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().

PRUEBAS JUNIT

- CrearPeli() → Declaramos los objetos peli1, peli2 y peli3. Insertamos solo peli1 y peli2, mediante el "assertEquals()" Nos aseguramos que los datos se introducen en sus respectivos atributos. Con la ayuda del método "buscar()" podremos realizar una consulta (SELECT *FROM películas where nombre="nombre") hacia la base de datos y que nos retorne un nombre o "nulo" si no ha encontrado nada.
    1. assertEquals → Comprobamos que el nombre del objeto "peli1" se ha insertado correctamente.
    2. assertEquals → Comprobamos que la fecha del objeto "peli2" se ha insertado correctamente.
    3. assertNotEquals → demuestra que la peli1 está en la BBDD, ya que el método buscar() nos debe devolver el nombre del objeto de peli1, que no es igual a "nulo"
    4. assertNotEquals → Misma función que el punto (3.) pero con el objeto peli2
    5. assertEquals → Comprueba que el objeto peli3 no está en la base de datos ya que no lo hemos insertado. En este caso el método buscar () nos retornará "nulo" ya que al lanzar la consulta no ha encontrado resultado.

id	**CREARPELI**
Descripción	Comprobar que el objeto se recibe e inserta correctamente en la base de datos
Condiciones previas	Creación de objeto e inserción de objeto
Gravedad	Crítica
Entorno	Eclipse
Pasos prueba	1.Crear objetos 2. Objeto introducidos
Resultado esperado	Peli1, peli2 están en la base de datos. Peli3 no está
Estado	Aprobado

- ModificarPeli() → He declarado el método “MetodoModificarPeli” para tener una instancia del método original donde solo se modifique el registro completo, ya que he puesto la opción de introducir por teclado alguna de las opciones a elegir y sería menos flexible al realizar las pruebas, además que las otras opciones son variantes de la consulta que ahora vamos a explicar. Dicho método contiene un objeto auxiliar “aux” el cual, en el programa original por teclado, en este caso lo prestablecemos.
  En este caso aprovecho la prueba anterior ya que creo los registros en la base de datos.
  1. assertNotEquals → Indica que el objeto “aux” se encuentra ya en la base de datos aportándonos un nombre y no “nulo”
  2. assertEquals → Indica que el objeto peli2 que anteriormente se ha introducido en la base de datos, ya no está en ella ya que lo hemos modificado por el objeto “aux”, por lo que da “nulo”

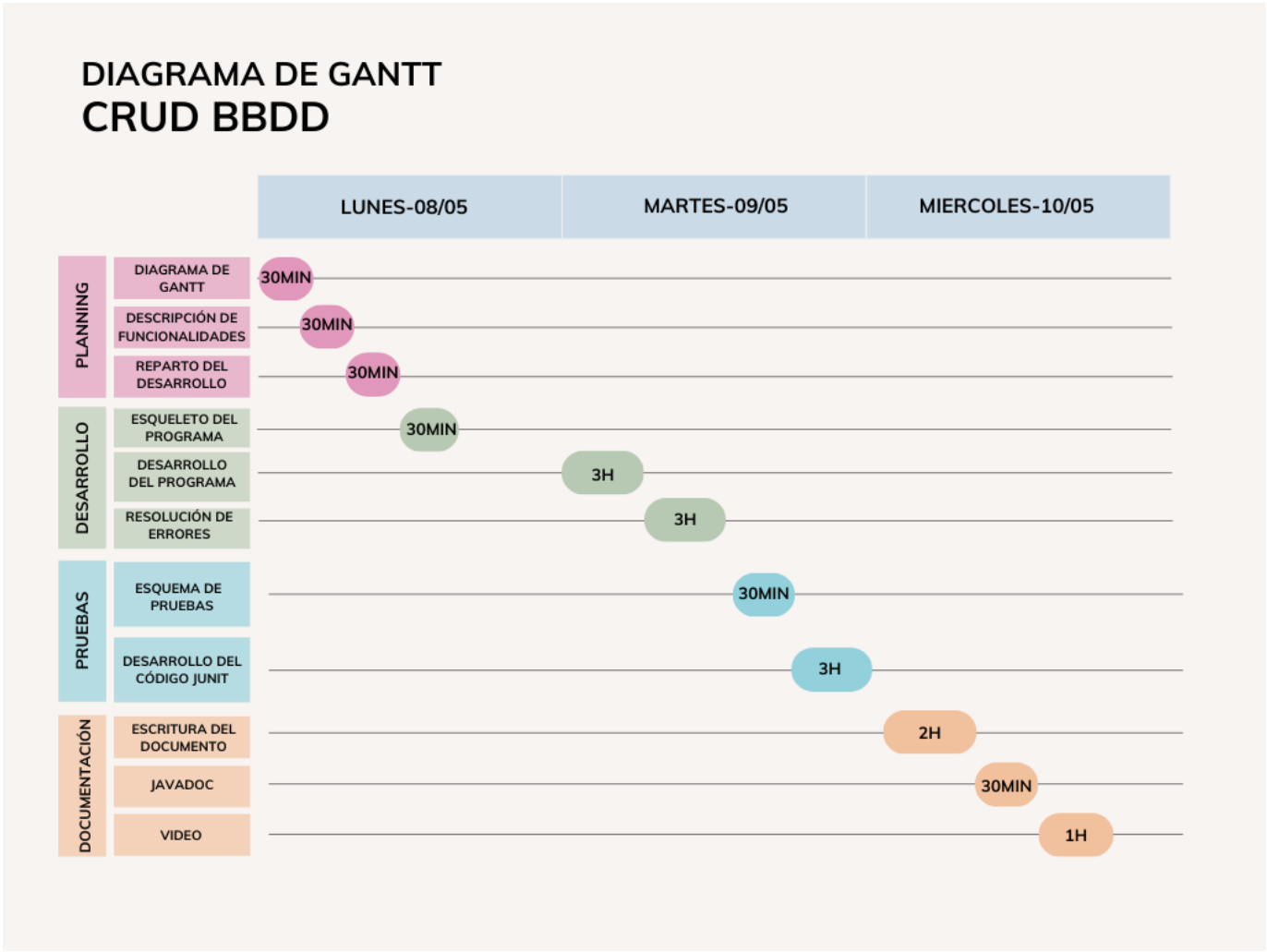
id	**MODIFICARPELI**
Descripción	Comprobar que el método recibe un nuevo objeto y lo modifica por el objeto elegido
Condiciones previas	Creación de objeto1 y objeto auxiliar, inserción del objeto auxiliar en el nombre del objeto1.
Gravedad	Crítica
Entorno	Eclipse
Pasos prueba	1.Crear objetos 2. Objetos introducidos
Resultado esperado	Aux está en la bbdd y peli2 ya no está
Estado	Aprobado

- EliminarPeli() → Declaramos el objeto peli1 y aux para tenerlos como referencia. Probaré que los objetos se insertan correctamente y que posteriormente se eliminan de la base de datos.
  1. assertEquals → Indica que el objeto peli3 todavia no está en la base de datos
  2. assertNotEquals → Una vez introducido en la base de datos se comprueba que el resultado del método buscar() es distinto de “nulo”.
  3. assertEquals → Una vez utilizado el método borrarpeli(), comprobar que el objeto peli3 ya no existe y es igual “nulo”.

id	**ELIMINARPELI**
Descripción	Comprobar que el método eliminar el objeto de la base de datos
Condiciones previas	Creación de objeto, inserción del objeto y eliminación.
Gravedad	Crítica
Entorno	Eclipse
Pasos prueba	1.Crear objetos 2. Insertar objeto 3. Eliminar objeto
Resultado esperado	Peli3 ya no está en la base de datos
Estado	Aprobado

- BUSCARPELI() → Declararemos el objeto “peli1” y “aux” y los insertamos en la base de datos. Mediante la consulta prestablecida en el método original, SELECT * FROM película WHERE nombre=”nombre”, comprobaremos si los objetos anteriormente declarados están dentro de la base de datos.
    1. assertNotEquals → Comprueba que el objeteo “peli” está dentro del registro
    2. assertNotEquals → Comprueba que el objeteo “aux” está dentro del registro
    3. assertEquals → Comprueba que el nombre del objeto “peli1” es igual al que se ha establecido en el registro de la base de datos.
    4. assertEquals → Comprueba que el nombre del objeto “aux” es igual al que se ha establecido en el registro de la base de datos.

id	**BUSCARPELI**
Descripción	Comprobar que el método busca correctamente
Condiciones previas	Introducir varios objetos
Gravedad	Crítica
Entorno	Eclipse
Pasos prueba	1.Crear objetos 2. Insertar objetos.
Resultado esperado	Peli1 y aux, están dentro de la base de datos
Estado	Aprobado



PRUEBAS VIDEO

Enlace: https://youtu.be/xENT85UDXMY

GIT

Enlace: https://github.com/MiniMestri/ActividadCRUDBBDD.git