Informe de Practica 02

Tema: Practica-02

Nota

Grupo	Escuela	Asignatura
Victor Mamani Anahua Rafael Nina Calizaya vmamanian@unsa.edu.pe rninacal@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la Programación II Semestre: II Código: ———

Practica	Tema	Duración
02	Practica-02	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 15 Enero de 2024	Al 16 Enero de 2024

1. Tarea

- El uso Base de Datos.
- Como conectar Java con la base de datos
- El uso del patron de diseño Singleton para la optimizacion en la creacion de objetos.

2. Equipos, materiales y temas utilizados

- Sistema Operativo Ubuntu GNU Linux 23 lunar 64 bits Kernell 6.2.v
- Visual Studio Code.
- VIM 9.0.
- OpenJDK 64-Bits 19.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Actividades del Prac02.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/VictorMA18/fp2-23b.git`
- URL para la prac02 en el Repositorio GitHub.
- `https://github.com/VictorMA18/fp2-23b/tree/main/Fase03/Examen-Prac02`

4. Actividades de la Prac02

4.1. Ejercicio Conectar

- El código Java proporciona una implementación básica de un patrón Singleton para la gestión de conexiones a una base de datos MySQL. La clase Conectar utiliza el patrón Singleton para garantizar la existencia de una única instancia de la conexión a la base de datos. La clase contiene un método estático getInstance() que devuelve la instancia única de la clase, y un método conexion() que establece la conexión a la base de datos MySQL local utilizando el controlador com.mysql.jdbc.Driver. La conexión se establece con credenciales específicas (usuario: root", contraseña:) y la base de datos "$fp2_23b$"

  Cabe mencionar que el código presenta una redundancia en la implementación del patrón Singleton con la presencia de dos campos estáticos singleton y instancia, y la ausencia de una variable de instancia real, lo cual podría llevar a confusión y no sigue estrictamente el patrón. Además, el código maneja las excepciones generales sin detallar mensajes específicos, lo cual podría dificultar la depuración en caso de errores.

Listing 1: Las lineas de codigos del metodo creado:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;

public class Conectar{
   private static Conectar singleton = null;
   public static synchronized Conectar getInstance(){
      if (singleton == null)
         singleton = new Conectar();
      return singleton;
   }
   private static Conectar instancia;
   Connection conectar = null;
   public Connection conexion(){
      try{
         Class.forName("com.mysql.jdbc.Driver");
         conectar = (Connection)
             DriverManager.getConnection("jdbc:mysql://localhost:3306/fp2_23b","root","");
         System.out.println("Conexion aceptada");
      } catch (Exception e){
         System.out.println(e);
      }
      return conectar;
   }
}
```

4.2. Ejercicio Main

- El código Java presenta una comparación entre la creación de instancias de la clase Conectar sin utilizar el patrón Singleton y utilizando el patrón Singleton. En la primera parte del programa, se crean tres instancias separadas de Conectar mediante el constructor, y se imprime el hashcode de cada instancia junto con la conexión a la base de datos. En la segunda parte, se utiliza el método estático getInstance() del patrón Singleton para obtener instancias de Conectar, y se imprime el hashcode junto con la conexión a la base de datos. Al usar el Singleton, se observa que las instancias tienen el mismo hashcode, indicando que se trata de la misma instancia de la clase Conectar, lo cual es característico del patrón Singleton. Este enfoque evita la creación innecesaria de múltiples instancias y garantiza una única conexión a la base de datos, promoviendo la eficiencia en la gestión de recursos.

Listing 2: Las lineas de codigos del metodo creado:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Main {
   public static void main(String[] args){
      System.out.println("Sin Singleton");
      Conectar conn1 = new Conectar();
      System.out.println("El hashcode de conn es : " + conn1.hashCode());
      conn1.conexion();
      Conectar conn2 = new Conectar();
      System.out.println("El hashcode de conn es : " + conn2.hashCode());
      conn2.conexion();
      Conectar conn3 = new Conectar();
      System.out.println("El hashcode de conn es : " + conn3.hashCode());
      conn3.conexion();
      System.out.println("Usando Singleton");
      Conectar conn4 = Conectar.getInstance();
      System.out.println("El hashcode de conn es : " + conn4.hashCode());
      conn4.conexion();
      Conectar conn5 = Conectar.getInstance();
      System.out.println("El hashcode de conn es : " + conn5.hashCode());
      conn5.conexion();
      Conectar conn6 = Conectar.getInstance();
      System.out.println("El hashcode de conn es : " + conn6.hashCode());
      conn6.conexion();
   }
}
```

```
user@victus:~/fp2-23b/Fase03/Examen-Prac02$ cd /home/user/fp2-23b/Fase03/Examen-Prac02 ; /usr/bin/env /usr/lib/jvm/java-19-openjdk-amd64/bin/java -XX:+ShowCodeDetailsIn
ExceptionMessages -cp /home/user/.config/Code/User/workspaceStorage/98737364007893d652d9e9325fe22337/redhat.java/jdt_ws/jdt.ls-java-project/bin p2.Main
Sin Singleton
El hashcode de conn es : 2060468723
java.lang.ClassNotFoundException: com.mysql.jdbc.Driver
El hashcode de conn es : 349885916
java.lang.ClassNotFoundException: com.mysql.jdbc.Driver
El hashcode de conn es : 414493378
java.lang.ClassNotFoundException: com.mysql.jdbc.Driver
Usando Singleton
El hashcode de conn es : 1984697014
java.lang.ClassNotFoundException: com.mysql.jdbc.Driver
El hashcode de conn es : 1984697014
java.lang.ClassNotFoundException: com.mysql.jdbc.Driver
El hashcode de conn es : 1984697014
java.lang.ClassNotFoundException: com.mysql.jdbc.Driver
```

4.3. Links Drive y los commits

- Aca dejaremos el link del video Drive y el commit
- `https://drive.google.com/file/d/1Qde24VzpziKqNg6vQzDnIRyhvrLZiPSU/view?usp=sharing`

Listing 3: Commit

```
$ git commit -m "Agregando los archivos para la practica02"
$ git commit -m "Imprimiendo casos donde se aplican objetos creados con y sin singleton"
```

4.4. Estructura de EXAMEN-Prac02

- El contenido que se entrega en esta practica02 es el siguiente:

```
/EXAMEN-PRAC02
|----Conectar.java
|----Latex
       |----img
              |----logo_abet.png
              |----logo_episunsa.png
              |----logo_unsa.jpg
              +----Screen.png
       |----InformePrac-02.aux
       |----InformePrac-02.fdb_latexmk
       |----InformePrac-02.fls
       |----InformePrac-02.log
       |----InformePrac-02.out
       |----InformePrac-02.pdf
       |----InformePrac-02.synctex.gz
       +----InformePrac-02.tex
+----Main.java
```

5. Rúbricas

5.1. 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.

5.2. 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 **Grupo** 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	2.5	
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	2	
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	2	
Total		20		16.5	