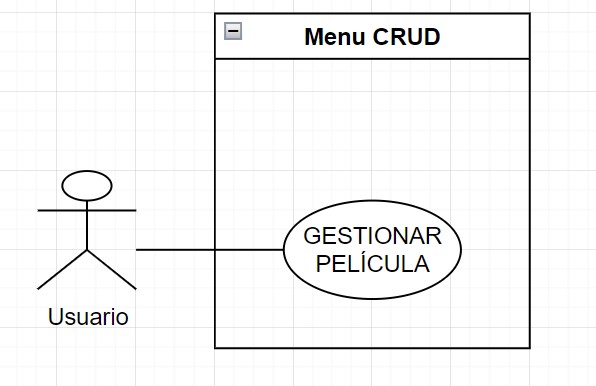
**Desarrollo de programa CRUD con Java**

Alfonso López Delgado

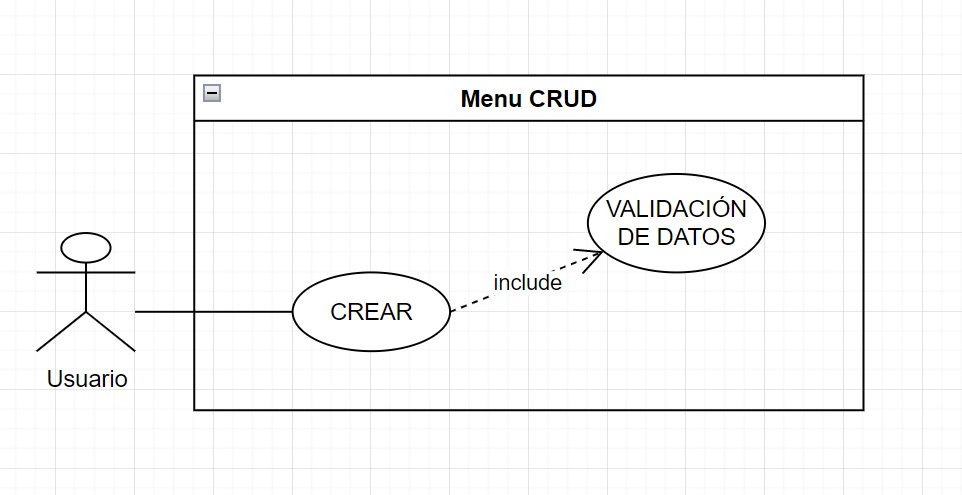
1ºDAM

**Análisis de casos de uso CRUD tema películas**

1. Gestionar película

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | NOMBRE | DESCRIPCIÓN | |
| |  |  | | --- | --- | | Gestionar película | El usuario puede elegir entre 5 opciones para tratar los datos, crear, modificar, borrar, consultar y salir del menú | |
| |  |  | | --- | --- | | PRECONDICIONES | POSTCONDICIONES | |
| |  |  | | --- | --- | | * Iniciar programa * Mostrar menú * Introducción por teclado | * Dirección opciones | |
| |  |  |  | | --- | --- | --- | | DATOS | DATOS ENTRADA | DATOS SALIDA | |
| |  |  |  | | --- | --- | --- | | * Mostrar menú | * Mostrar menú * Dato | * Mostrar opción | |

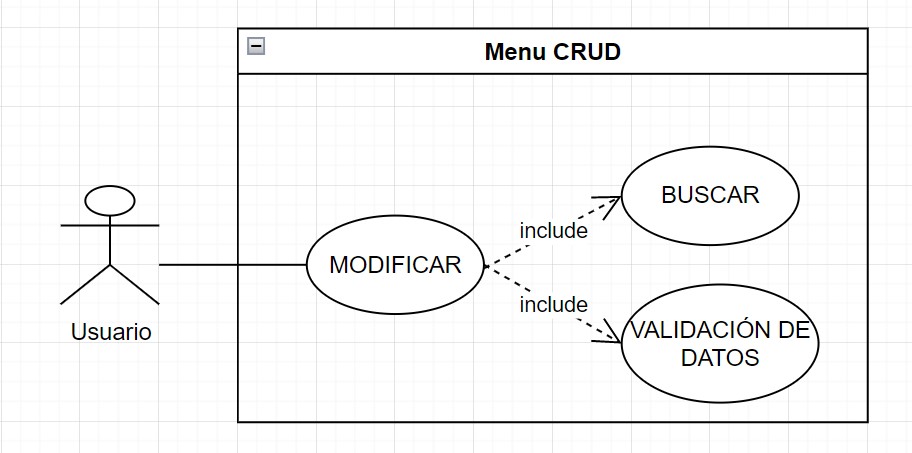
1. Crear



|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | NOMBRE | DESCRIPCIÓN | |
| |  |  | | --- | --- | | Crear película | El usuario crea una película con los datos que se le pide | |
| |  |  | | --- | --- | | PRECONDICIONES | POSTCONDICIONES | |
| |  |  | | --- | --- | | * Array para almacenar el nombre * Array para almacenar la fecha * Array para almacenar la valoración * Introducción por teclado de película | * Película nueva previa validación de datos. | |
| |  |  |  | | --- | --- | --- | | DATOS | DATOS ENTRADA | DATOS SALIDA | |
| |  |  |  | | --- | --- | --- | | * Array nombre * Array fecha * Array valoración | * Mensaje | * Dato (nombre, fecha, valoración) | |

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | NOMBRE | DESCRIPCIÓN | |
| |  |  | | --- | --- | | Validación de datos | Validación de datos introducidos por el usuario | |
| |  |  | | --- | --- | | PRECONDICIONES | POSTCONDICIONES | |
| |  |  | | --- | --- | | * Dato introducido por teclado * Dato dentro de los parámetros | * Dato insertado en los arrays dentro de los parámetros establecidos | |
| |  |  |  | | --- | --- | --- | | DATOS | DATOS ENTRADA | DATOS SALIDA | |
| |  |  |  | | --- | --- | --- | | * Array nombre * Dato | * Mensaje * Dato sin validar | * Dato (nombre, fecha, valoración) validado | |

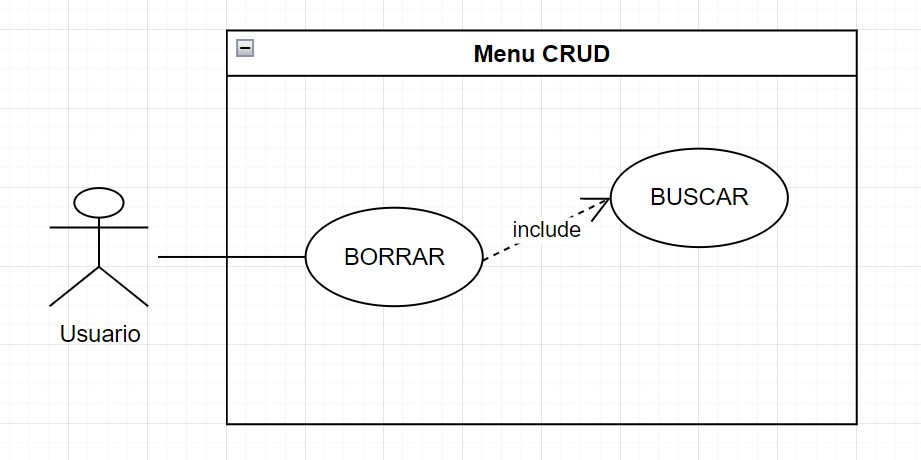
1. Modificar

****

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | NOMBRE | DESCRIPCIÓN | |
| |  |  | | --- | --- | | Modificar | Modificación de algún dato, que haya introducido el usuario previamente. | |
| |  |  | | --- | --- | | PRECONDICIONES | POSTCONDICIONES | |
| |  |  | | --- | --- | | * Dato almacenado en Array * Introducción del nuevo dato | * Eliminación del antiguo dato y sustitución por el nuevo | |
| |  |  |  | | --- | --- | --- | | DATOS | DATOS ENTRADA | DATOS SALIDA | |
| |  |  |  | | --- | --- | --- | | * Array nombre * Dato | * Mensaje | * Dato (nombre, fecha, valoración) nuevos | |

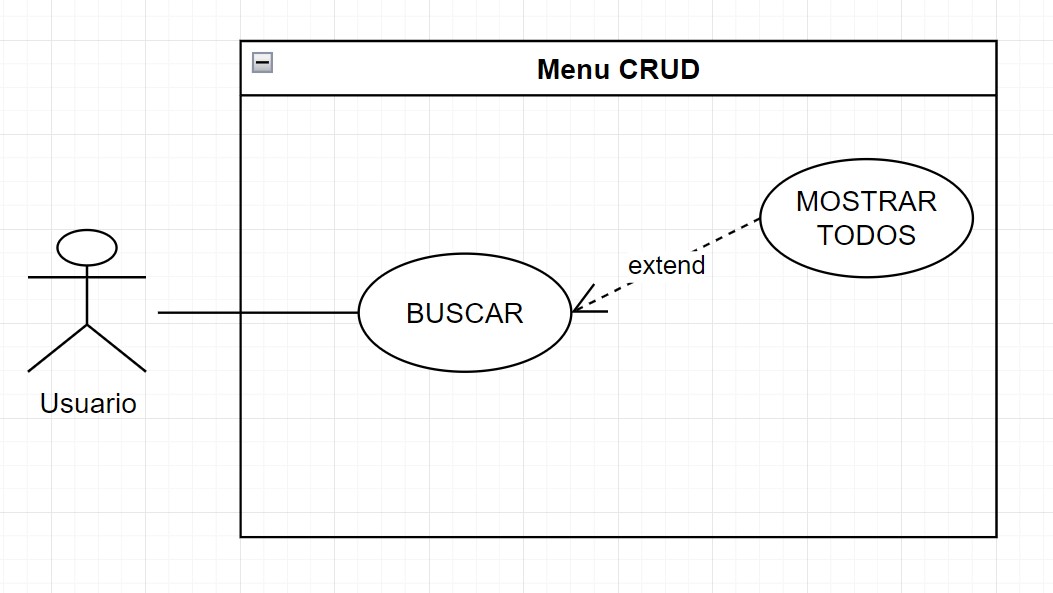
|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | NOMBRE | DESCRIPCIÓN | |
| |  |  | | --- | --- | | Buscar | Buscar el dato introducido por el usuario | |
| |  |  | | --- | --- | | PRECONDICIONES | POSTCONDICIONES | |
| |  |  | | --- | --- | | * Dato introducido por teclado * Dato almacenado en Array | * Dato encontrado | |
| |  |  |  | | --- | --- | --- | | DATOS | DATOS ENTRADA | DATOS SALIDA | |
| |  |  |  | | --- | --- | --- | | * Array nombre * Dato posición | * Dato teclado | * Dato posición | |

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | NOMBRE | DESCRIPCIÓN | |
| |  |  | | --- | --- | | Validación de datos | Validación de datos introducidos por el usuario | |
| |  |  | | --- | --- | | PRECONDICIONES | POSTCONDICIONES | |
| |  |  | | --- | --- | | * Dato introducido por teclado * Dato dentro de los parámetros | * Dato insertado en los arrays dentro de los parámetros establecidos | |
| |  |  |  | | --- | --- | --- | | DATOS | DATOS ENTRADA | DATOS SALIDA | |
| |  |  |  | | --- | --- | --- | | * Array nombre * Array fecha * Array valoración * Dato | * Mensaje * Dato sin validar | * Dato (nombre, fecha, valoración) validado | |

1. Borrar

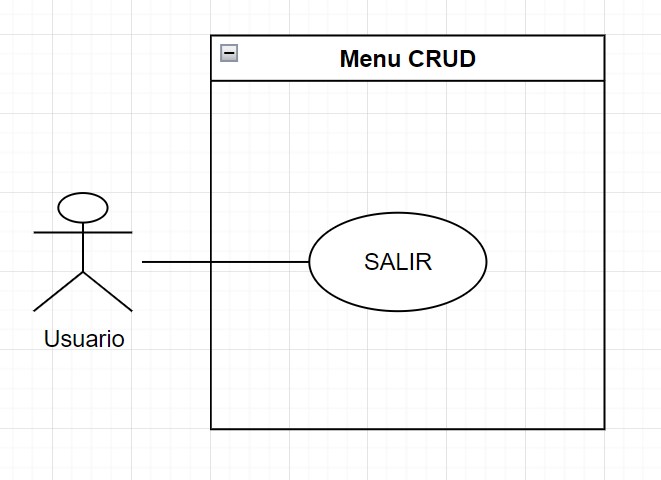
|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | NOMBRE | DESCRIPCIÓN | |
| |  |  | | --- | --- | | Borrar | Borrado de un dato en el array mediante la sustitución de otro dato neutral. | |
| |  |  | | --- | --- | | PRECONDICIONES | POSTCONDICIONES | |
| |  |  | | --- | --- | | * Dato almacenado en Array | * Sustitución por elemento neutral (como un espacio) | |
| |  |  |  | | --- | --- | --- | | DATOS | DATOS ENTRADA | DATOS SALIDA | |
| |  |  |  | | --- | --- | --- | | * Array nombre | * Mensaje | * Dato (“”, “”, “”) | |

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | NOMBRE | DESCRIPCIÓN | |
| |  |  | | --- | --- | | Buscar | Buscar el dato introducido por el usuario | |
| |  |  | | --- | --- | | PRECONDICIONES | POSTCONDICIONES | |
| |  |  | | --- | --- | | * Dato introducido por teclado * Dato almacenado en Array | * Dato encontrado | |
| |  |  |  | | --- | --- | --- | | DATOS | DATOS ENTRADA | DATOS SALIDA | |
| |  |  |  | | --- | --- | --- | | * Array nombre * Dato posición | * Dato teclado | * Dato posición | |

1. **Buscar**

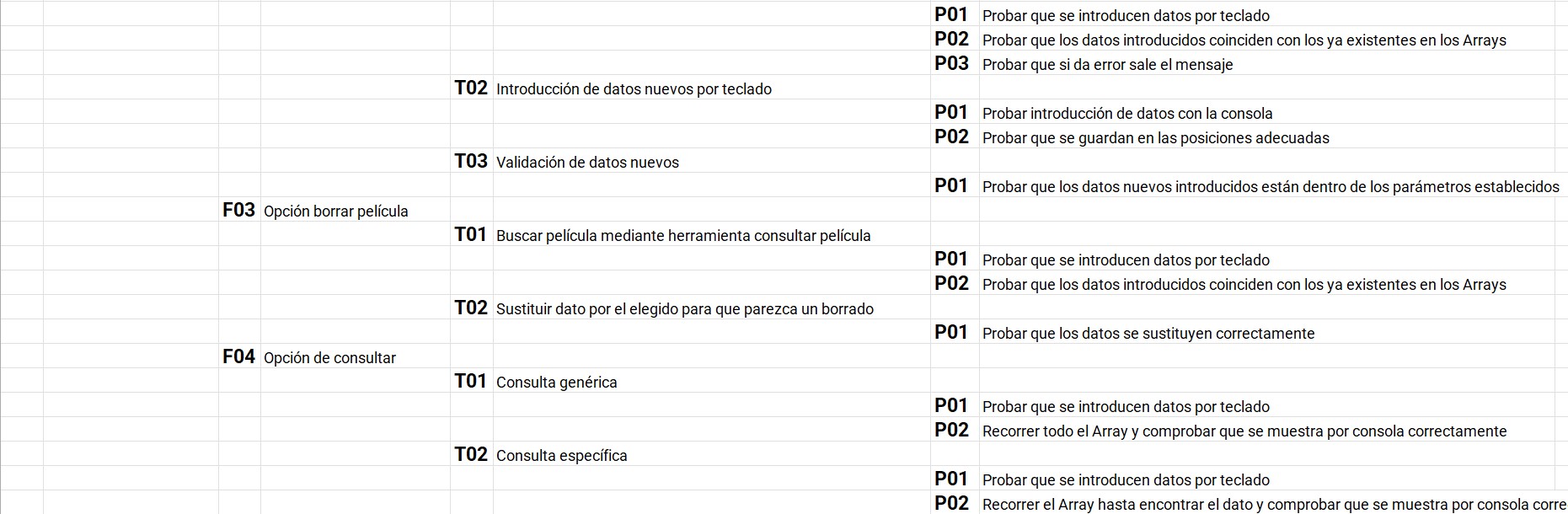
|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | NOMBRE | DESCRIPCIÓN | |
| |  |  | | --- | --- | | Buscar | Buscar el dato introducido por el usuario | |
| |  |  | | --- | --- | | PRECONDICIONES | POSTCONDICIONES | |
| |  |  | | --- | --- | | * Dato introducido por teclado * Dato almacenado en Array | * Dato encontrado * Mensaje | |
| |  |  |  | | --- | --- | --- | | DATOS | DATOS ENTRADA | DATOS SALIDA | |
| |  |  |  | | --- | --- | --- | | * Array nombre * Dato posición | * Dato teclado | * Mensaje | |

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | NOMBRE | DESCRIPCIÓN | |
| |  |  | | --- | --- | | Mostrar todos | El usuario puede elegir si hacer una búsqueda específica o mostrar todos los datos del array | |
| |  |  | | --- | --- | | PRECONDICIONES | POSTCONDICIONES | |
| |  |  | | --- | --- | | * Elegir opción mostrar todo | * Mostrar todos los datos del array | |
| |  |  |  | | --- | --- | --- | | DATOS | DATOS ENTRADA | DATOS SALIDA | |
| |  |  |  | | --- | --- | --- | | * Array nombre |  | * Mensaje | |

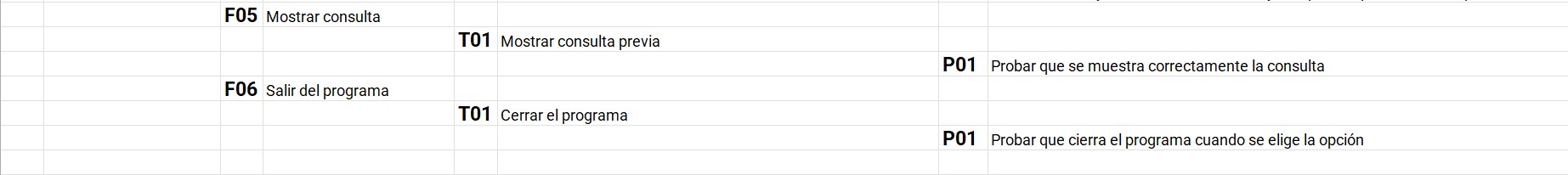
1. Salir

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | NOMBRE | DESCRIPCIÓN | |
| |  |  | | --- | --- | | Salir | Opción salir del menú | |
| |  |  | | --- | --- | | PRECONDICIONES | POSTCONDICIONES | |
| |  |  | | --- | --- | | * Elegir opción | * Mensaje * Apagar consola | |
| |  |  |  | | --- | --- | --- | | DATOS | DATOS ENTRADA | DATOS SALIDA | |
| |  |  |  | | --- | --- | --- | |  |  | * Mensaje | |

**RFTP**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

****

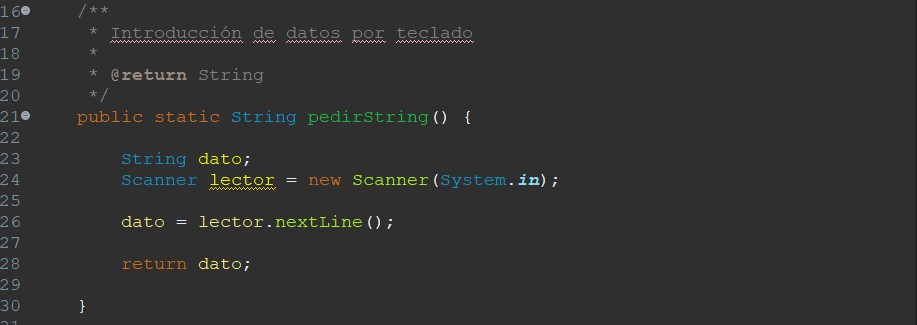
**Código comentado**

Este apartado será una breve explicación del código del programa que se ha realizado.

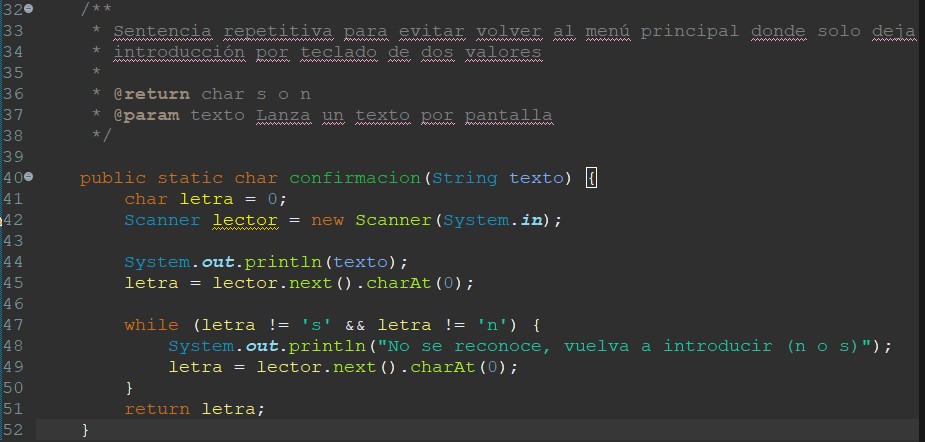
**Clase utilidades**

Es creado con la intención de la reutilización del código que se repetirá en reiteradas ocasiones durante el proyecto.

* **Método pedirString()** 🡪 Creado con la finalidad de introducir por teclado los valores String que necesite el usuario.

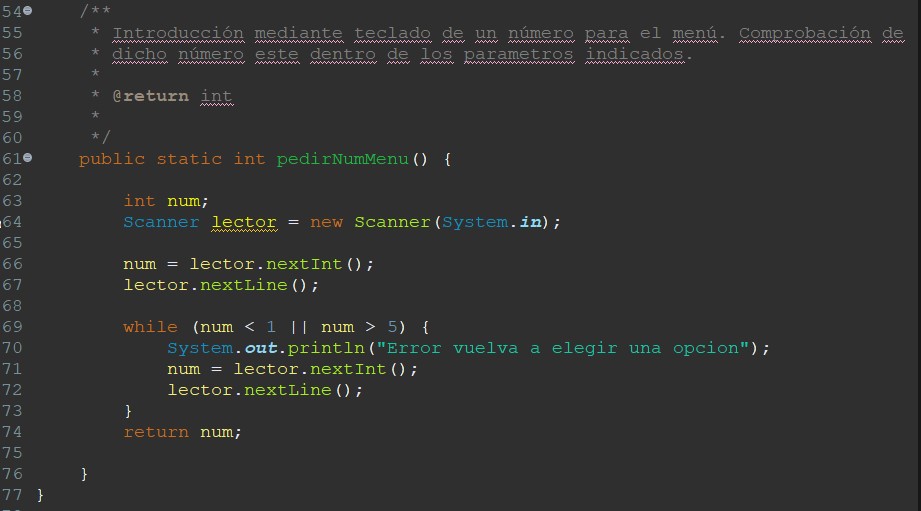
***Explicación del código:*** Creación de método donde retornara un dato String pedida en la variable “dato” mediante la clase Scanner.

* **Método confirmación()** 🡪 Creado con la finalidad de seguir en la misma opción del menú si el usuario lo requiere y evitar salir y volver a entrar.

***Explicación del código:*** Creación de un método donde retornara un dato char. Se pedirá una letra con la variable tipo char “letra” con la clase Scanner. El dato introducido entrará en un bucle while donde se repetirá mientras la letra sea distinto de ‘s’ o ‘n’, unidos por un operador lógico AND, y lanzará un mensaje de “error”.

* **Método pedirNumMenu()** 🡪 Creado con la finalidad de pedir una opción al usuario para elegir en el menú.

***Explicación de código***: Creación de método en el cual retornara un dato int.

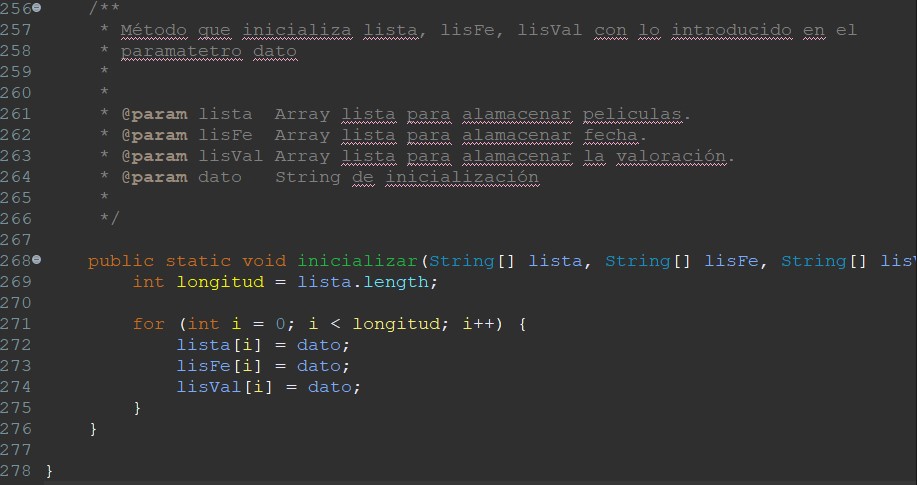
Se pedirá un número recogido con la variable “num”, de tipo int, por teclado mediante la clase Scanner. El dato entra en un bucle while el cual se repetirá mientras dichos parámetros se cumplan, unidos por un operador lógico OR, y lanzará un mensaje por pantalla de “error”.

**Clase menuFinal**

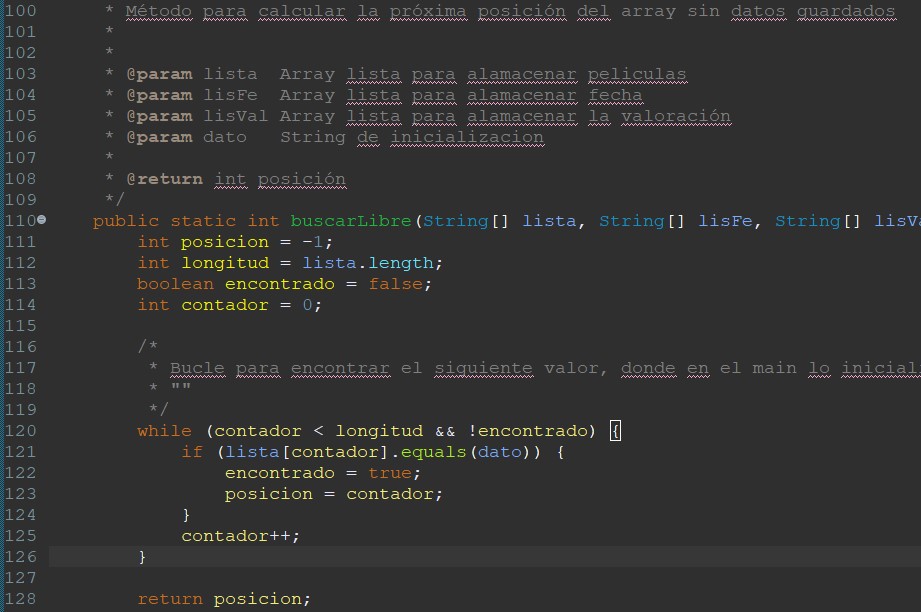
* **Método menú()** 🡪 La estética del menú que se imprimirá por consola.

***Explicación del código:*** Mediante el conjunto de instrucciones System.out.println(“”); se le dará una forma gráfica.

* **Método inicializar(String[] lista, String[] lisFe, String[] lisVal, String dato)** 🡪 Su finalidad es rellenar todas las posiciones de los tres arrays lista, lisFe, lisVal con un dato.

***Explicación del código:*** Creación de un método donde no devolverá nada y le dirá al programa que rellene los parámetros declarados. Mediante la variable “longitud” y la función “length” recogeremos la longitud del array y con un bucle for recorreremos todo el array introduciendo en cada posición del array el parámetro “dato”.

* **Método buscarLibre(String[] lista, String[] lisFe, String[] lisVal, String dato) 🡪** Su finalidad es buscar la primera posición del array donde haya un hueco libre.

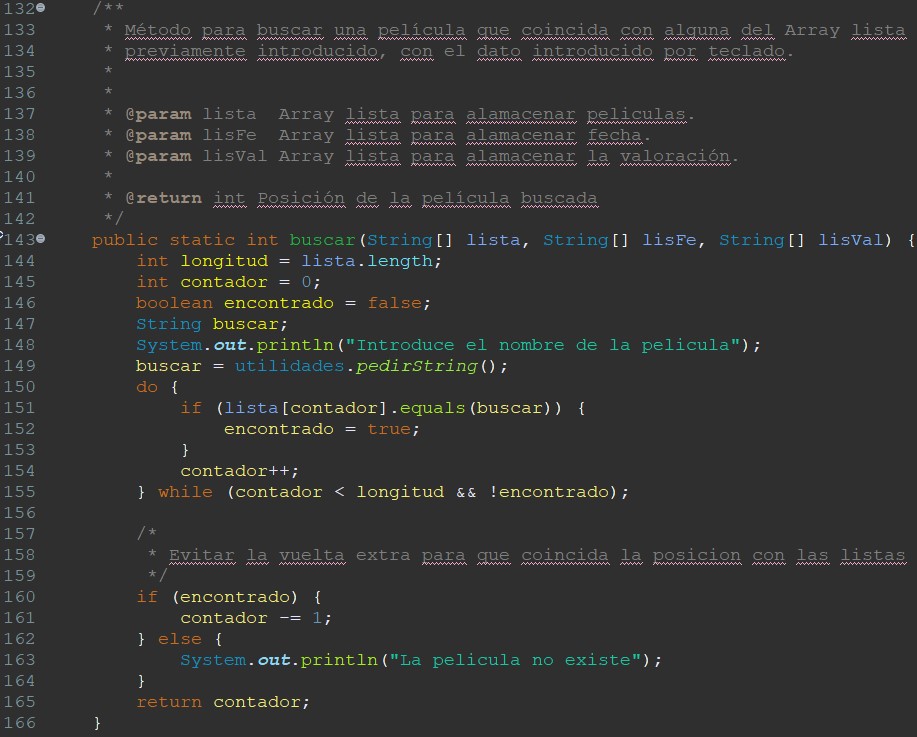
***Explicación del código:*** Creación de un método donde retornará una Int el cual será la posición donde se localiza el primer hueco libre. Mediante un bucle while con la condición de que se ejecute mientras la variable contador sea menor que la longitud del array AND la variable booleana distinta de la declarada, y anidando una sentencia if comparando cada posición del array con el “dato”. Cuando la variable booleana es “true” sale del while con la variable “posición” igual al número de vueltas que ha dado el contador y retornará la posición donde se ha encontrado el primer “dato”.

* **Método buscar(String[] lista, String[] lisFe, String[] lisVal) 🡪** Su finalidad es buscar la posición del dato introducido por el usuario.

***Explicación del código:*** Mediante los parámetros lista, lisFe, lisVal.

* + 1. Solicitamos la clase utilidades.pedirString(); creada previamente, se pedirá un dato tipo String que se guardará en la variable “buscar”.
    2. Utilizaremos un bucle do while con la condición, mientras la variable “contador” sea menor a la longitud del array AND la variable “encontrado” distinta a la declarada.

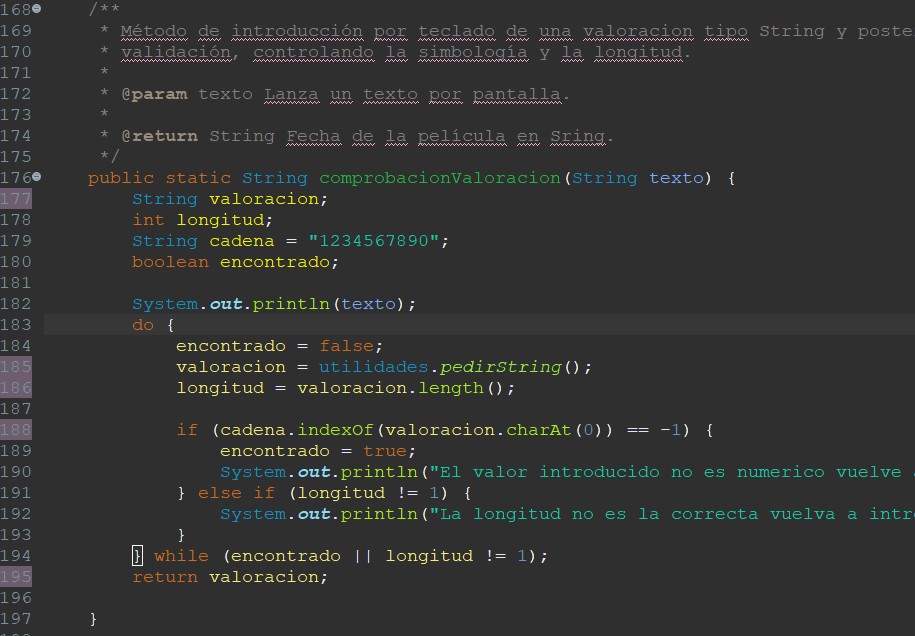
Anidando una sentencia if dentro del bucle con la condición de comparar cada posición de la lista del array con la que ha introducido el usuario en la variable “buscar”

* + 1. Ponemos una variable contador con incremento para que recoja el valor de cada vuelta del bucle.
    2. La siguiente sentencia if(encontrado) se encarga de dos acciones.
       1. Si en la sentencia anterior ha encontrado la igualdad, la variable “encontrado” es igual a “true” por lo que se cumple la sentencia y resta una vuelta al contador ya que dará el máximo de vueltas permitido (20) y nuestro array es de 19 posiciones por lo que daría un error.
       2. Si no ha encontrado el valor esperado significa que ha recorrido todo el array y no ha encontrado similitud por lo que la película no existe.
* **Método comprobacionValoracion(String texto)** 🡪 Su finalidad es comprobar que cuando el usuario introduzca una valoración se corresponda a la longitud pedida y con la simbología correcta.

***Explicación de código:*** El parámetro texto lanzará un mensaje por pantalla.

1. Utilizamos un bucle do while (ya que queremos que se ejecute al menos una vez la sentencia) con la condición booleana “encontrado” OR longitud distinto de 1.

Dentro inicializamos la variable “encontrado” igual a “false”, inicializamos la variable “valoración” pidiendo la clase utilidades.pedirString() y inicializamos la longitud con el tamaño del String introducido por el usuario.

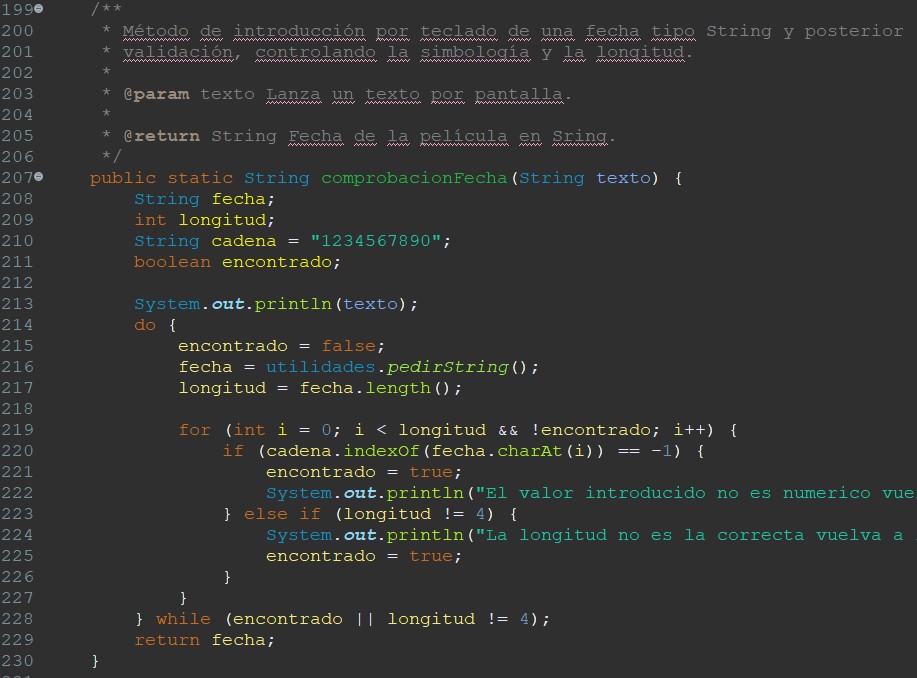
1. Utilizamos una sentencia if con la condición: mientras la instrucción indexOf de la variable “cadena” hacia la variable ”valoracion” en la primera posición sea igual a -1, saque un mensaje de texto de “error de simbología”. Ya que significa que en el dato que ha introducido el usuario no hay ninguno que se parezca a lo que contiene la variable “cadena” por lo que devuelve un -1.
2. Utilizamos un else if para insertar la condición de la longitud y manda un mensaje de error de longitud

* **Método comprobacionFecha(String texto)** 🡪 Su finalidad es la comprobación de que los datos introducidos por el usuario en el array fecha se cumplan con los parámetros indicados.

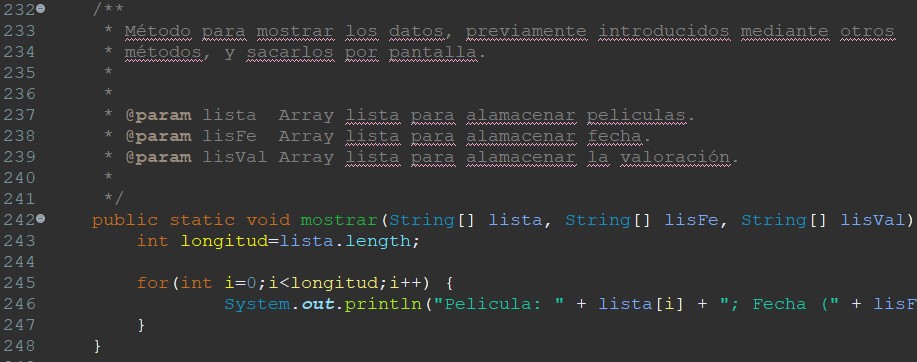
***Explicación de código:*** El parámetro texto lanzará un mensaje por pantalla.

1. Utilizamos un bucle do while (ya que queremos que se ejecute al menos una vez la sentencia) con la condición booleana “encontrado” OR longitud distinto de 4.

Dentro inicializamos la variable “encontrado” igual a “false”, inicializamos la variable “fecha” pidiendo la clase utilidades.pedirString() y inicializamos la longitud con el tamaño del String introducido por el usuario.

1. En este caso utilizaremos un bucle for para poder recorrer las posiciones del String fecha y realizar su comprobación una a una.
2. Utilizamos una sentencia if con la condición: mientras la instrucción indexOf de la variable “cadena” hacia la variable ”fecha” en la primera posición sea igual a -1, saque un mensaje de texto de “error de simbología”. Ya que significa que en el dato que ha introducido el usuario no hay ninguno que se parezca a lo que contiene la variable “cadena” por lo que devuelve un -1.
3. Utilizamos un else if para insertar la condición de la longitud y manda un mensaje de “error” de longitud

* **Método mostrar(String[] lista, String[] lisFe, String[] lisVal) 🡪** Su finalidad es mostrar el contenido de los arrays

***Explicación del código:*** mediante un bucle for le indicamos al System.out.println(“”) que muestre por pantalla el mensaje concatenado que hay en su interior cogiendo los valores de los parámetros declarados.

* Método main 🡪 Su finalidad es ejecutar los métodos anteriormente mencionados haciendo una combinación de instrucciones donde se lanzará por pantalla un menú y el usuario debe elegir una de las opciones añadir, modificar, borrar, consultar, salir.

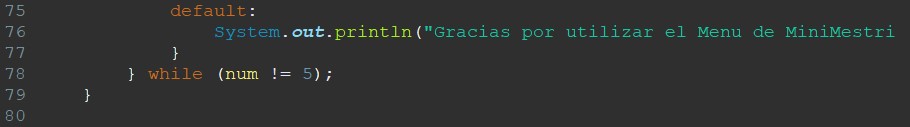
***Explicación de código:***

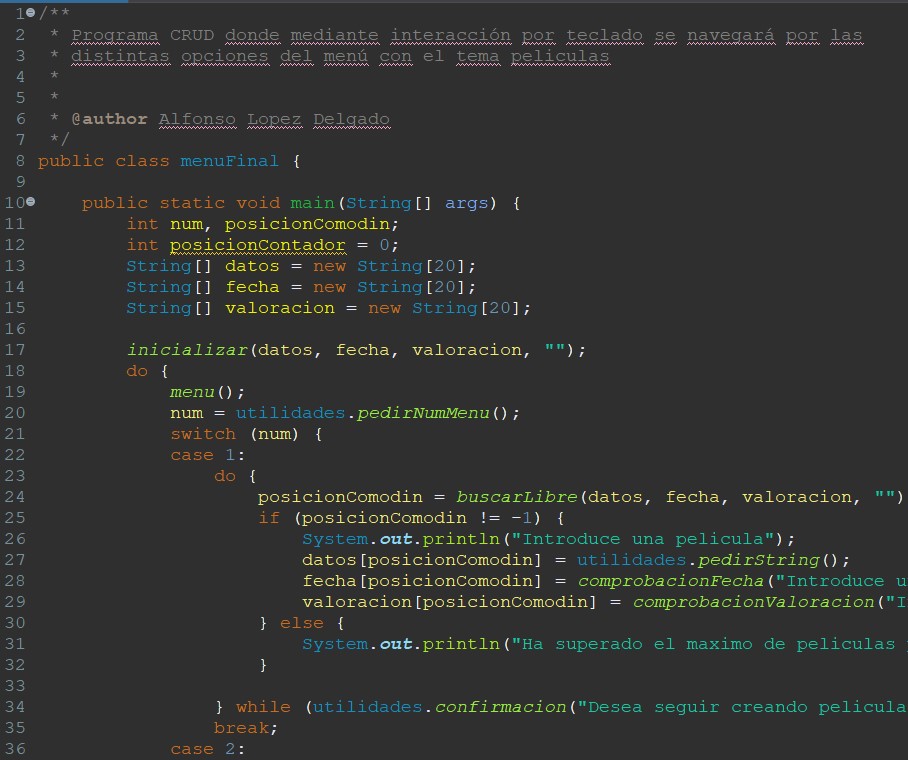
1. Utilización de método inicializar(datos,fecha,valoración, inicializándolos a “”); (Ahora las 20 posiciones de los arrays tienen el valor “”).
2. Utilización de un bucle do while donde su condición sea que se repita mientras que la variable “num” , que es igual a la clase utilidades.pedirNumMenu(); sea distinto de 5.
3. Utilización de sentencia switch para la elección de la opción previamente introducida.
   * + 1. Caso 1(Añadir): Utilizamos un bucle do while para realizar una sentencia repetitiva que nos evitará salir al menú principal y poder añadir películas sin interrupción.
          1. Se pide al método buscarLibre(); que busque la primera posición con el contenido “”.
          2. Con la sentencia if nos cercioramos de que haya hueco en el array ya que el programa devolverá un valor=-1 por lo que saldrá mensaje de “array lleno”.

Si hay hueco se ejecutarán las instrucción donde se guardarán , con la posición recogida anteriormente en el método buscarLibre, en los tres arrays mediante la utilización de los métodos, utilidades.pedirString(), comprobacionFecha(), comprobacionValoracion.

* + - 1. Caso 2 (Modificar): Utilizamos un bucle do while para realizar una sentencia repetitiva que nos evitará salir al menú principal y poder añadir películas sin interrupción.

1. Utilizaremos una variable “posicionComodin” para recoger el valor del método buscar();
2. Con la sentencia if me cercioro de que el valor recogido por buscar sea distinto del tamaño del array ya que en ese caso daría error de consola y significaría que no ha encontrado el valor que el usuario ha introducido.
3. Se ejecutarán las instrucción donde se guardarán los nuevos datos introducidos en los arrays con los métodos, utilidades.pedirString(), comprobacionFecha(), comprobacionValoracion.
   * + 1. Caso 3 (Borrar): ): Utilizamos un bucle do while para realizar una sentencia repetitiva que nos evitará salir al menú principal y poder añadir películas sin interrupción.
          1. Utilizaremos una variable “posicionComodin” para recoger el valor del método buscar();
          2. Con la sentencia if me cercioro de que el valor recogido por buscar sea distinto del tamaño del array ya que en ese caso daría error de consola y significaría que no ha encontrado el valor que el usuario ha introducido.
          3. En este caso se completarán las posiciones previamente buscadas con el valor (“”);
       2. Caso 4 (Consultar): Utilizamos un bucle do while para realizar una sentencia repetitiva que nos evitará salir al menú principal y poder añadir películas sin interrupción.
          1. Con la sentencia if else elegiremos una opción u otra
4. Anidando otra sentencia if me cercioro de que el valor recogido por buscar sea distinto del tamaño del array ya que en ese caso daría error de consola y significaría que no ha encontrado el valor que el usuario ha introducido.
5. Salida de mensaje concatenado con los parámetros.

Texto

Descripción generada automáticamente

**GIT**

El programa GitHub es bastante útil para trabajar en equipo, la detección de fallos sin tocar el programa principal o actuar como un manual histórico de los pasos que hemos hecho.

En primer lugar, he creado una rama principal donde guardaré todos mis archivos que contendrán el proyecto final y en paralelo he creado una rama nueva para trabajar sobre ella para evitar errores o fallos indeseados que puedan estropear el proyecto principal.

Mientras trabajaba con esta segunda rama he podido especificar cada cambio que hacia en el proyecto mediante comentarios o he descartado los antiguos.

Después de cada jornada actualizaba los cambios en la nube.

Una vez finalizado todas las partes del proyecto fusioné la rama secundaria con la principal para ver una única rama.