

Facultad de Ingenierías en Electricidad y Computación



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

2015 - 2016

|  |  |
| --- | --- |
| http://ocw.upm.es/lenguajes-y-sistemas-informaticos/estructuras-de-datos/contenidos/estructurasDeDatos.jpg  Estructura de datos | aLUMNos:  guillermo bernal  Carlos cedeño  bryan Ordóez  Paralelo: 2  Docente: Ing. Jorge Rodríguez Echeverría  I PARCIAL – II TÉRMINO |

# **Identifique los TDA’s presentes en el proyecto y explique porque los creó.**

Entre los TDA’s presentes en nuestro proyecto, hay que hacer una diferenciación entre los cuales fueron creados y cuales fueron implementados.

De los TDA’s que fueron creados hay solo 1:

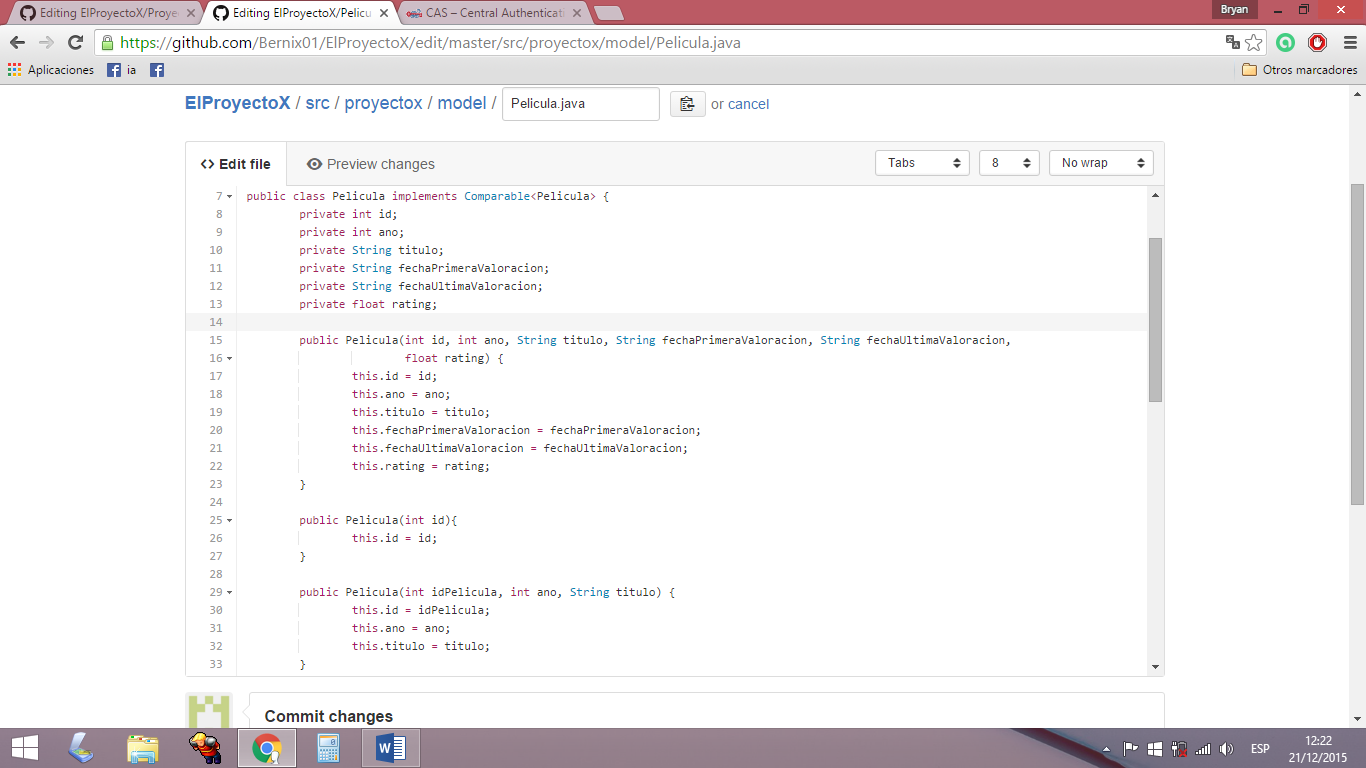
* Película:

El TDA implementa Comparable por su ID. Este Comparable nos ayudará mucho luego cuando comencemos a ordenar los datos para el debido funcionamiento de la aplicación del proyecto, por ahora solo señalaremos que está.

Las características o parámetros que forman parte de Pelicula son:

* ID: Es la ID asignada a cada Película, para que cada una sea única aun cuando exista otra película con el mismo nombre.
* Año(Ano): Contiene el año en que una película fue estrenada. Debido a que los datos contenidos son entre 1998-2005; en los datos no hay películas más allá del 2005.
* Titulo: Contiene el nombre o título de una película. Pueden haber similitudes entre los títulos de varias películas

Estas tres últimas características fueron puestas luego de que se encontraran muchos problemas en el tiempo de ejecución. Estos parámetros son un resumen de otro posible TDA llamada Calificación que fue eliminado para que el programa tenga mayor rapidez.

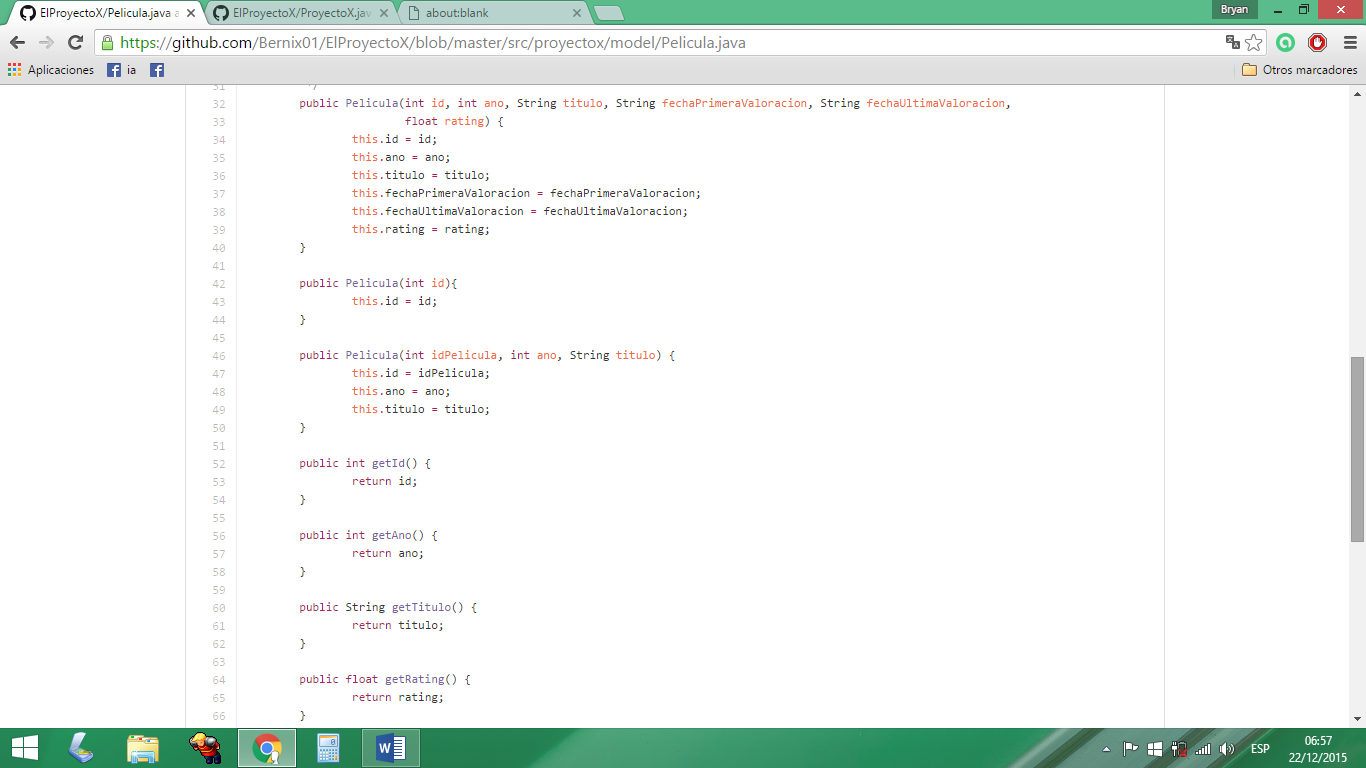
* FechaPrimeraValoración: Contiene la primera valoración hecha por un usuario a la película.
* FechaUltimaValoración: Contiene la última valoración hecha por un usuario a la película.
* Rating: Contiene un promedio de las valoraciones hechas por lo usuarios a una determinada película. 

Ahora, los métodos presentes en el TDA película son:

* Método Constructor de 6 parámetros:

Public Película(int id, int ano, String Titulo, String fechaPrimeraValoracion, String fechaUltimaValoracion, float rating)

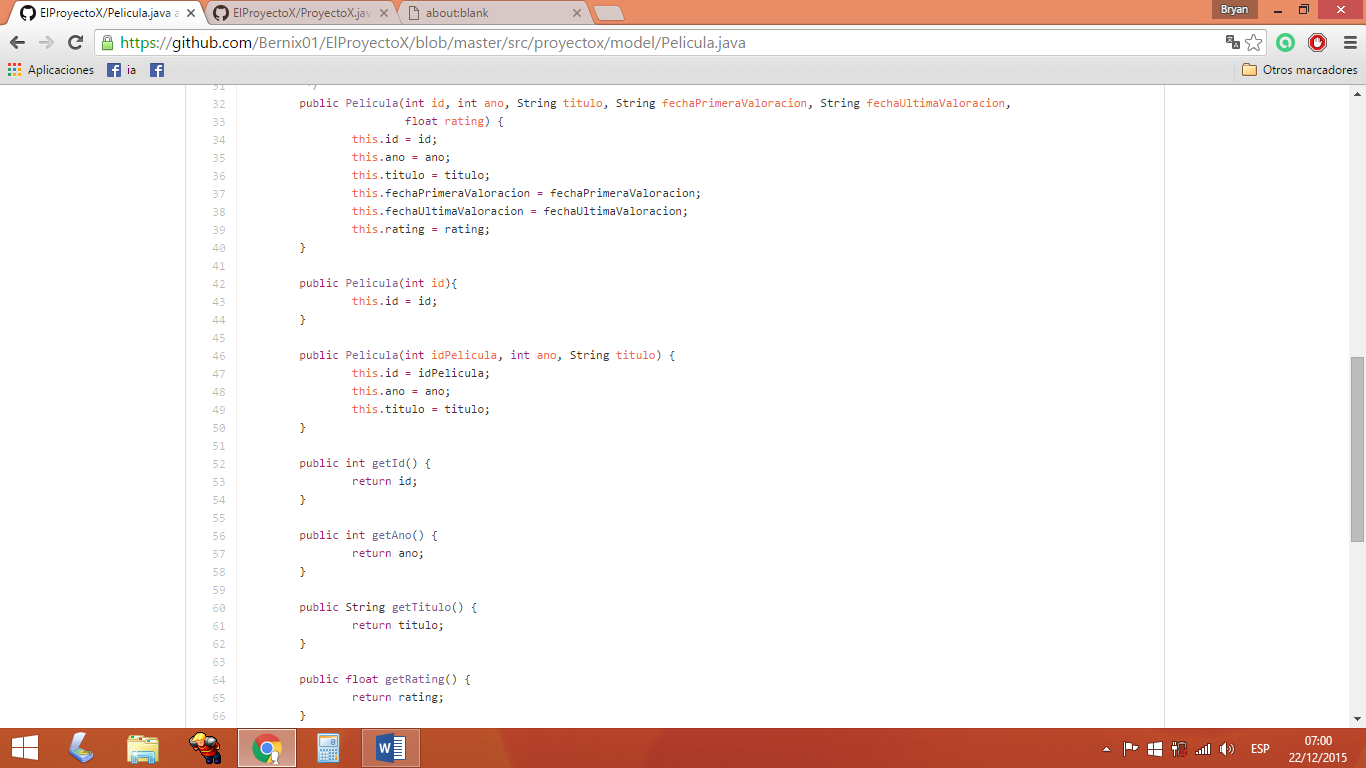
Este método constructor es el general, donde se asume que las películas contienen todos los datos necesarios. Sin embargo, no todos las películas tienen todos los datos, algunas les falta el año de emisión, a otros le faltan valoraciones, etc.



* Método Constructor de 1 parámetro

public Pelicula(int idPelicula)

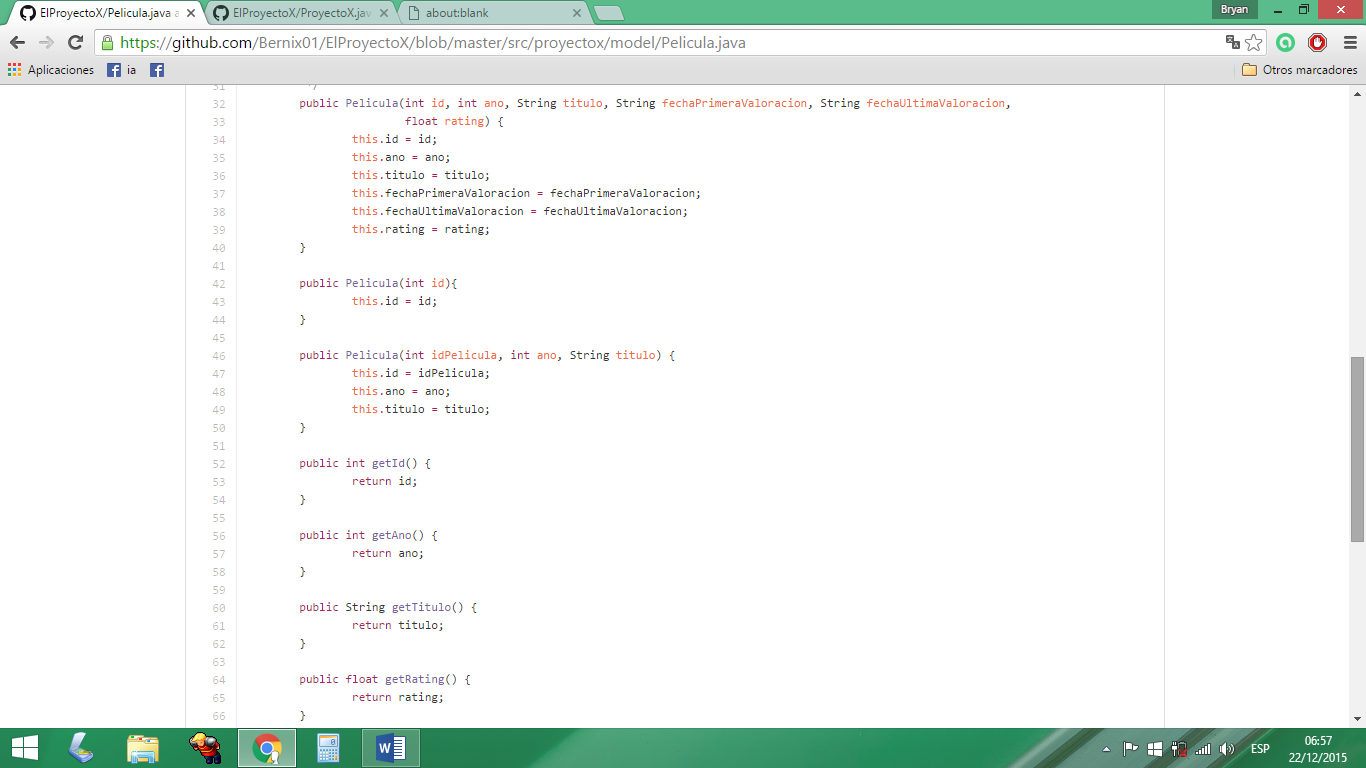
Este método solo recibe el parámetro IdPelicula. No contiene nombre de película, ni año de emisión, mucho menos rating y valoraciones



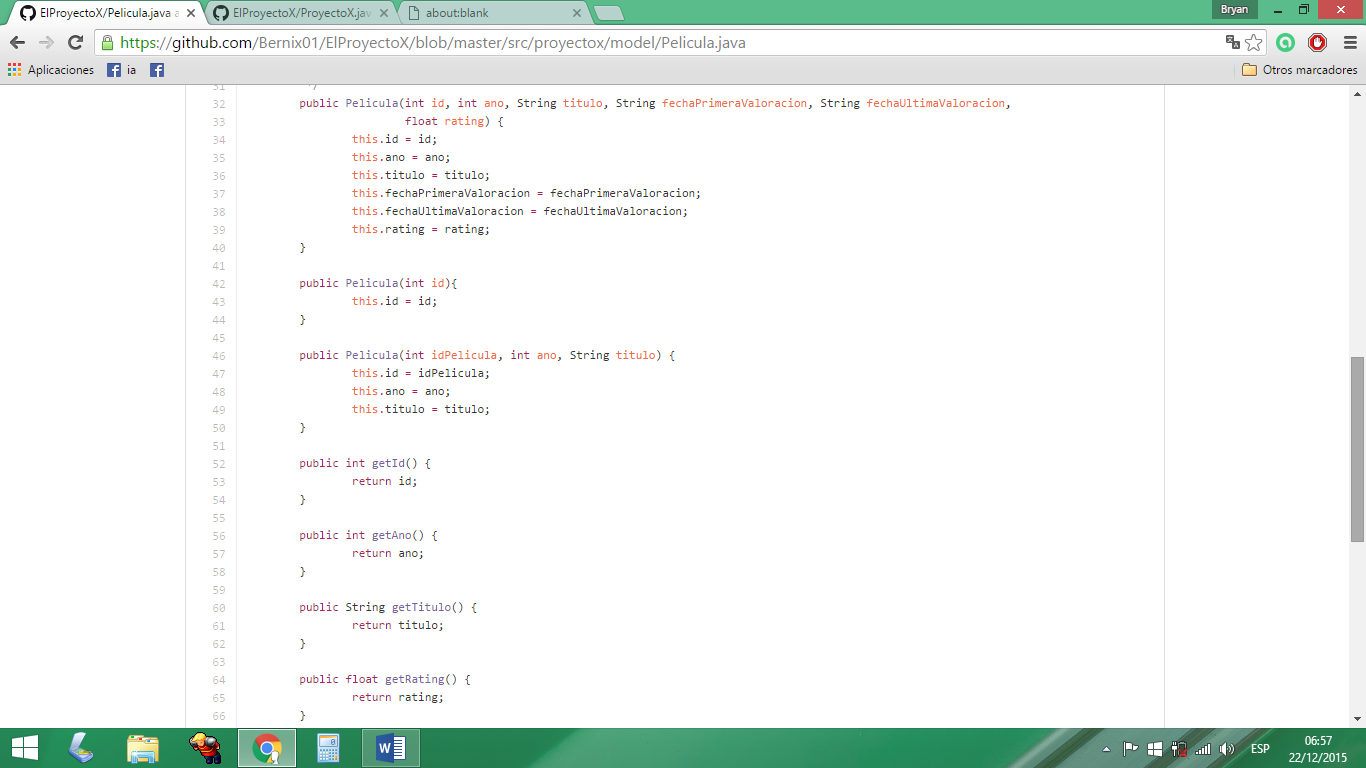
* Método Constructor de 3 parámetros

public Pelicula(int idPelicula, int ano, String titulo)

Este método recibe únicamente estos 3 parametros debido a que nunca fue valorado, por lo tanto carece de String fechaPrimeraValoracion, String fechaUltimaValoracion, float rating.

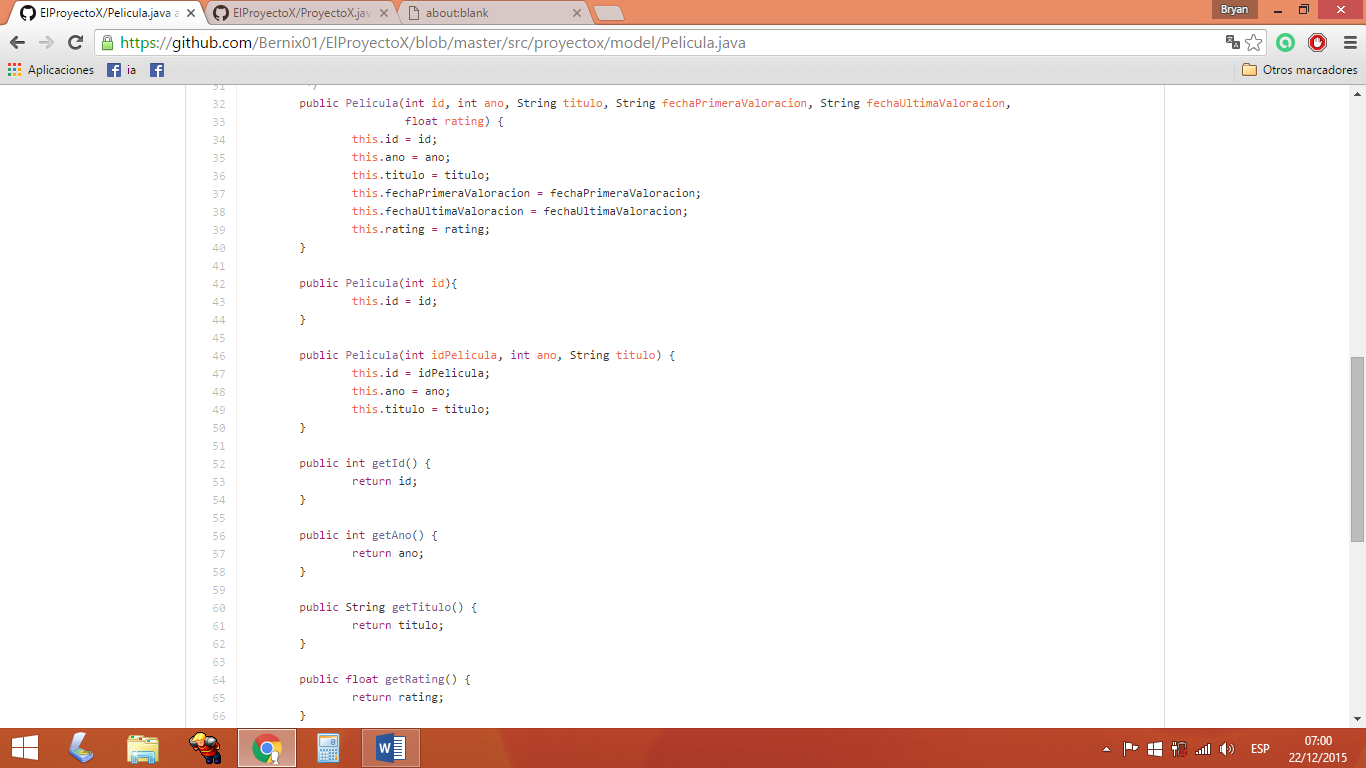


* GetId()

Este método permite obtener el dato privado Id de película. 

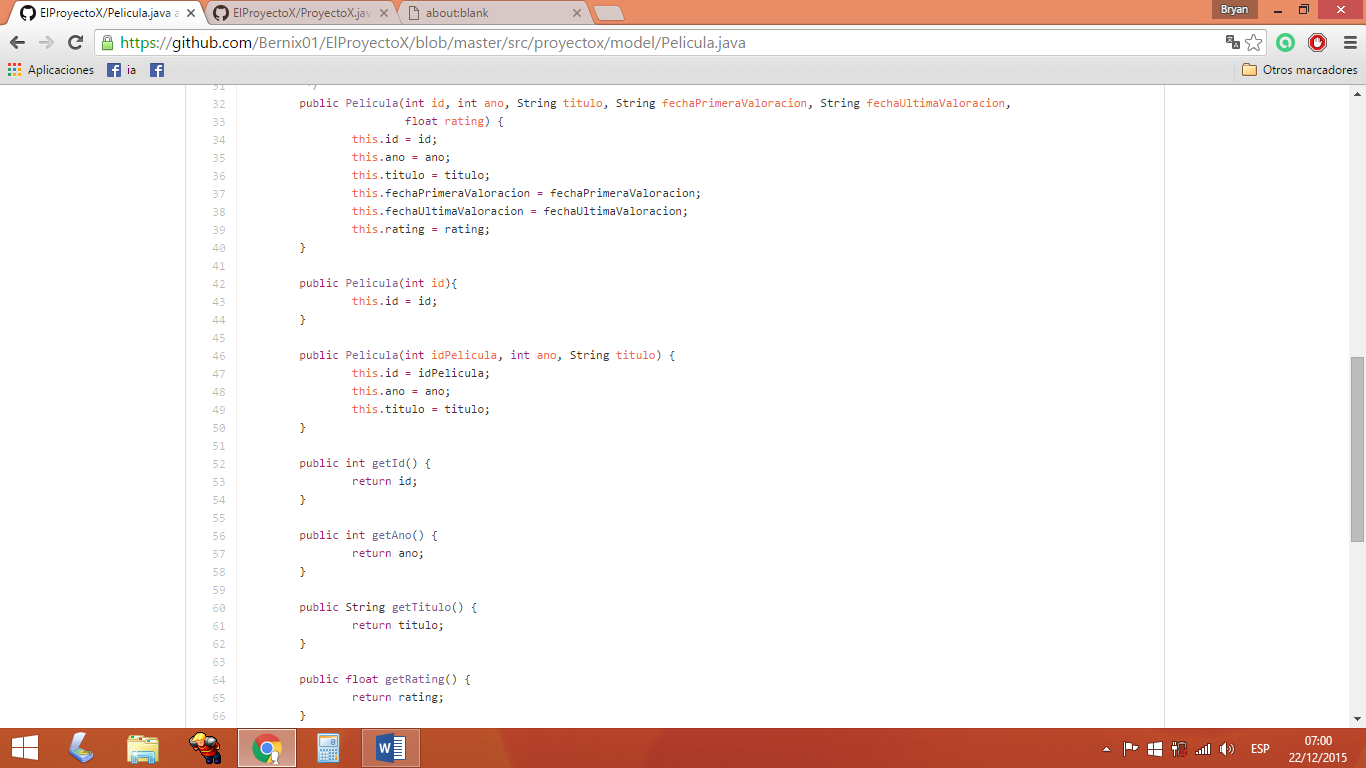
* GetAno()

Este método permite obtener el dato privado Año(ano) de película.



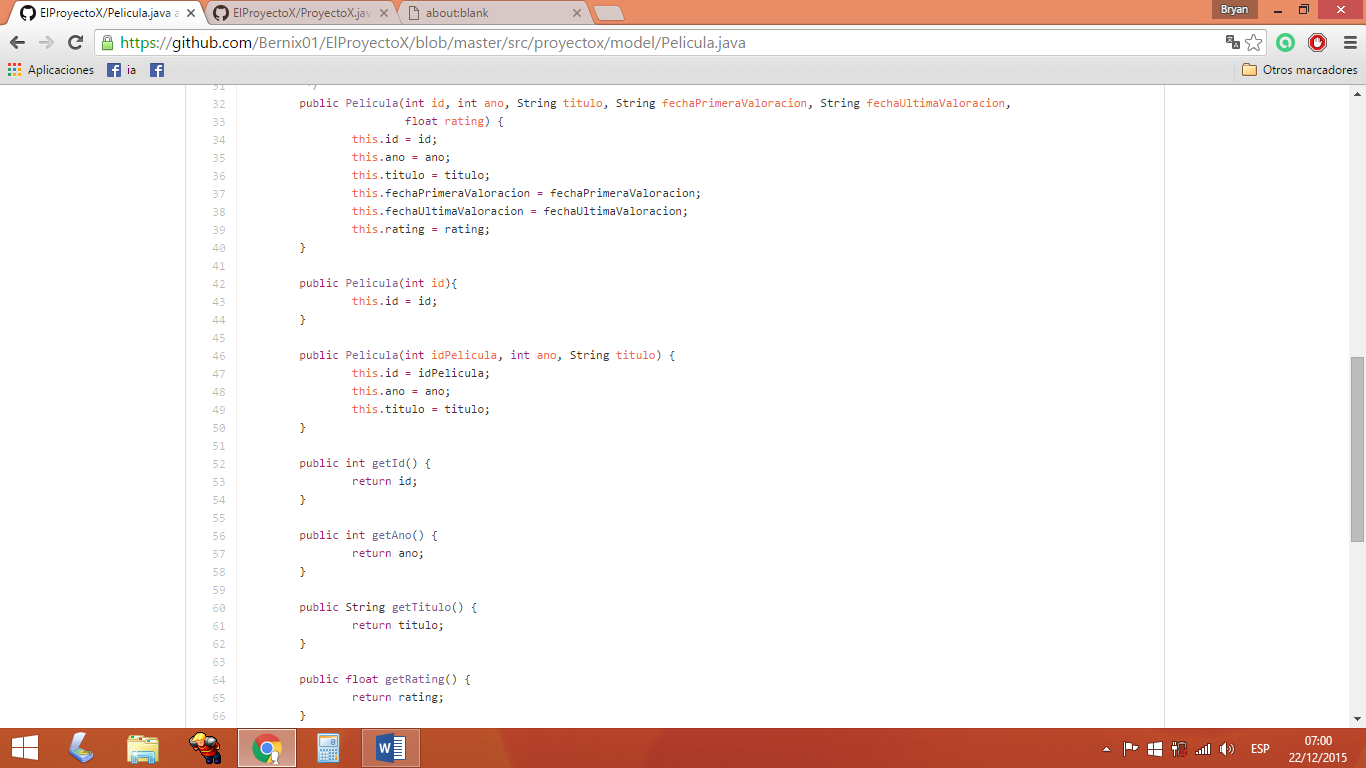
* GetTitulo()

Este método permite obtener el dato privado título de película.



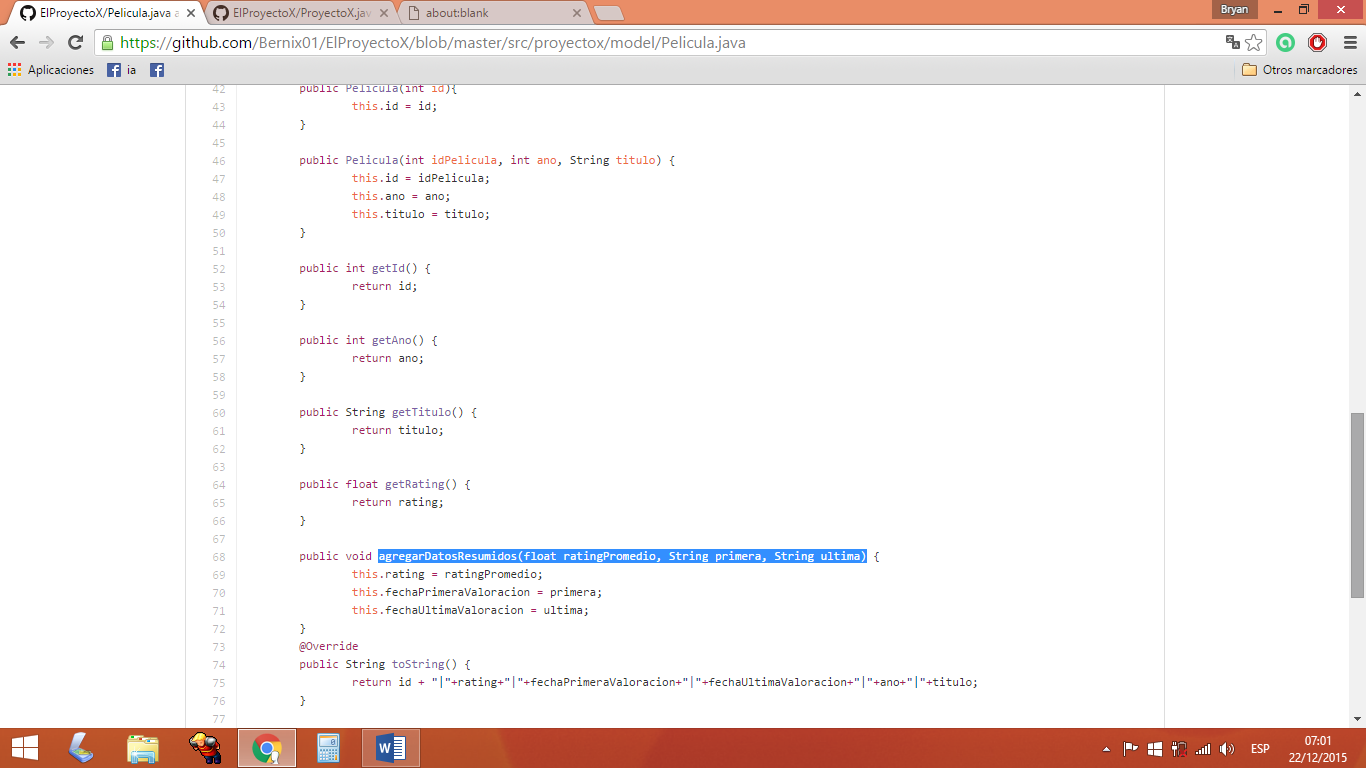
* GetRating()

Este método permite obtener el dato privado Rating dentro de película.



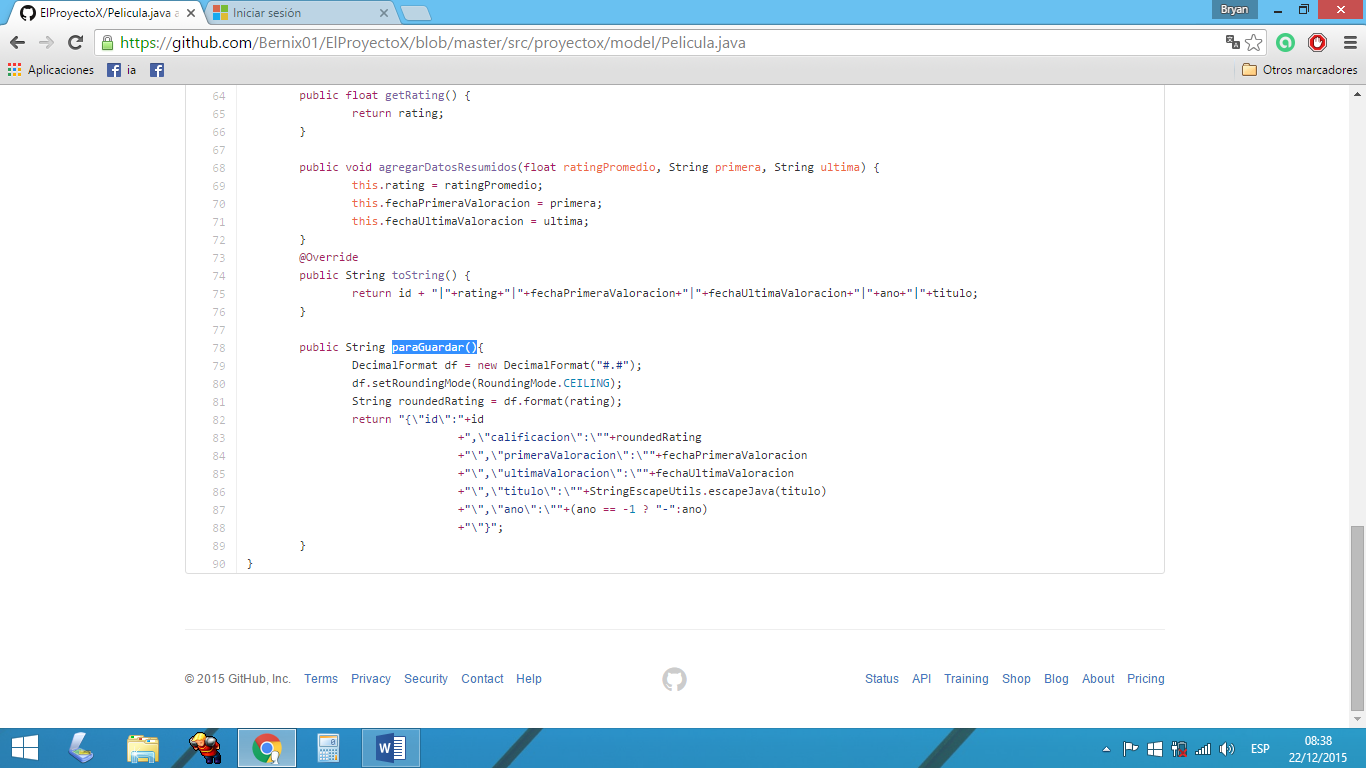
* AgregarDatosResumidos(float ratingPromedio, String primera, String ultima)

Este método permite ingresar los datos resumidos de Rating, sacando un rating promedio las calificaciones en el archivo de datos, además de la primera y ultima valoración.



* paraGuardar()

Este método genera el objeto en formato .json que posteriormente es utilizado en el reporte html.



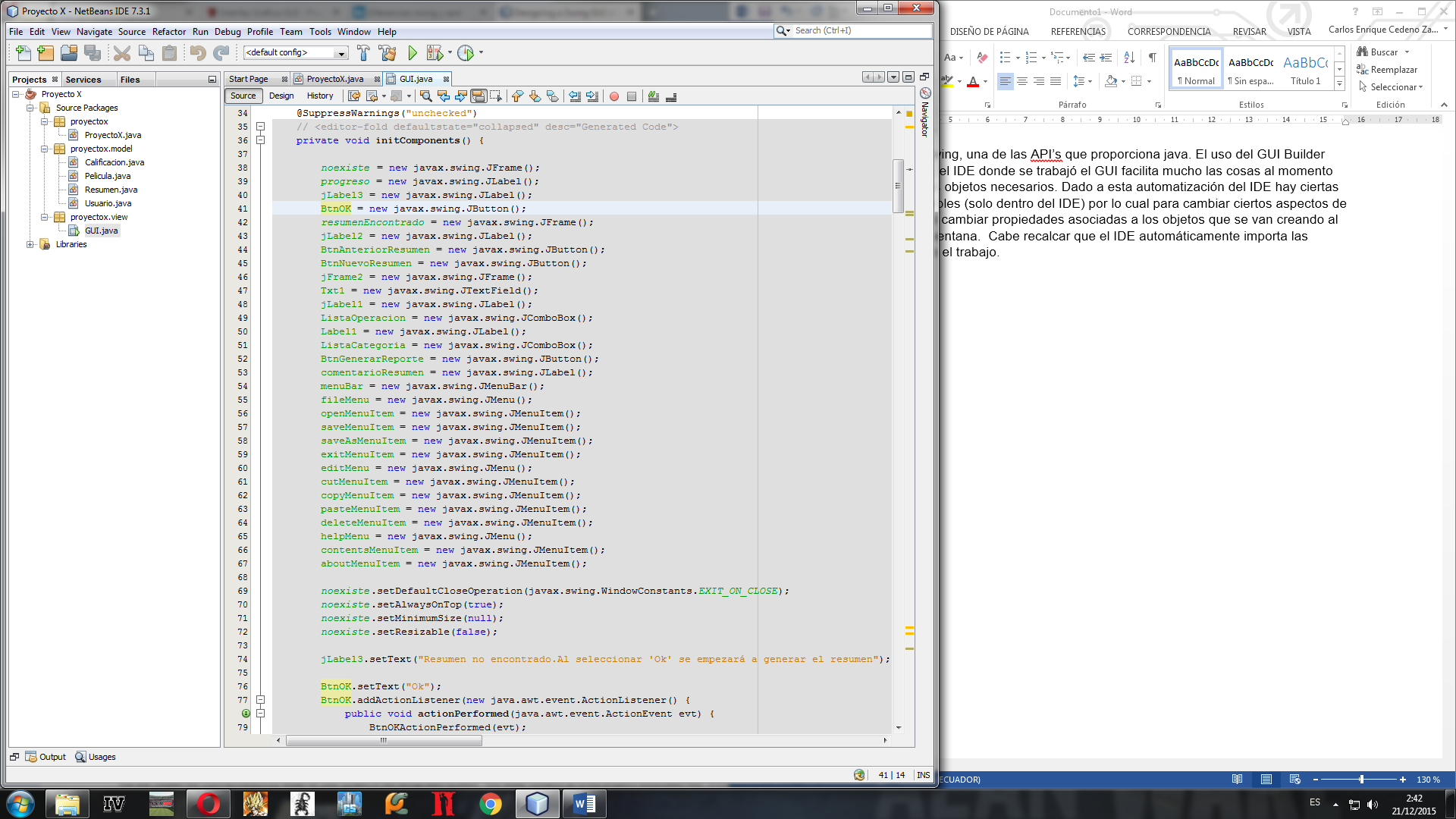
El TDA películas es el más importante pues contiene los datos en bruto de lo que será nuestra aplicación. En un principio estaban previsto crear 3 TDA’s que serían: Película, Usuario y Calificación, pero tras diversos problemas con el tiempo de ejecución de la aplicación por el exceso de datos, se decidió eliminar el TDA Usuario y resumir el TDA calificación en los parámetros antes descritos.

# **Mostrar y detallar los conceptos de la materia que se encuentran implementados en su código.**

* **Lista de Arreglos (ArrayList):** Los datos de las películas están ordenados por el TDA ArrayList. Se ingresaron todos los datos de las películas en bruto en una lista de arreglos de orden N, donde N es el número total de películas.
* **Lista(LinkedList):** Los linkedList fueron utilizados para el ordenamiento de películas para las diferentes búsquedas
* **Binary Search:** El método de Binary Search busca en el arreglo dado de enteros para el valor especificado utilizando la búsqueda binaria gama algorithm. El arreglo debe estar ordenado antes de hacer esta llamada. Si el arreglo no está ordenado, los resultados están indefinidos
* **Interfaz Gráfica GUI (Graphical User Interface)** es el conjunto de componentes gráficos que posibilitan la interacción entre el usuario y la aplicación. Es decir ventas, botones, listas, cajas de diálogo, campos de texto, etc.

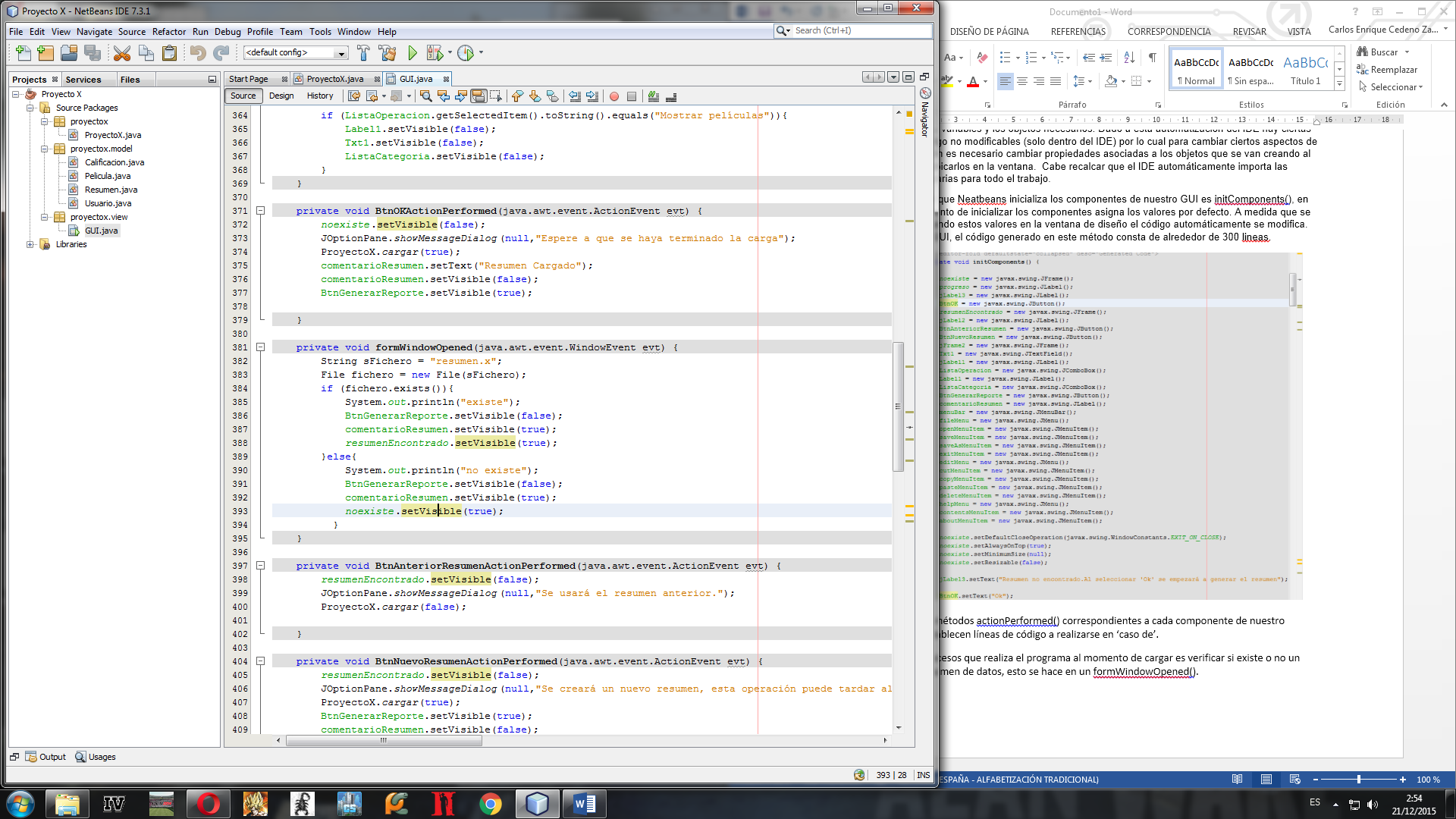
Para nuestro proyecto usa Swing, una de las API’s que proporciona java. El uso del GUI Builder proporcionado por Netbeans, el IDE donde se trabajó el GUI facilita mucho las cosas al momento de declaras las variables y los objetos necesarios. Dado a esta automatización del IDE hay ciertas partes del código no modificables (solo dentro del IDE) por lo cual para cambiar ciertos aspectos de la programación es necesario cambiar propiedades asociadas a los objetos que se van creando al momento de ubicarlos en la ventana. Cabe recalcar que el IDE automáticamente importa las librerías necesarias para todo el trabajo.

El método con que Netbeans inicializa los componentes de nuestro GUI es initComponents(), en donde al momento de inicializar los componentes asigna los valores por defecto. A medida que se vayan modificando estos valores en la ventana de diseño el código automáticamente se modifica. Para nuestro GUI, el código generado en este método consta de alrededor de 300 líneas.

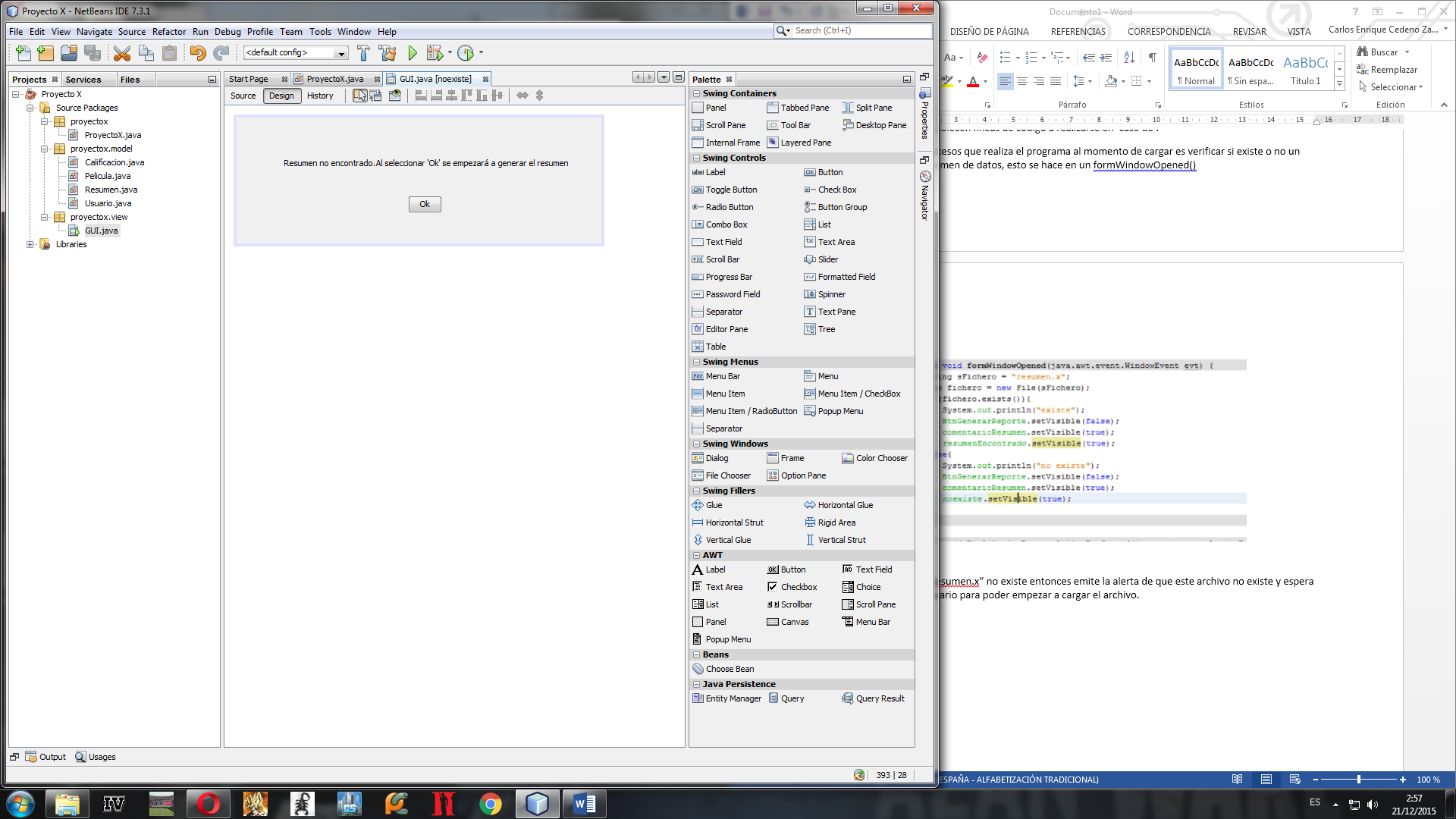


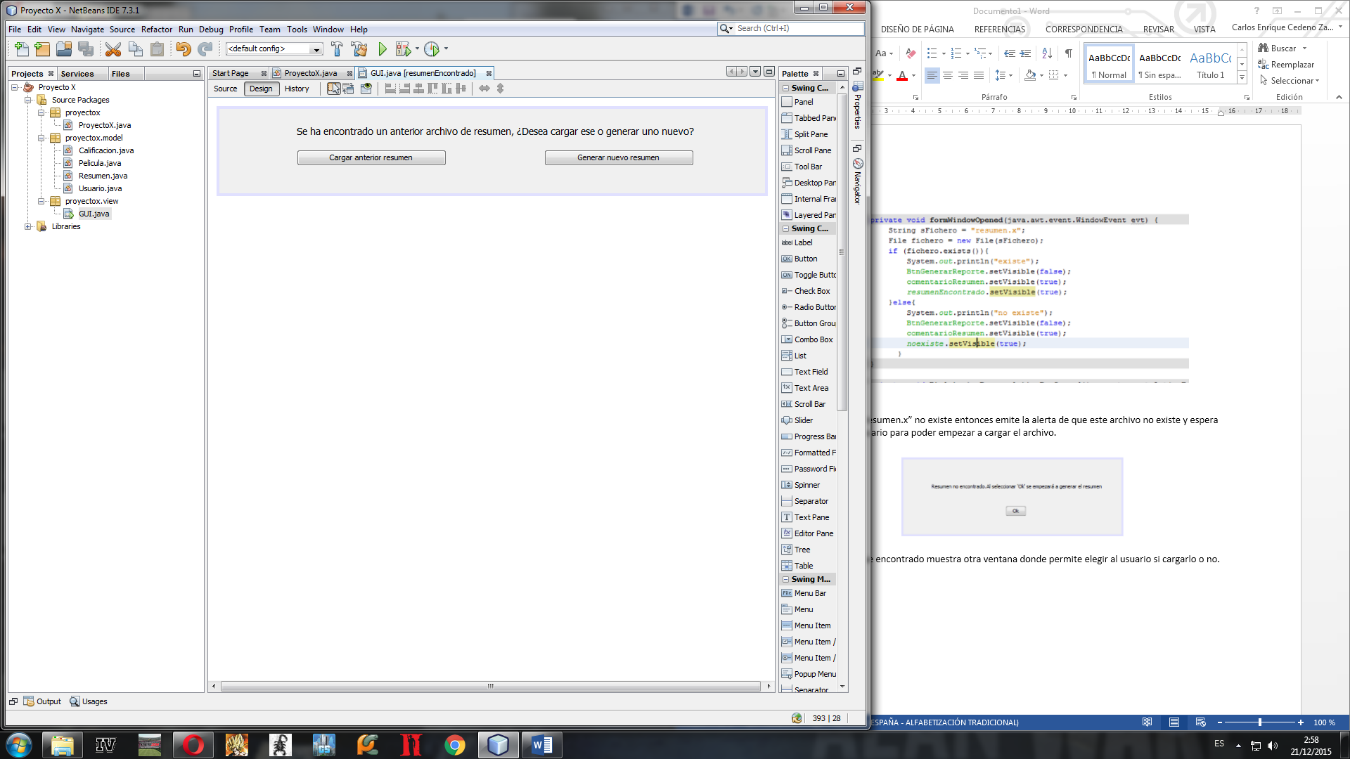
Dentro de los métodos actionPerformed() correspondientes a cada componente de nuestro entorno se establecen líneas de código a realizarse en ‘caso de’.

Uno de los procesos que realiza el programa al momento de cargar es verificar si existe o no un archivo de resumen de datos, esto se hace en un formWindowOpened()



Si el archivo “resumen.x” no existe entonces emite la alerta de que este archivo no existe y espera la señal del usuario para poder empezar a cargar el archivo.



Si el archivo fue encontrado muestra otra ventana donde permite elegir al usuario si cargarlo o no.

# **Explicar errores que se tuvieron durante la implementación del codio mediante imágenes de pantalla capturadas que muestren operaciones de depuración (bug) en los cuales se visualicen las variables y sus valores. Explique cuál era el error y cuál fue su solución.**

**ERRORES GENERALES (CÓDIGO JAVA).**

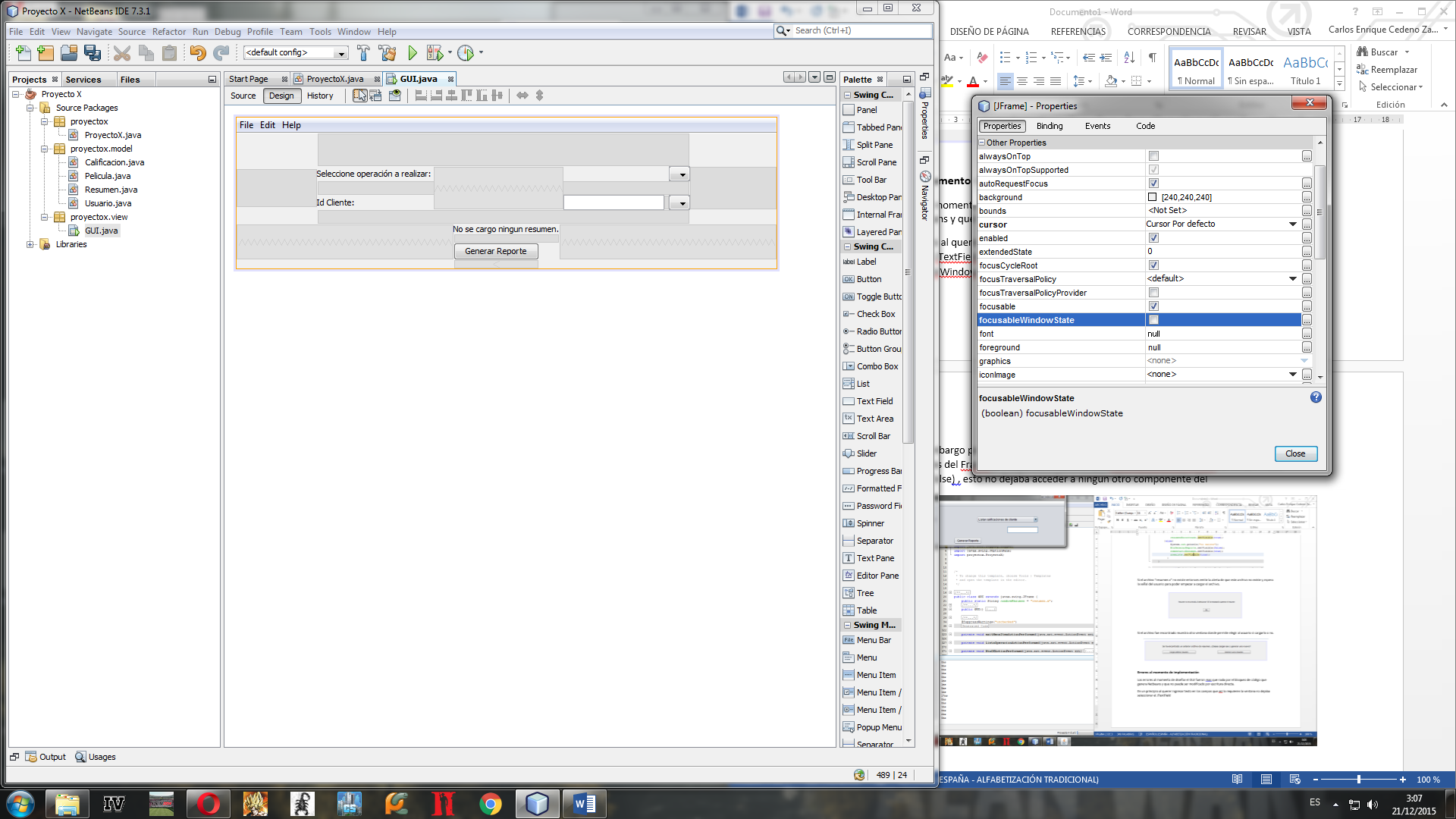
Uno de los problemas más frecuentes para mí, Bryan Ordóñez, por lo cual no pongo siquiera impresiones de pantalla fueron los errores de sintaxis. Debido a que en el semestre anterior usé Python, mi lenguaje de programación está grandemente influido, y cuando programaba con apuro, no podía evitar poner varias líneas como si este fuera Python. Debido a esto, mis problemas fueron:

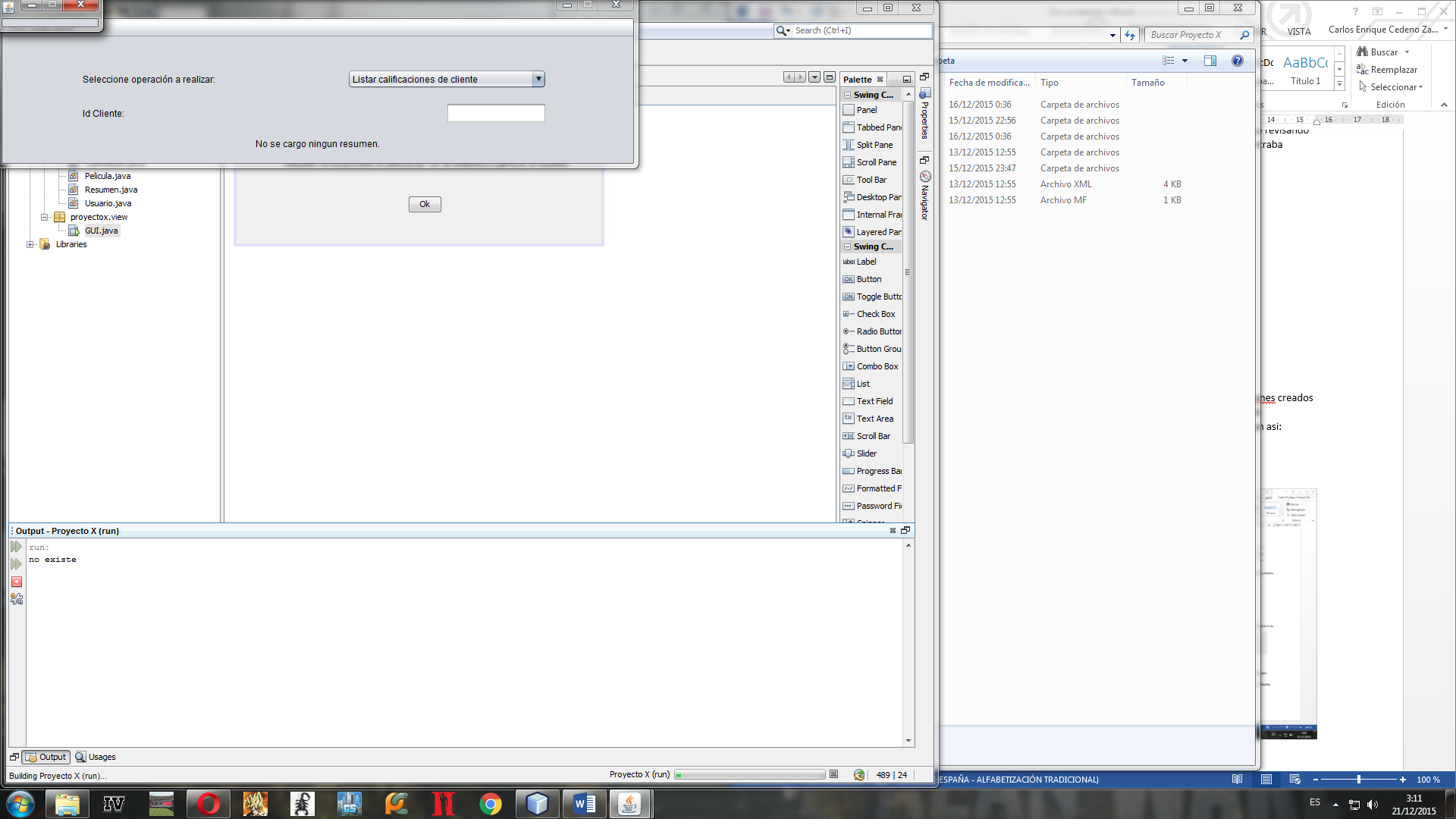
* Errores de sintaxis: Es el error más común al mesclar lenguajes de programación, por ejemplo los bucle for que usaba, tenían expresiones de Python.
* Errores de Casting

**ERRORES AL MOMENTO DE DISEÑAR EN GUI**

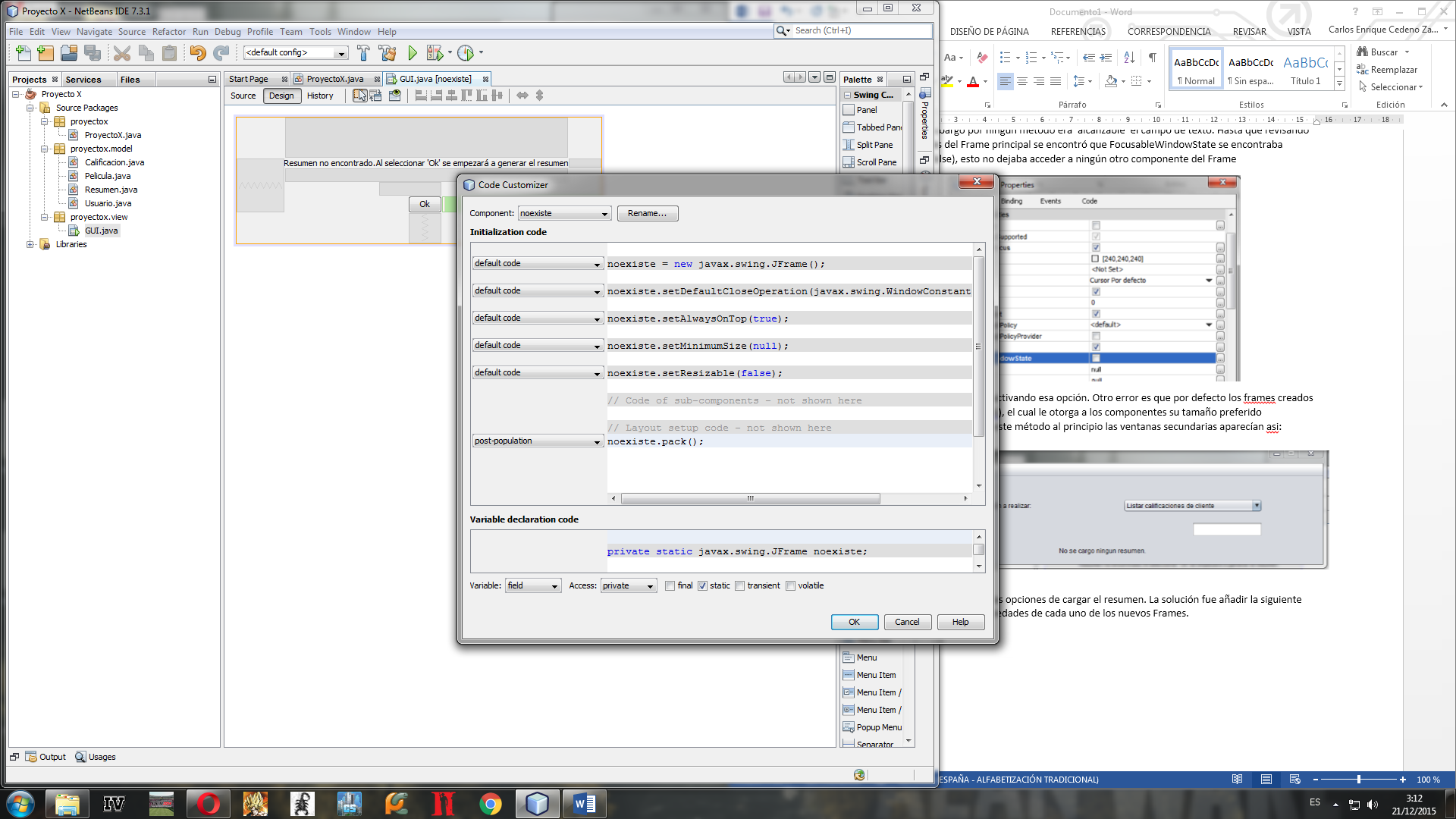
Los errores al momento de diseñar el GUI fueron más que nada por el bloqueo de código que genera Netbeans y que no puede ser modificado por escritura directa.

En un principio al querer ingresar texto en los campos que asi lo requieren la ventana no dejaba seleccionar el JTextField por ningún método. Eran aplicados focusRequest() y requestFocusInWindow() los cuales no necesariamente pueden ir en el código, solo facilitan el uso del GUI, sin embargo por ningún método era ‘alcanzable’ el campo de texto. Hasta que revisando las propiedades del Frame principal se encontró que FocusableWindowState se encontraba desactivado (false), esto no dejaba acceder a ningún otro componente del Frame

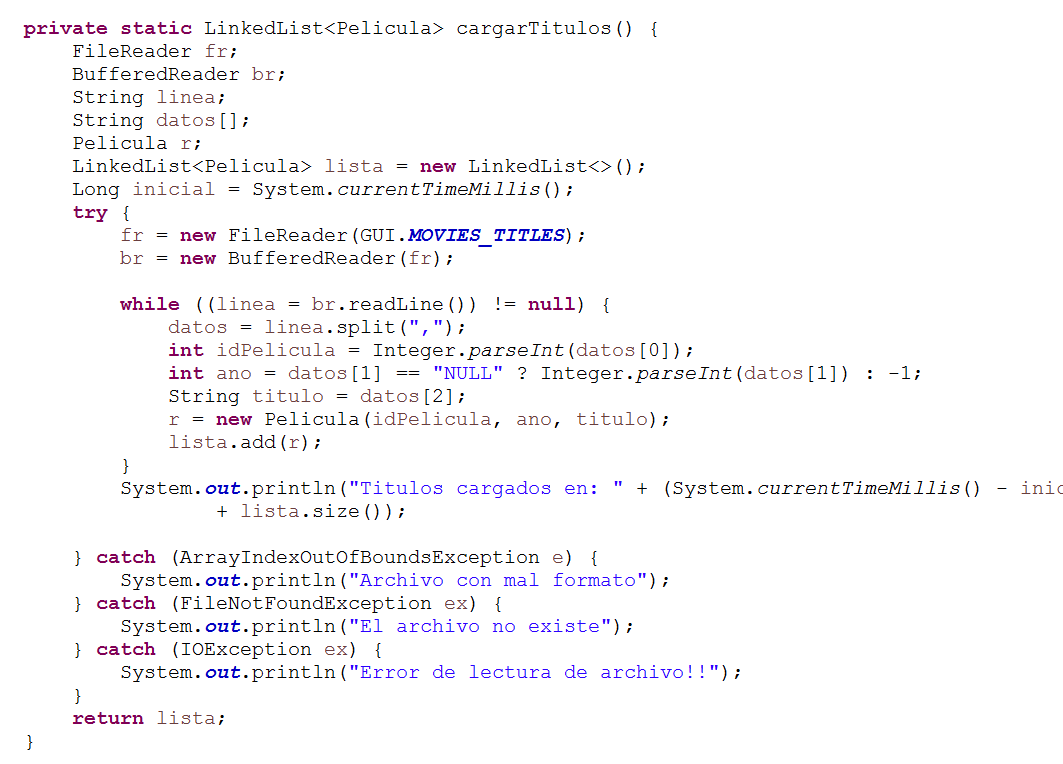


Esto pudo ser solucionado activando esa opción. Otro error es que por defecto los frames creados no incluyen el método pack(), el cual le otorga a los componentes su tamaño preferido establecido. Por la falta de este método al principio las ventanas secundarias aparecían asi:

Lo cual hacia inaccesible a las opciones de cargar el resumen. La solución fue añadir la siguiente línea en la ventana de propiedades de cada uno de los nuevos Frames.



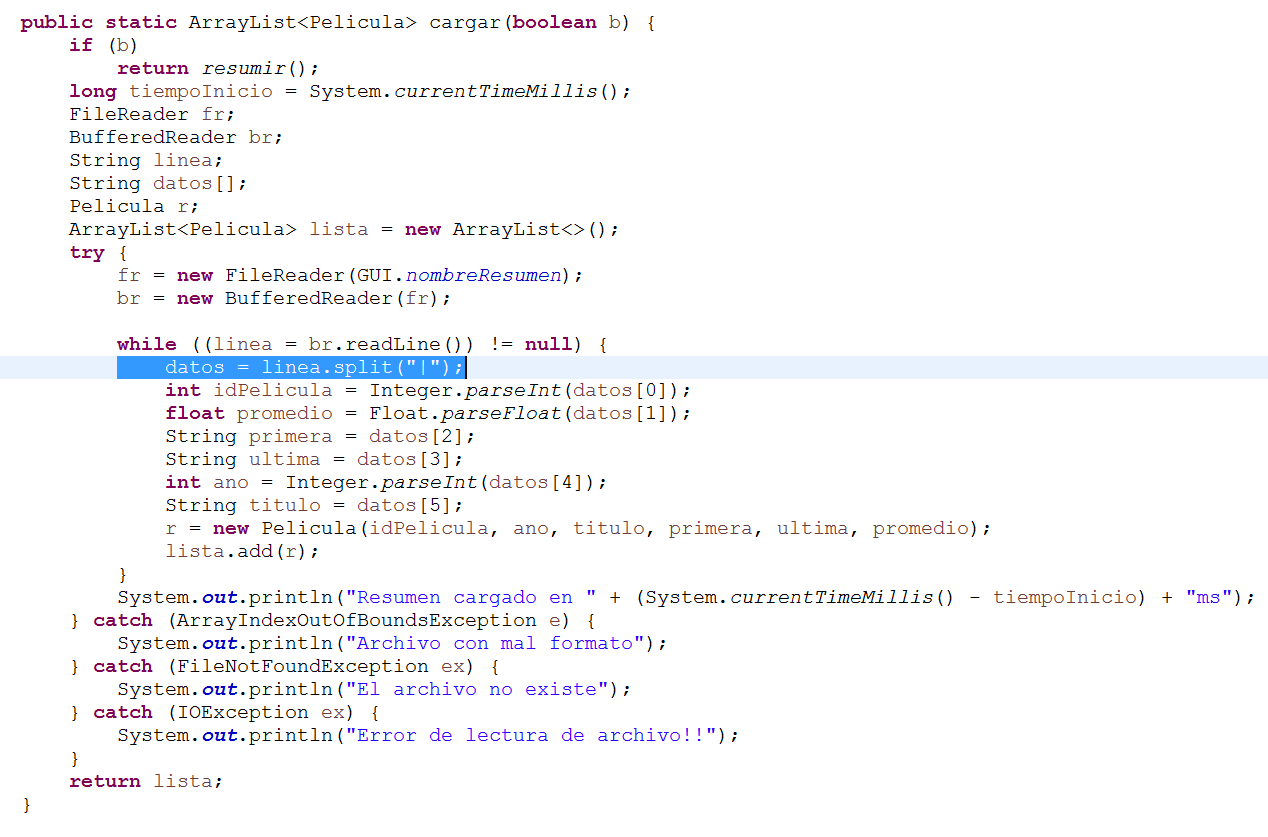
**OTROS ERRORES**



**Problema:** Algunas películas no contenían el año de lanzamiento.

**Error:** Integer.parseInt() no puede procesar “NULL”.

**Solución:** agregar condicional para “NULL” e inicializar con -1;



**Problema:** datos resumidos no cargaban desde el resumen.

**Error:** Carácter “|” no escapado correctamente.

**Solución:** cambiar el parámetro de linea.split(“|”) a línea.split(“\\|”).



**Problema:** Eficiencia para encontrar la película asociada al archivo de historial.

**Solución:** Búsqueda binaria. Con la búsqueda binaria se puede encontrar la posición del objeto dentro de la colección. Para luego REMOVERLO de la lista de títulos por resumir, disminuyendo así el tiempo de carga con cada vez que se ejecuta. Nota: ArrayList y LinkedList poseen O(n) para la operación remove(index); por lo que usar la ED previamente escogida para los títulos a resumir no afecta.

# **Describir el uso de su aplicación, es decir ejecútelo como usuario del mismo y explique cada una de las operaciones siguiendo un flujo de ejecución.**

Al ejecutar la aplicación hay dos rumbos que toma:

1. Si no existe un archivo “resumen.x” muestra el mensaje que no existe el resumen y espera la aprobación del usuario para continuar y generar ell resumen
2. Si existe un resumen previo pregunta al usuario si desea usar ese nuevo resumen encontrado o si desea generar uno nuevo.

Una vez cargado el archivo de resumen el usuario puede seleccionar en una lista la operación que desea realizar.

1. Si el usuario desea generar el top 10 de películas por categoría (calificación) se le permite elegir que categoría desea evaluar.
2. Si el usuario desea listar las películas por un año especifico se le permite ingresar un año (Ej. 2002).
3. Si el usuario quiere realizar una búsqueda por medio de un texto se le permite ingresar la frase o palabra que desea.

Estos 3 métodos reciben argumentos, los 3 siguientes no necesitan ningún tipo de argumento:

1. Mostrar películas ordenadas alfabéticamente descendente
2. Mostrar películas ordenadas descendentemente por año
3. Mostrar películas ordenadas por categoría.

# **Comprobación del rendimiento de la aplicación cuando carga los datos en listas usando ArrayList y LinkedList**

Al momento de cargar usamos LinkedList para generar los títulos de películas porque la cantidad de películas es desconocida. Luego de esto lo guardamos en un ArrayList para evitar tener doble referencia de todos los títulos. Mantuvimos los datos en un ArrayList de capacidad inicial igual al tamaño de los títulos para el fácil acceso aleatorio.

En pruebas iniciales efectuadas con ArrayList el resumen se demoraba un poco más (un 140% más) que con LinkedList.