**Programa CRUD MySQL**

Alfonso López Delgado

1ºDAM

He comenzado el proyecto en GIT creando un repositorio para guardar el progreso que voy desarrollando y he utilizado el entorno de eclipse para escribir el código, pruebas Junit y respectiva documentación JavaDoc.

Para el diagrama de Gantt he utilizado la aplicación de CANVA en el cual apuntaba las horas que realizaba en cada momento del proyecto.

**EXPLICACIÓN DEL PROGRAMA**

He estructurado el programa en tres paquetes:

* ***Paquete utilidades*** 🡪 Que contendrá métodos de interacción con el usuario, validaciones y representaciones gráficas:
  + ***Validaciones*** 🡪 Esta clase se encargará de la interacción con el usuario, pidiendo textos y datos numéricos por consola y validaciones de los datos introducidos.
  + ***Utilidades*** 🡪 Contendrá la representación gráfica del menú, el menú de elección y la creación de los objetos.
* ***Paquete datos*** 🡪 Se encargará de realizar las conexiones con MySQL para el almacenamiento de los datos, contiene la clase objeto y los métodos para el tratamiento de los datos con la base de datos.
  + ***Pelicula*** 🡪 Es la clase objeto que contiene el constructor, getter, setter y toString.
  + ***Sql*** 🡪 En esta clase se realizará la mayoría de procedimientos que tengan que ver con la base de datos y el tratamiento de sus datos.
* ***Paquete control*** 🡪 Contiene la clase con la que se va a ejecutar el programa.

Para este programa he reutilizado algunos métodos que desarrollé en la anterior práctica 2 para ahorrar tiempo, además de utilizar la estructura de conexión SQL que proporcionaste en las tutorías.

En este proyecto de base de datos he dejado a un lado los métodos que retornan alguna posición ya que la programación conjunta con bases de datos resulta muy versátil gracias a las consultas de MySQL.

**EXPLICACIÓN DEL CÓDIGO**

**PAQUETE UTILIDADES**

**Utilidades 🡪** Es la clase donde confluyen la mayoría de los elementos creados en otras clases del programa.

* + - **Método caracteristicasPeli(String texto) 🡪** Método el cual generará un objeto de la clase Películas y se introducirán los valores correspondientes, reutilizando los métodos para introducir datos por teclado de la clase “validacion”.
    - **Método menú() 🡪** Se encarga de mostrar el menú gráfico que se lanzará por consola y de implementar la mecánica del menú para poder interactuar con las opciones creadas, mediante un switch.
      * **Case 1:** Llama al método “CrearPeli”, el cual recibirá un objeto creado, en la clase “caracteristicasPeli();” y lo añadirá a la base de datos.
      * **Case 2:** Para que el código sea más reutilizable y dinámico he creado un sentencia IF donde, dependiendo de la condición, reflejará una opción “Error” o llama al método ModificarPeli();.
      * **Case 3:** Llama al método “BorrarPeli”, el cual recibirá una cadena de texto y se encargará de eliminar el registro en la base de datos
      * **Case 4:** Llama al método “Buscar”, recibirá una cadena de texto
      * **Default:** Mostrará un texto de salida

Por último se controlaran las excepciones correspondientes.

**Validaciones 🡪** Esta clase se encarga de realizar las validaciones de los datos que el usuario introduzca por consola, con el objetivo de hacer el código reutilizable.

* + **Método pedirNum()🡪** Pedirá un número por teclado el cual deberá estar en un rango de 1-5, si no se repetirá hasta introducir la opción correcta. Por último, devolverá el número a la clase que lo solicite.
  + **Método pedirTexto() 🡪** Pedirá una cadena de texto por teclado, tendrá en cuenta si no se ha introducido ningún dato por teclado. Por último, devolverá la cadena a la clase que lo solicite.
  + **Método pedirVal() 🡪** Pedirá un número decimal por teclado, el cual tendrá un rango de 0-10 y tendrá en cuenta , mediante el control de excepciones, si el dato es numérico.
  + **Método pedirFecha() 🡪** Pedirá una fecha, que será una cadena de texto, por teclado la cual estará validada teniendo en cuenta que cumpla la longitud de 4 cifras y su integridad sea completamente numérica.

**PAQUETE DATOS**

**Películas 🡪** Es la clase encargada de generar el objeto, en este caso la información de la película. Contiene cuatro atributos (nombre, fecha, genero y valoracion), su constructor, sus getters y setters y el toString.

**sql 🡪** Esta clase se encarga de los principales métodos del CRUD, además de realizar las conexiones con la base de datos MySQL.

* + **CrearPeli(Película películas)** 🡪 Controla las excepciones correspondientes, inicia el driver para la conexión entre java y la bbdd. Recibirá un objeto por parámetro el cual, en la variable “sql” donde se realiza la sentencia (INSERT), se descompondrá el objeto en sus respectivos getters de cada atributo para que cuadré los datos introducidos por el usuario en la sentencia sql.
  + **ModificarPeli(String texto)** 🡪 He reutilizado y adaptado el código para esta práctica, tendrá el mismo menú que la práctica2. Recogerá una cadena de texto por parámetro que será el nombre de la película que el usuario ha introducido por consola para la inserción en la sentencia(UPDATE), previamente mediante el método “buscar()” se verifica que la película esté en la base de datos.

Una vez confirmada que la película existe, se podrá elegir entre cada una de las 5 opciones para modificar el registro:

**Case 1** 🡪 Modificará todo el registro elegido en el método “buscar()”, que retornará una cadena de texto que servirá de referencia en la sentencia para modificar el registro completo.

**Case 2** 🡪 Modificará únicamente el atributo “nombre” del objeto, recogiendo el parámetro que será una cadena de texto y modificando la columna “nombre” del registro.

**Case 3** 🡪 Modificará únicamente el atributo “fecha” del objeto, recogiendo el parámetro que será una cadena de texto y modificando la columna “fecha” del registro.

**Case 4** 🡪 Modificará únicamente el atributo “genero” del objeto, recogiendo el parámetro que será una cadena de texto y modificando la columna “genero” del registro.

Default 🡪 Modificará únicamente el atributo “valoracion” del objeto, recogiendo el parámetro que será una cadena de texto y modificando la columna “valoracion” del registro.

* **BorrarPeli(String texto)** 🡪 Controla las excepciones correspondientes, inicia el driver para la conexión entre java y la bbdd. Recibirá una cadena de texto por parámetro el cual le indicará a la sentencia (DELETE), que registro eliminar.
* **Buscar(String texto)** 🡪 Controla las excepciones correspondientes, inicia el driver para la conexión entre java y la bbdd. Declaramos un objeto resultset para lanzar la consulta y con el bucle while lo que hará es recorrer todo el conjunto de registros que ha recogido el resultset para encontrar la película buscada y sumará uno al contador, lo que nos indicará que existe la película, en caso que no exista el resultado será 0 .

La sentencia IF sirve para reutilizar el método buscar en el apartado de modificarpeli, ya que si no ha encontrado ningún registro el parámetro texto valdrá “nulo” y no entrará a modificar ningún registro y lanzará un mensaje de “no existe registro”, en caso contrario no entrará en la sentencia IF y retornará la cadena que ha introducido inicialmente el usuario por consola y se insertará en el parámetro del método “ModificarPeli()”. Es como una especie de cribado para que no haya que entrar en el método “ModificarPeli()” con dos parámetros.

* **Confirmación(int num)** 🡪 Es un método que lanzará un mensaje por pantalla de “procedimiento ok” o “no ok”, principalmente parar informar al usuario de si la película existe o no.

**PAQUETE CONTROLADOR**

Es el paquete que contiene la clase controlador, que se encargará de ejecutar el método menu()