AINTZINAKO ANDRA MARI IKASTETXEA MURIALDOKO JOSEFINOAK COLEGIO NTRA. SRA. DE LA ANTIGUA JOSEFINOS DE MURIALDO	Asignatura	Acceso a Datos	NOTA
	Evaluación	1ª Evaluación	
	Fecha	13 de Noviembre de 2024	
Nombre y Apellidos			

(Todos los ejercicios valen 2 puntos, salvo el primero que vale 1 y el cuarto que vale 3)

1. **Ejercicio de Escritura en Archivo de Texto (1 puntos)**
   a. Escribe un método en Java que reciba un `List<String>` de nombres de series famosas y un nombre de archivo, y escribe cada nombre de serie en una línea del archivo de texto especificado.

```
public static void escribirNombres(List<String> nombres, String nombreArchivo) {
    // Escribe aquí tu código
}
```

Sugerencias:

- ✓ En el método **escribirNombres** usa **buffers** para mejorar el rendimiento.
- ✓ Este es el **main** donde se crea una lista con nombres de series famosas y llama al método de escritura para guardarlos en el archivo **series.txt**.

```
public static void main(String[] args) {
      // Lista de nombres de ejemplo
      List<String> nombresSeries = new ArrayList<>();
      nombresSeries.add("Stranger Things");
      nombresSeries.add("Outer Banks");
      nombresSeries.add("One Tree Hill");

      // Llamar al método para escribir en el archivo de texto
      escribirNombres (nombresSeries, "series.txt");
}
```

- ✓ Utiliza try-with-resources ó implementa un bloque try-catch-finally pero no olvides manejar errores correspondientes a los ficheros y cerrar objetos.

2. **Ejercicio de Escritura en Archivo Binario (2 puntos)**
   a. Crea una clase llamada **Alumno** que implemente la interfaz Serializable y que tenga los siguientes atributos:

```
public class Alumno implements Serializable {
      private String nombre;
      private int edad;
      private String curso;
      // Crea Constructor, getters y setters
      // Escribe aquí tu código
}
```

   b. Crea un método llamado **escribirAlumnosEnArchivo** que reciba una lista de objetos Alumno y un nombre de archivo, y escriba los objetos en el archivo binario especificado, **alumnos.dat**.

```
public static void escribirAlumnosEnArchivo(List<Alumno> listaAlumnos, String
nombreArchivo) {
      // Escribe aquí tu código
}
```

Sugerencias:

- ✓ En el método **escribirAlumnosEnArchivo** usa el método correspondiente para escribir la lista de objetos en el archivo binario.
- ✓ Este es el **main** para crear algunos objetos Alumno (3 objetos), agregarlos a una lista de objetos y llamar al método de escritura para guardarlos en el archivo alumnos.dat.

```
public static void main(String[] args) {
      // Crear una lista de alumnos
      List<Alumno> listaAlumnos = new ArrayList<>();
      // Añade 3 alumnos a la lista
      listaAlumnos.add(new Alumno("Pedro", 21, "DAM2"));
      listaAlumnos.add(new Alumno("Laura", 23, "DAM2"));
      listaAlumnos.add(new Alumno("Marta", 20, "DAM2"));

      // Llama al método escribir
      escribirAlumnosEnArchivo(listaAlumnos, "alumnos.dat");
}
```

- ✓ Utiliza try-with-resources ó implementa un bloque try-catch-finally pero no olvides manejar errores correspondientes a los ficheros y cerrar objetos.

3. Lectura de un Archivo XML (2 puntos)

a. Escribe un método **leerSuperheroesDesdeXML** en Java que lea un fichero XML llamado **superheroes.xml** y muestre en la consola los nombres de los superhéroes que contiene.

```
public static void leerSuperheroesDesdeXML (String rutaFichero) {
          // Escribe aquí tu código
}
```

Sugerencias:

- ✓ Para leer un archivo XML en Java y mostrar los nombres de los superhéroes que contiene, utiliza la API DOM de Java para cargar y analizar el archivo XML.
- ✓ Recuerda obtener todos los elementos con la etiqueta superheroe e iterar sobre la lista de elementos para obtener el valor del elemento nombre.
- ✓ Este es el **main** para leer el archivo xml.

```
public static void main(String[] args) {
        leerSuperheroesDesdeXML("superheroes.xml");
}
```

- ✓ Y este es el fichero XML de superhéroes, superheroes.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<superheroes>
      <superheroe>
                <nombre>Iron Man</nombre>
                <identidad_secreta>Tony Stark</identidad_secreta>
                <bando>héroe</bando>
      </superheroe>
      <superheroe>
                <nombre>Spider-Man</nombre>
                <identidad_secreta>Peter Parker</identidad_secreta>
                <bando>héroe</bando>
      </superheroe>
</superheroes>
```

4. Ejercicio de Conexión y Consulta a Base de Datos (3 puntos)

Dado el siguiente script en MySQL que crea la base de datos **Escuela**, la tabla **Alumnos** y dos nuevos registros, se pide:

```
CREATE DATABASE Escuela;
USE Escuela;
CREATE TABLE Alumnos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100),
    edad INT,
    email VARCHAR(100)
);
INSERT INTO Alumnos (nombre, edad, email)
VALUES ('Alberto', 20, 'a.fernandez@colegiojosefinos.com');
INSERT INTO Alumnos (nombre, edad, email)
VALUES ('Susana', 19, 's.perez@colegiojosefinos.com');
```

Implementar las siguientes operaciones CRUD utilizando JDBC:

a. Escribe un método en Java que inserte un nuevo alumno en la tabla con tus datos y muestre por pantalla el número de filas afectadas.

```
public void insertarAlumno(String nombre, int edad, String email) {
      // Escribe aquí tu código
}
```

b. Escribe un método en Java que muestra todos los alumnos almacenados en la base de datos y muestre el nombre y edad de cada alumno en consola.

```
public void mostrarTodosLosAlumnos() {
      // Escribe aquí tu código
}
```

c. Escribe un método en Java que actualice la edad de un alumno dado su ID, modifica el que quieras para probar el método.

```
public void actualizarEdad(int id, int nuevaEdad) {
      // Escribe aquí tu código
}
```

d. Crea una clase principal **main** para probar los métodos.

Sugerencias:

- ✓ Acuérdate de añadir lo que necesites en tu proyecto.
- ✓ Abre y cierra la conexión en cada método.
- ✓ Utiliza **PreparedStatement** para prevenir inyecciones SQL en las sentencias SQL.
- ✓ Recuerda usar los métodos **set** de PreparedStatement para asignar valores a cada marcador de posición (?) de tu consulta.
- ✓ Recuerda dependiendo del tipo de operación, usa uno de estos métodos:
    - Consulta (SELECT): Usa **executeQuery**() para obtener un **ResultSet** con los resultados.
    - Actualización (INSERT, UPDATE, DELETE): Usa **executeUpdate**() para obtener el número de filas afectadas.
- ✓ Utiliza try-with-resources ó implementa un bloque try-catch-finally pero no olvides manejar errores y cerrar objetos.

5. Manejo de transacciones (2 puntos)

a. Crear un programa en Java que realice una transferencia de dinero entre dos cuentas en una base de datos MySQL. La operación debe ser transaccional, es decir, debe asegurar que ambas actualizaciones (deducir de la cuenta A y sumar a la B) se realicen correctamente o que ninguna de las dos se realice si hay un error.

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

public class TransferenciaDinero {

    private static final String URL = " "; //Añade aquí tu URL
    private static final String USER = " "; //Añade aquí tu usuario
    private static final String PASSWORD = " "; //Añade aquí tu contraseña

    public static void main(String[] args) {
        int cuentaOrigen = 1; // ID de Cuenta A
        int cuentaDestino = 2; // ID de Cuenta B
        double cantidad = 200.00; // Cantidad a transferir

        realizarTransferencia(cuentaOrigen, cuentaDestino, cantidad);
    }

    public static void realizarTransferencia(int cuentaOrigen, int cuentaDestino, double
cantidad) {
         // Escribe aquí tu código
    }
}
```

Sugerencias:

- ✓ Acuérdate de añadir lo que necesites en tu proyecto.
- ✓ Usa **setAutoCommit(false)** para desactivar el modo de auto-confirmación.
- ✓ Implementa las dos operaciones UPDATE :
  - String sqlUpdateCtaA = "UPDATE cuentas SET saldo = saldo - ? WHERE id = ?"
  - String sqlUpdateCtaB = "UPDATE cuentas SET saldo = saldo + ? WHERE id = ?"
- ✓ Usa **commit()** para confirmar los cambios o **rollback()** para deshacer los cambios si hay errores.
- ✓ Usa el bloque try-catch-finally para garantizar que se realice commit o rollback según corresponda.
- ✓ Recuerda volver activar el modo de auto-confirmación, **setAutoCommit(true),** al finalizar.
- ✓ Utiliza este script de MySQL para crear la BBDD:

```
CREATE DATABASE banco;

USE banco;

CREATE TABLE cuentas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    saldo DECIMAL(10, 2) NOT NULL
);

INSERT INTO cuentas (nombre, saldo) VALUES ('Cuenta A', 1000.00);
INSERT INTO cuentas (nombre, saldo) VALUES ('Cuenta B', 500.00);
```