estos se imprimen a pantalla.

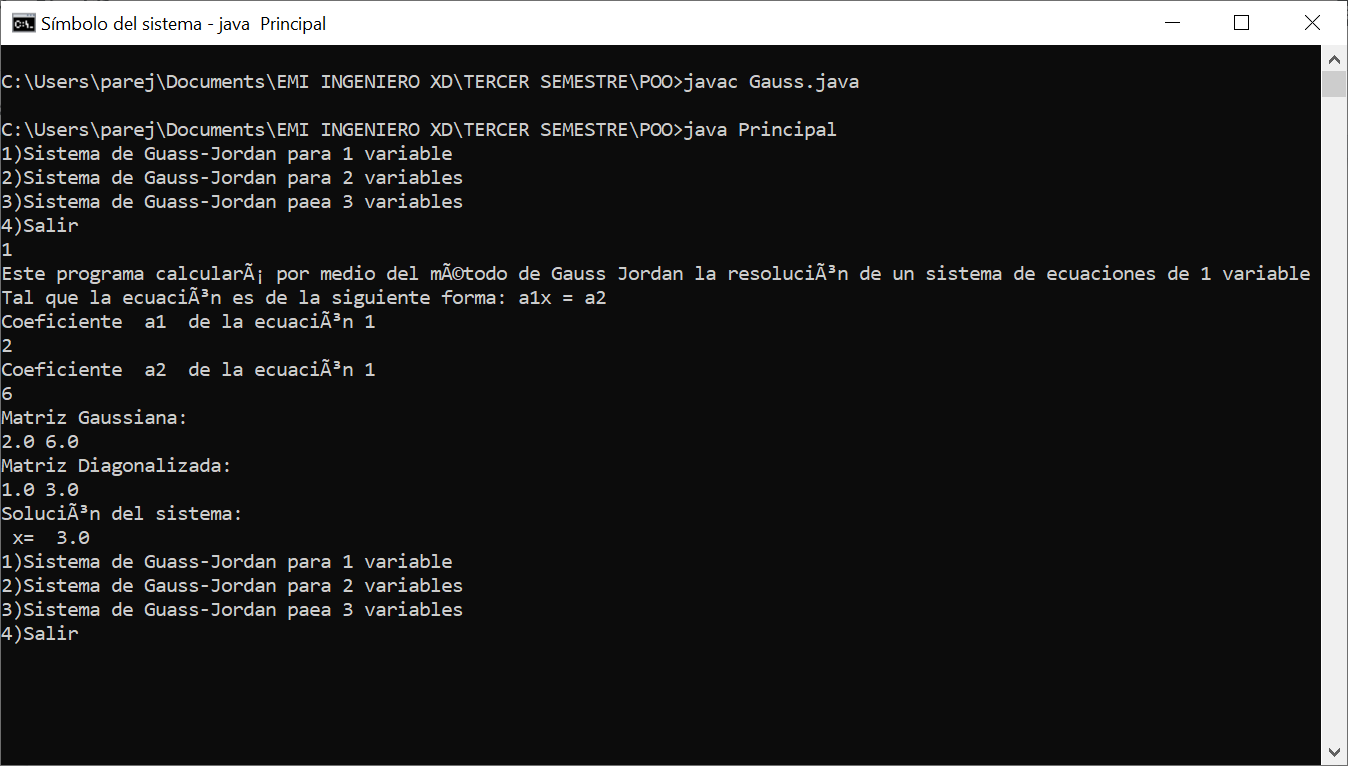
* **En la clase Principal:**



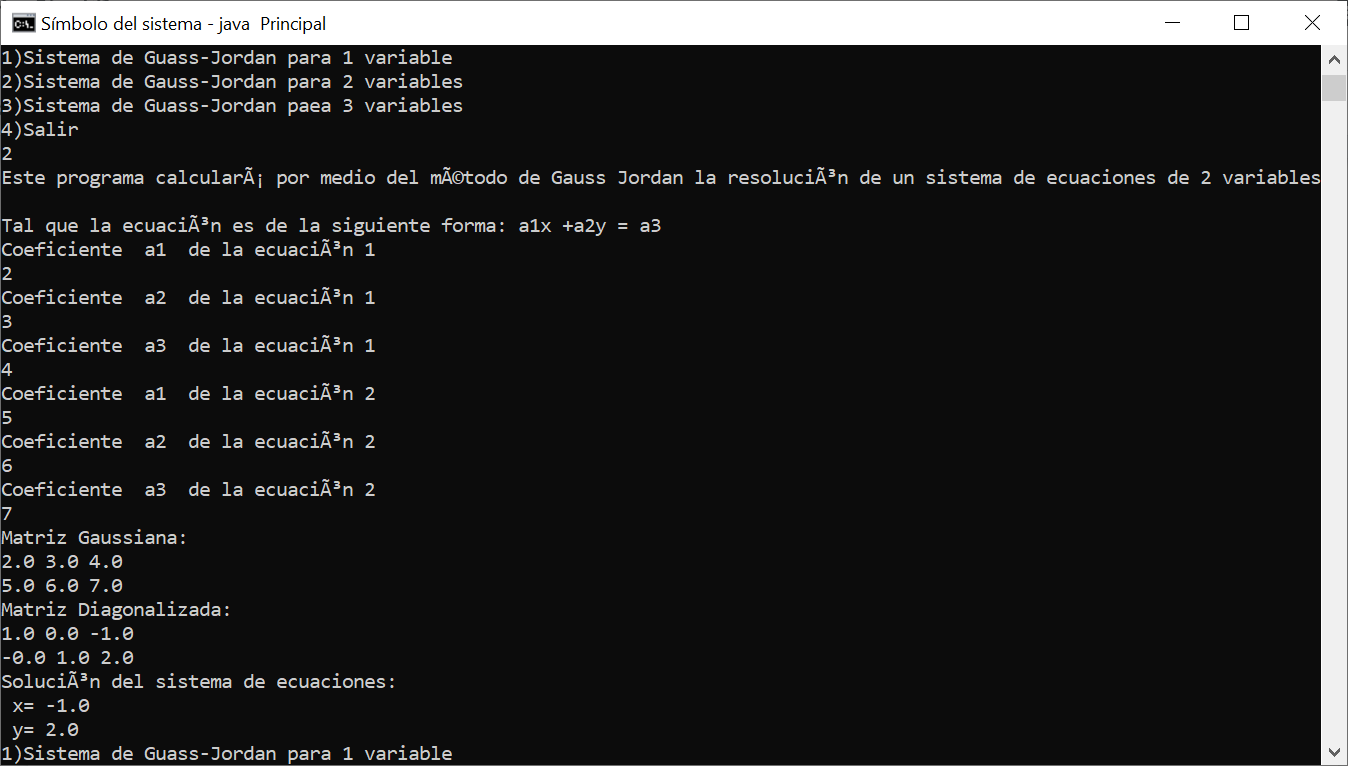
Para el diseño del main, primero decidimos que lo mejor para el usuario sería tener un menú, pero primero creamos 3 objeto para ejecutar la resolución de todos los sistemas de ecuaciones dependiendo del número de variables, entonces creamos un objeto de la clase Gauss1 llamado g1, otro objeto de la clase Gauss2 llamado g2 y un objeto de la clase Gauss3 llamado g3, después creamos una variable entera llamada opc que nos ayudaría al momento de que el usuario escogiera la resolución de un sistema de ecuaciones dependiendo del número de variables con las que contara y para leer la opción que escogería y que almacenaría la variable opc, creamos un objeto de la clase Scanner llamado sc.

Después para elaborar el menú en una estructura de repetición do-while, primer especificamos al usuario las opciones para realizar, donde si selecciona la opción 1 se hace la resolución de un sistema de ecuaciones por Gauss-Jordan para 1 variable, si selecciona 2, se hace la resolución de un sistema de ecuaciones por Gauss-Jordan para 2 variables, si escoge 3 se hace la resolución de un sistema de ecuaciones por Gauss-Jordan para 3 variables, y si escoge el número 4 o cualquier otra opción se abandona el menú. Después, con el método nextInt() del objeto sc leemos la opción que selecciona el usuario, si la opción es 1, hacemos uso del método Gaussvar1 del objeto g1, si la opción es 2, hacemos uso del método Gaussvar2 del objeto g2 y si la opción es 3 hacemos uso del método Gaussvar3 del objeto g3 y este menú estará en un ciclo mientras la variable opc sea mayor a 0 o menor a 4.

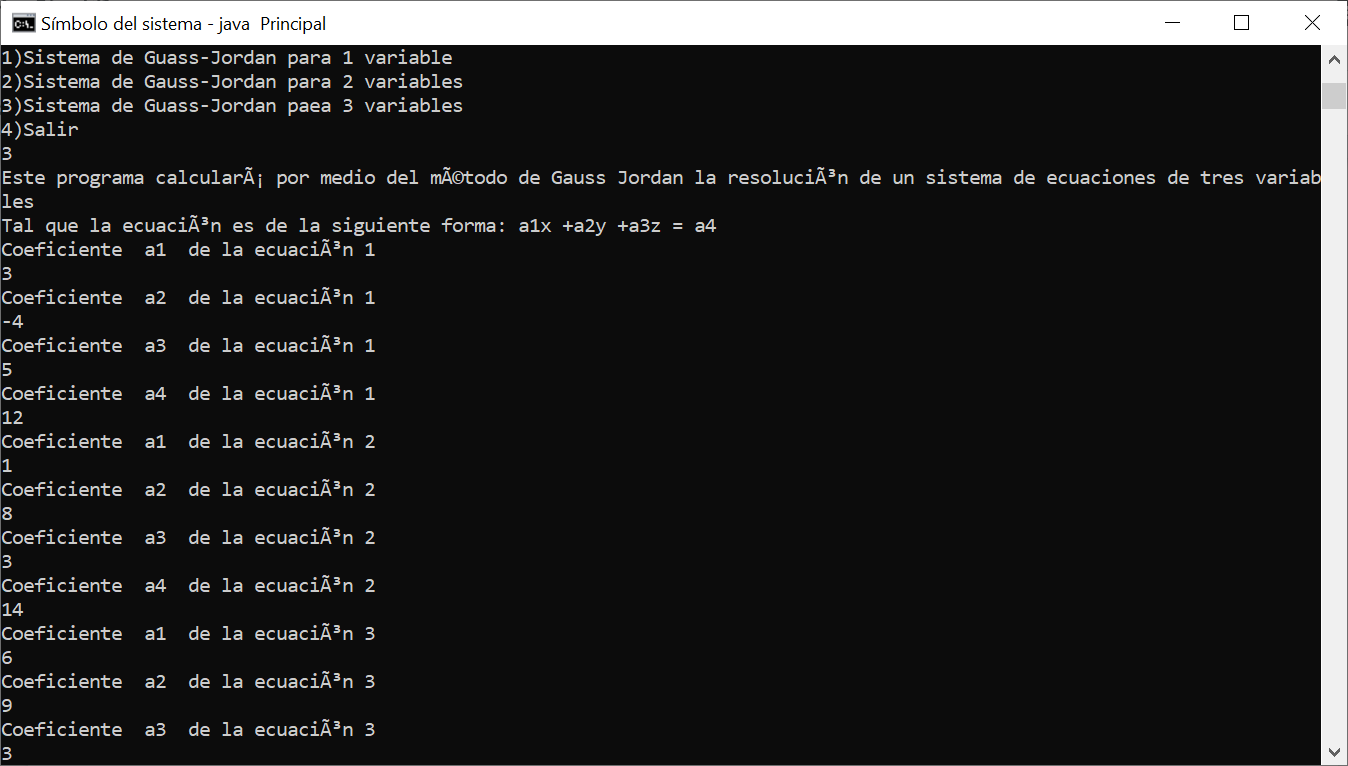
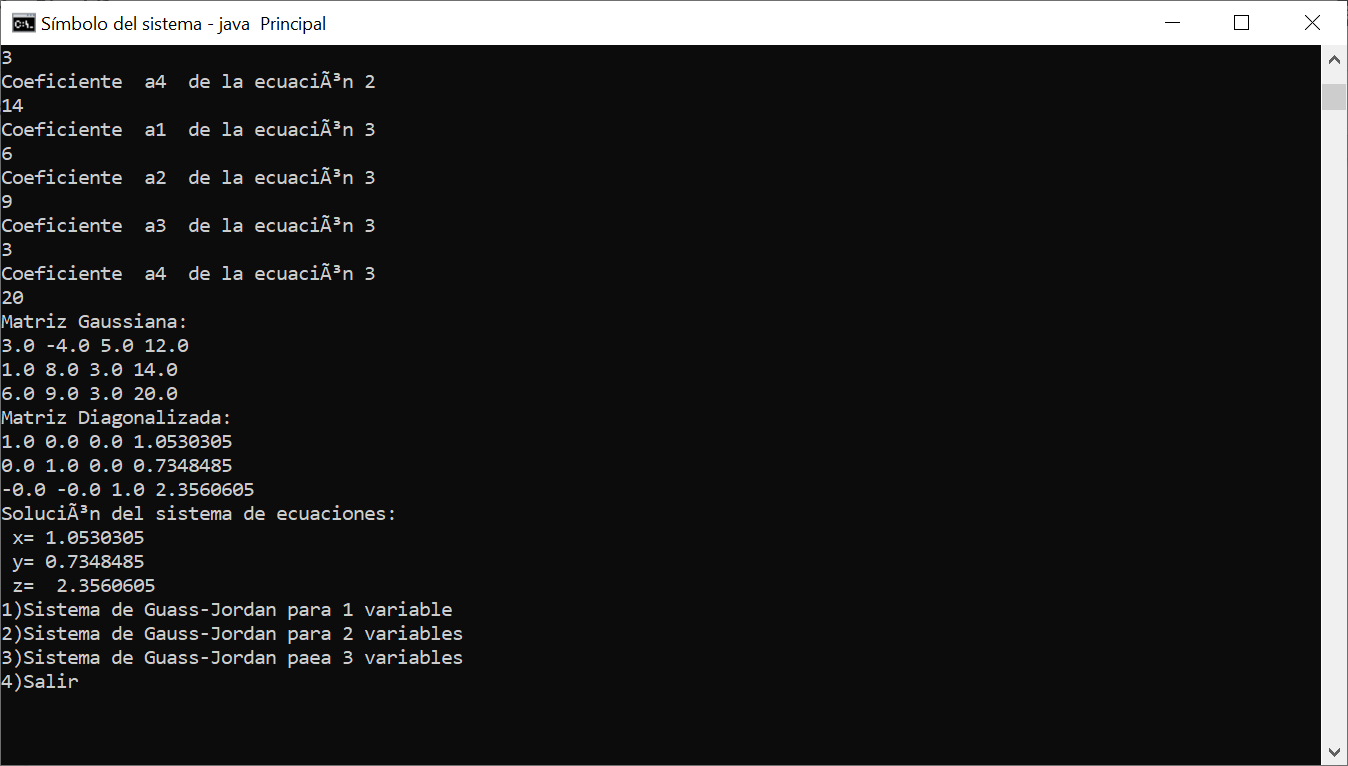
* **Ejecución del programa**

Gauss-Jordan para 1 variable:

Gauss-Jordan para 2 variables:



Gauss-Jordan para 3 variables:



**Ejercicio 3**

* **Análisis previo.**

El tercer ejercicio consiste en la elaboración de un programa que mostrase las propiedades del sistema, esto a través de la clase Properties.

Mediante la investigación de la clase se tiene que es una clase HashTable para las Strings, esto significa que tiene es una colección del tipo Hash, esto dice bastante para poder determinar los métodos que utiliza, ya que para obtener las propiedades se utiliza el método System.getProperties, el cual devuelve la tabla tipo Hash en la cual se encuentran las propiedades del sistema y su significado (Imagen 1).

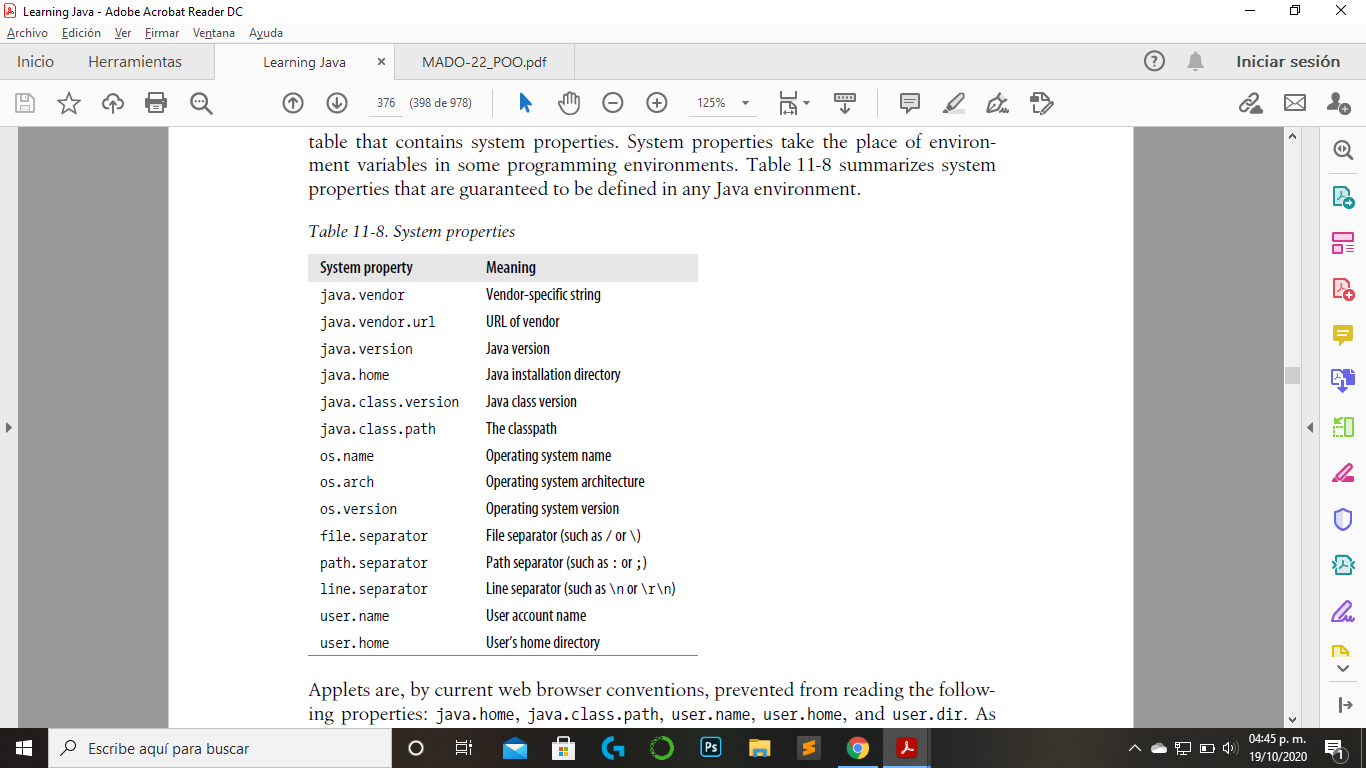
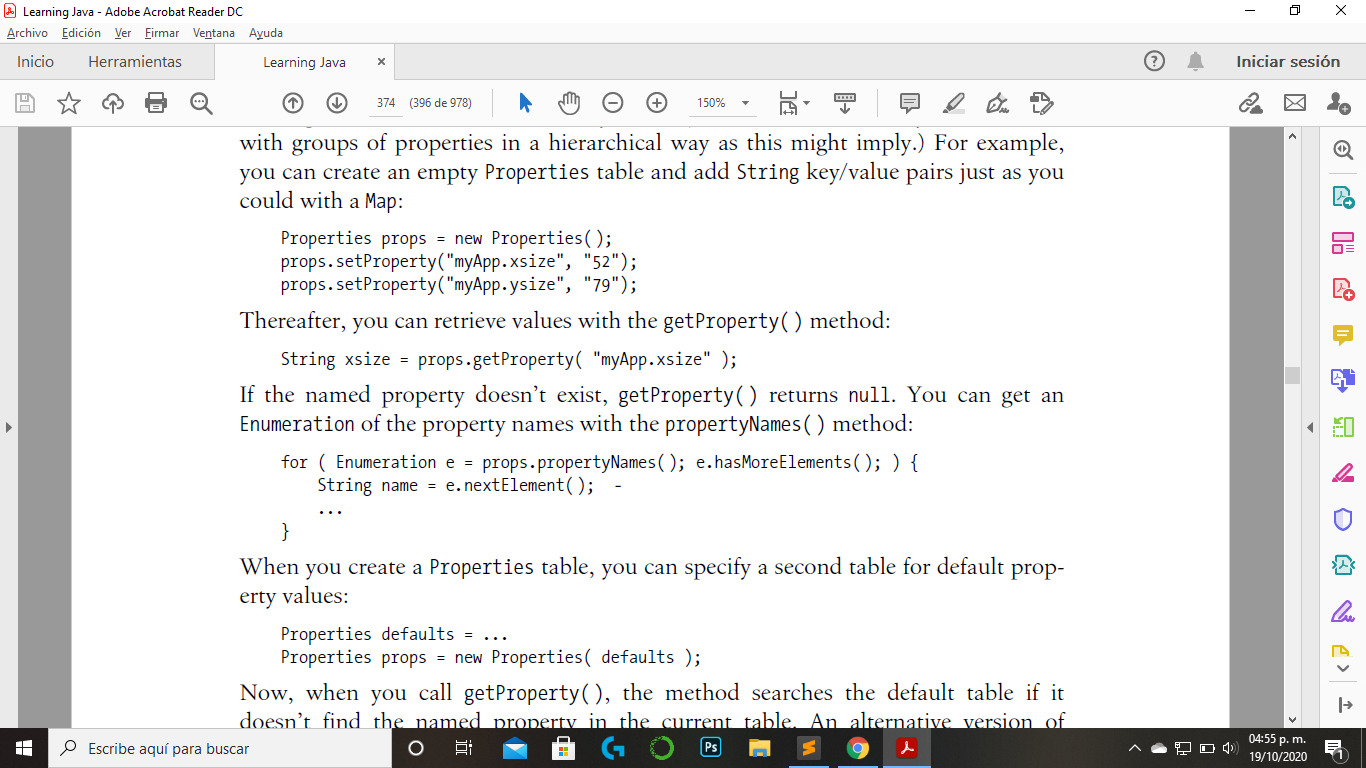


Imagen 1

Como es una clase importada es necesario crear una instancia de la clase para poder utilizar sus métodos, y para obtener las propiedades del sistema. Al ser una tabla de tipo Hash no es posible iterar por cada elemento, como en los arreglos, únicamente se puede acceder por medio de las claves o llaves, esto se realizar mediante el Enumeration que utiliza las llaves, y cuando se tienen las llaves se puede acceder al valor, que es el significado, esto es por medio del método propertyNames() y por medio del método getProperty() se obtiene el valor asociado a la llave.

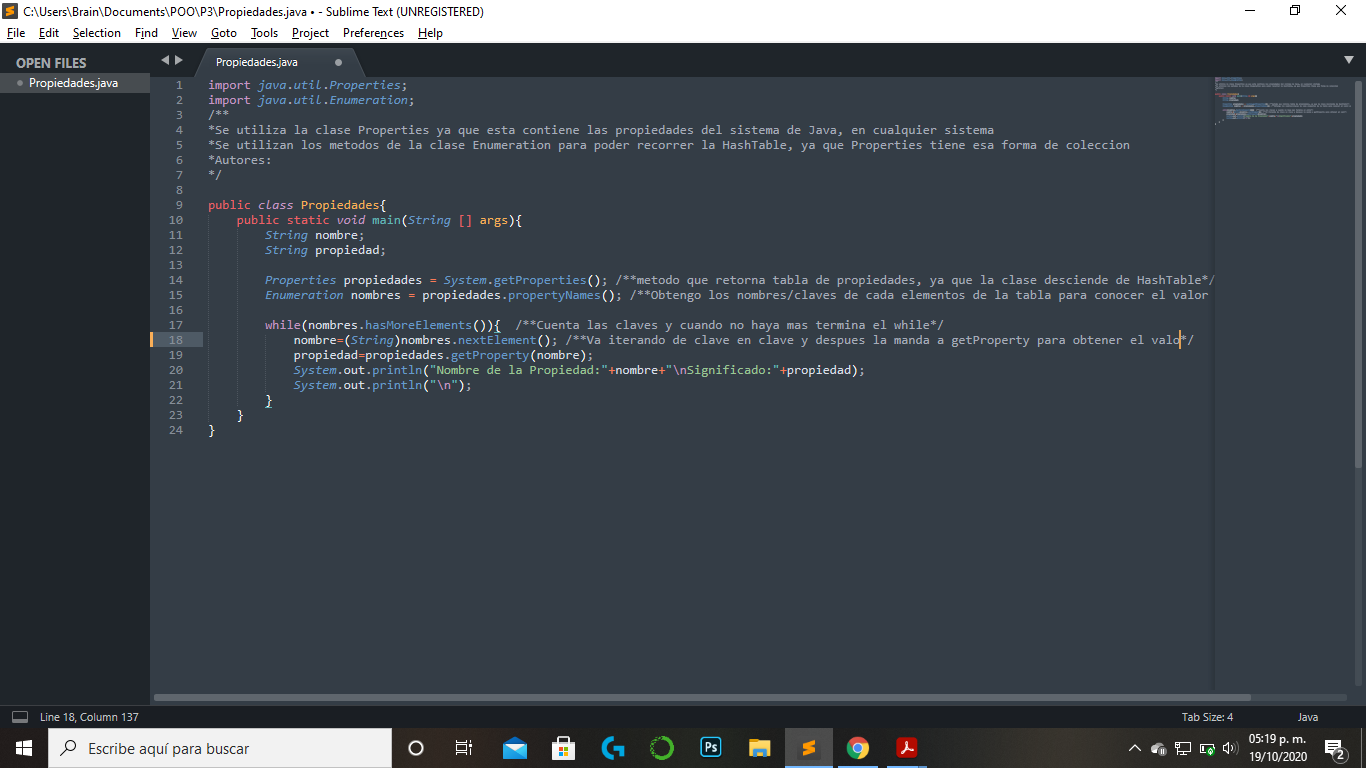


Obtención de las llaves

* **Desarrollo**

Lo que realizamos en el código fue la implementación de la clase Properties y también se utilizó la interfaz Enumeration para poder iterar en la HashTable de Properties, entonces lo primero que se realizó fue las importaciones, después la creación de la clase public y la creación del método principal.

Se declara una instancia de Properties para poder obtener la tabla de propiedades y por medio de Enumeration se obtenga se obtengan las claves de la tabla, este punto es importante porque se necesita tener la instancia de properties de lo contrario no se podria ya que no se tendría un acceso a la tabla de propiedades.



Se realiza un while para poder iterar y dentro de la condición del while se coloca el método de Enumeration hasMoreElements(), el cual regresa un booleano en caso de que se cumpla la condición la cual verifica que existan llaves en Enumeration.

Dentro del While sucede lo más importante ya que se obtienen las claves y se asignan a una variable de tipo String por medio del nextElement(), porque son nombres de las propiedades y en este caso Properties es una HashTable especializada para Strings, por eso se asigna una String.

Por otro lado se asigna a otra variable String el getProperty, el cual por medio la primera variable de tipo String se podrá acceder al valor que corresponda con la clave, ya que la primera variable contiene la clave y al ser mandada como argumento de getProperty se solicita el valor correspondiente a la clave (Imagen 2).

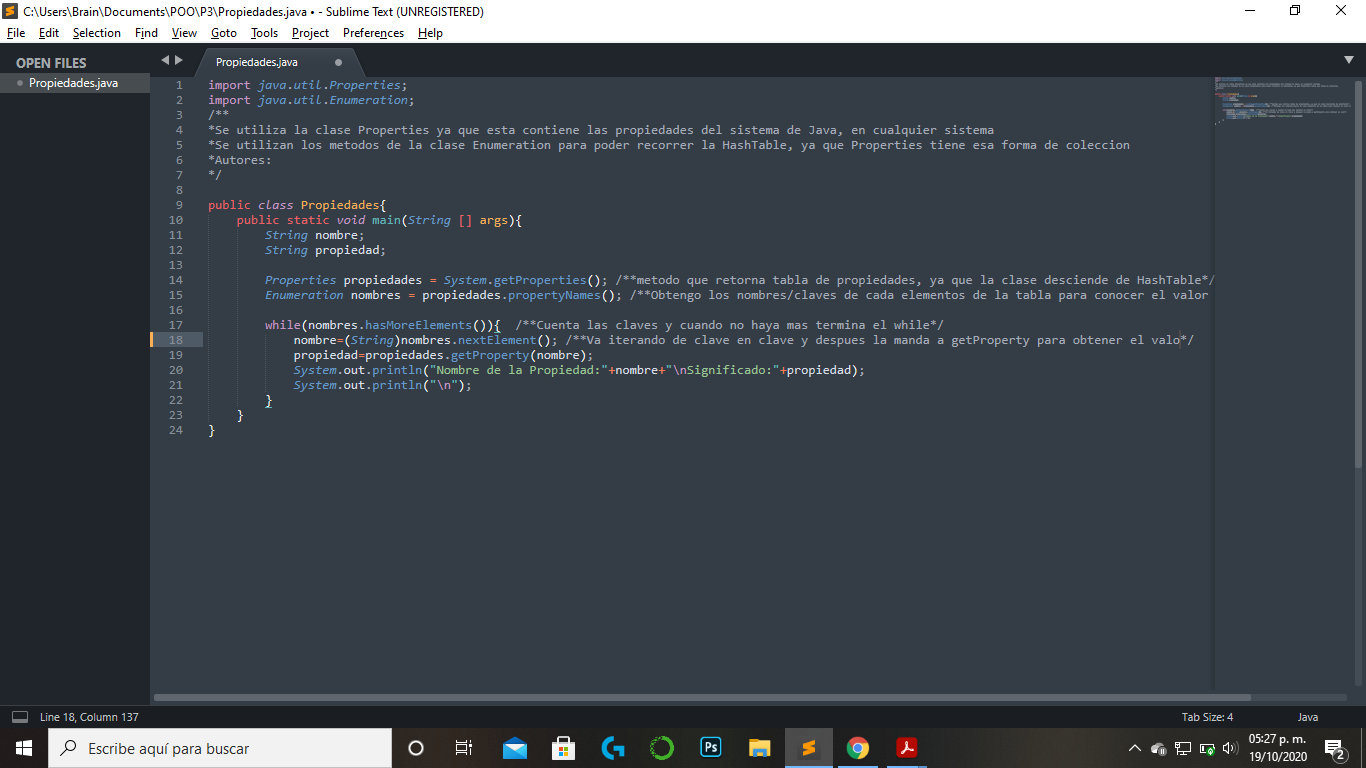
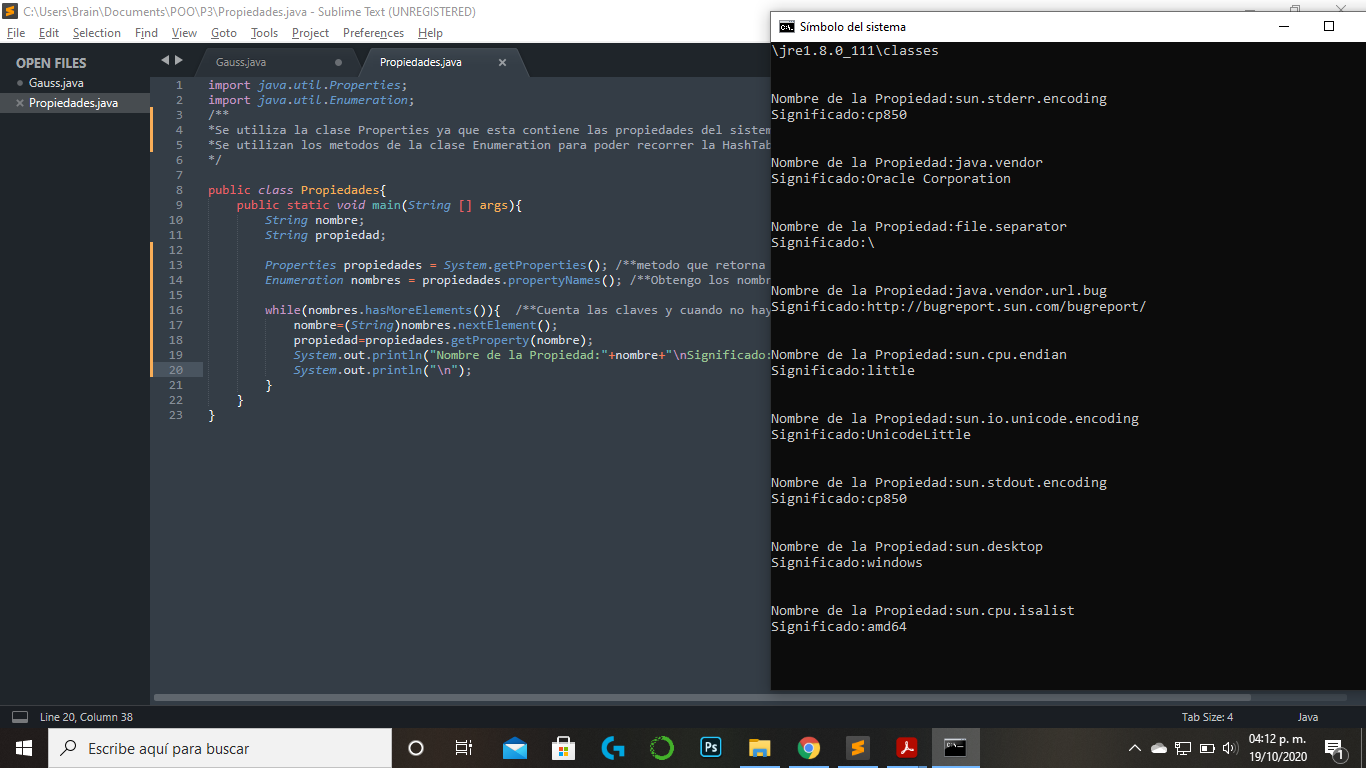


Imagen 2

Por último se envían a imprimir los valores, tanto de las llaves como del contenido que corresponde al valor de la llave.

* **Ejecución del programa**



**Conclusiones**

* Dávila Ortega Jesús Eduardo:

Durante la elaboración de los distintos programas se pudo observar la utilización de distintas bibliotecas propias del lenguaje, gracias a el implementar distintas bibliotecas del lenguaje me pude dar cuenta de cómo Java posee métodos dentro de sus bibliotecas parecidos a algunas funciones de biblioteca de las bibliotecas de C pero en algunos casos hay métodos en Java los cuales son desconocidos para mí y no soy muy hábil al momento de manejarlo y mediante la practica aprendí como utilizarlas y me ayudó a mejorar y entender el funcionamiento del lenguaje Java.

* Díaz Hernández Marcos Bryan:

De acuerdo con el objetivo de la práctica los ejercicios que elaboramos consisten en implementar las bibliotecas del lenguaje Java, estas tienen distintas funciones y eso se demuestra al implementarlas en distintos ejercicios como lo es el primero que resuelve números complejos, el segundo ecuaciones y el tercero obtiene las propiedades del sistema, todo por medio de métodos de distintas clases que vienen dentro de las bibliotecas de Java, las cuales proporcionan una herramienta muy útil para poder resolver problemas.

* Pareja Ávila Emiliano:

Se cumplieron los objetivos de la práctica, ya que aprendí mediante varios programas a utilizar las diferentes utilerías que nos proporciona Java, además de java.lang y también sus diferentes operaciones, sus métodos sobrecargados, y sus valores de retorno. Además con estas utilerías me puedo ayudar a hacer códigos mucho más simplificados y que cuando haga tareas que involucren hacer funciones complejas, me pueda ayudar de una de estas bibliotecas para hacer menos difícil dichas tareas.

* Vázquez Zavala Oliver Alex:

Conforme al desarrollo de la práctica es visible la frecuencia con que se realizan operaciones que son muy comunes al elaborar un programa, tales como elevar al cuadrado un número o el comparara una colección con otra, para esto es tarea del desarrollador verificar si existen clases que ayudan a realizar dichas operaciones ya que estas clases pueden ayudarnos a mejorar nuestros programas, ahorrando tiempo, reutilizando código y evitando errores, debido a que dichas clases han sido probadas completamente para asegurar su eficiencia, esta práctica me ha dejado conocer más sobre las bibliotecas y algunas clases propias del lenguaje de programación java, las cuales me serán de gran utilidad al momento de desarrollar un programa.

**Bibliografía**

* Patrick N. (2005). Learning Java. USA: O’Reilly.