Informe: práctica 02

Tema: Bases de Datos

Nota

Estudiantes	Escuela	Asignatura
Jhonatan Benjamin Mamani Céspedes y Álvaro Raúl Quispe Condori	Escuela Profesional de Ingeniería de Sistemas	Práctica Semestre: II Código: 20230467

Práctica	Tema	Duración
02	Bases de Datos	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	14 de enero	16 de enero

1. Actividades

- Crear una base de datos con MARIADB.
- Conectarse a una base de datos con java.
- Usar Singleton para conectarse con un único objeto.

2. Equipos, materiales y temas empleados

- Sistema Operativo POP os, 22.04 LTS.
- NVIM 0.6.1.
- OpenJDK 64-Bits 17.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.

3. URL de Repositorio Github y video-grabación

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/ALVARO-QUISPE-UNSA/fp2-23b`
- URL para la práctica 02 en el Repositorio GitHub.
- `https://github.com/ALVARO-QUISPE-UNSA/fp2-23b/tree/main/fase03/prac02`
- URL para la grabación de la compilación
- `https://drive.google.com/file/d/1CE_5pEXvPoFedSczyST0HGrHY-bEKgYR/view?usp=sharing`

4. Desarrollo de la actividad

4.1. Conexión a la base de datos

- Se ha escrito un programa en Java que utiliza el controlador JDBC de MySQL para establecer una conexión con una base de datos MariaDB local.
- La clase `Class.forName(çom.mysql.cj.jdbc.Driver")` se ha empleado para cargar el controlador JDBC necesario para la conexión.
- La URL de conexión se ha definido como jdbc : mysql://localhost:3306/fp2-23b, especificando la dirección del servidor, el puerto y el nombre de la base de datos.
- Se han proporcionado las credenciales de usuario (`user`) y contraseña (`password`) para autenticarse en la base de datos.
- Se ha utilizado la clase `SingleDriver` para obtener la conexión mediante el método `getDriver`, aunque la implementación exacta de esta clase no se presenta en el código proporcionado.

Listing 1: Main.java

```java
import java.sql.*;
public class Main {
  public static void main(String args[]) {
    try {
      // Corrección del nombre del controlador
      Class.forName("com.mysql.cj.jdbc.Driver");

      // Conexión con la base de datos
      String url = "jdbc:mysql://localhost:3306/fp2_23b";
      String user = "fp2_23b";
      String password = "12345678";
      Connection miConexion = SingleDriver.getDriver(url, user, password);

      if (miConexion != null) {
        // Operaciones con la conexión exitosa

        // Declaración y ejecución de la consulta SQL
        String consultas = "SELECT first_name, last_name FROM owners";
        Statement statement = miConexion.createStatement();
        ResultSet resultados = statement.executeQuery(consultas);

        // Procesamiento de los resultados
```

```
        while (resultados.next()) {
          String firstName = resultados.getString("first_name");
          String lastName = resultados.getString("last_name");
          System.out.println("First Name: " + firstName + ", Last Name: " + lastName);
        }
      } else {
        // Manejo del caso de conexión fallida
        System.out.println("La conexión no se pudo establecer");
      }

    // Manejo de las excepciones
    } catch (SQLException e) {
      System.err.println("\u001B[31mError: \n\u001B[0m" + e);
    } catch (ClassNotFoundException e) {
      System.err.println("\u001B[31mError: \n\u001B[0m" + e);
    }
  }
}
```

4.2. Implementación del Modelo Singleton en la Clase `SingleDriver`

- La clase `SingleDriver` ha sido diseñada siguiendo el patrón de diseño Singleton, que garantiza la existencia de una única instancia de la conexión a la base de datos.
- La instancia única de la conexión (`driver`) se declara como un campo privado y estático, asegurando que sea accesible a través de la clase y no a través de instancias de la misma.
- El constructor de la clase es privado (`private SingleDriver()`), lo que evita la creación de instancias adicionales desde fuera de la clase.
- El método estático `getDriver` se encarga de proporcionar la conexión a la base de datos. Si la conexión aún no ha sido inicializada, se crea una nueva conexión utilizando `DriverManager.getConnection(url, user, pass)` y se asigna a la instancia única de la conexión. Si la conexión ya existe, se devuelve la instancia existente.
- En caso de que ocurra una excepción `SQLException` durante la obtención de la conexión, se imprime un mensaje de error en la consola y se devuelve `null` para indicar que ha ocurrido un error en la obtención de la conexión.
- En resumen, la implementación del modelo Singleton en la clase `SingleDriver` asegura que solo exista una instancia de la conexión a la base de datos, proporcionando un mecanismo eficiente y controlado para acceder y gestionar dicha conexión.

Listing 2: SingleDriver.java

```
import java.sql.*;
public class SingleDriver {
  // Una única instancia de la conexión
  private static Connection driver;

  // Constructor privado para evitar instancias adicionales de la clase
  private SingleDriver() {}

  // Método estático para obtener una conexión a la base de datos
  public static Connection getDriver(String url, String user, String pass) {
```

```
    try {
      // Si la conexión no ha sido inicializada, crea una nueva conexión
      if (driver == null)
        driver = DriverManager.getConnection(url, user, pass);
      return driver;
    } catch (SQLException e) {
      // Si ocurre un error al intentar obtener la conexión, maneja la excepción
      System.err.println("\u001B[31mError: \n\u001B[0m" + e);
      // Devuelve null para indicar que ha ocurrido un error en la obtención de la conexión
      return null;
    }
  }
}
```

Listing 3: Compilación y ejecución

```
$ javac Main.java
$ java -cp .:/usr/share/java/mysql-connector-java-8.2.0.jar Main
First Name: George, Last Name: Franklin
First Name: Betty, Last Name: Davis
First Name: Eduardo, Last Name: Rodriquez
First Name: Harold, Last Name: Davis
First Name: Peter, Last Name: McTavish
First Name: Jean, Last Name: Coleman
First Name: Jeff, Last Name: Black
First Name: Maria, Last Name: Escobito
First Name: David, Last Name: Schroeder
First Name: Carlos, Last Name: Estaban
```

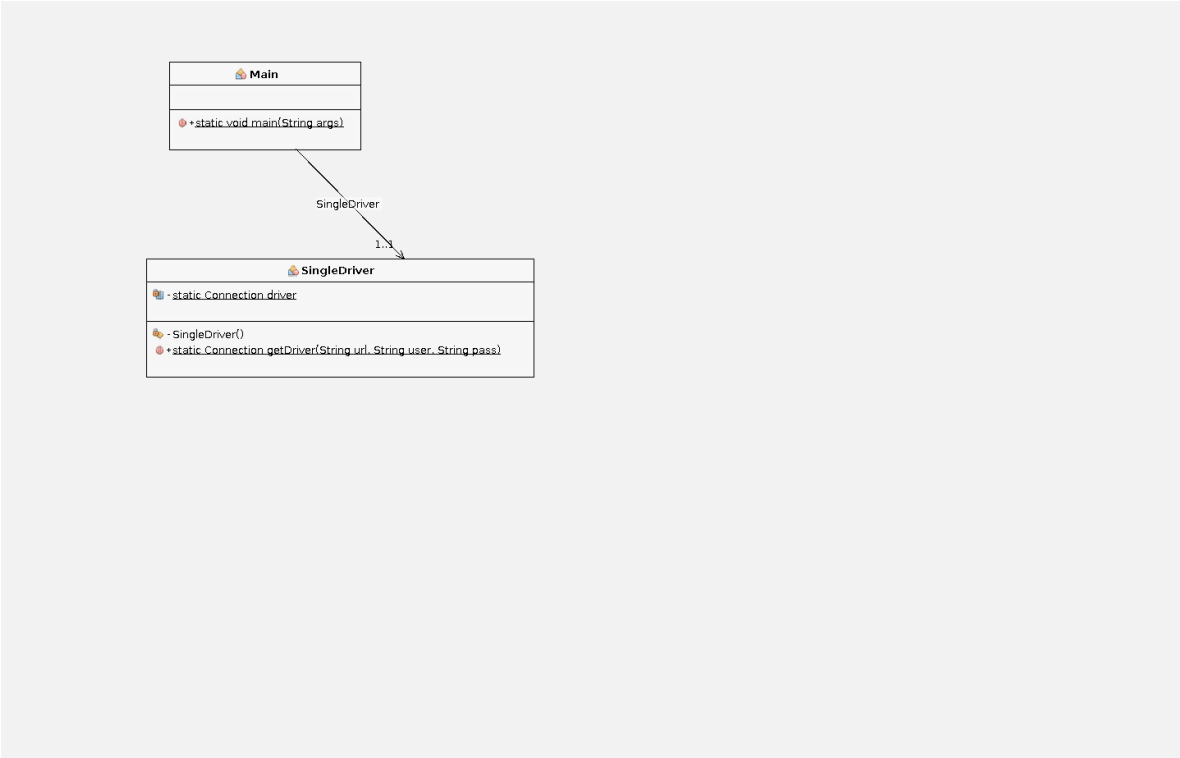
5. Estructura de práctica 2

- El contenido que se entrega en esta práctica es el siguiente:

```
.
|-- lib
|   |-- mysql-connector-j-8.3.0.jar
|-- Main.java
|-- SingleDriver.java
```

6. Diagrama UML del proyecto

- La estructura del proyecto se refleja en el siguiente diagrama UML



7. Referencias

- `https://docs.oracle.com/javase/8/docs/api/`

8. Rúbricas

8.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

8.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	1	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
Total		20		19	