Nom i Cognom:	**Victor Prieto Gallegos**
Enllaç al repositori Git:	**https://github.com/vipriga06/DAM2-AccesDadesVPrieto/tree/main/RA2/PT2.1**

Objectius:
- Gestionar BD sqlite amb JDBC

Instruccions:
- Desenvolupa el codi necessari per a cada exercici seguint les especificacions indicades. El codi ha de superar els tests proporcionats per garantir-ne la funcionalitat.
- En cas d'haver de contestar alguna pregunta en aquest document, cal fer-ho dins del quadre indicat.

Criteris d'avaluació:
Cada exercici te especificada la puntuació. Es valorarà:
- Format correcte del codi (indentació i compliment de la guia d'estil de Java).
- Noms clars i descriptius per a mètodes i variables.
- Qualitat general del codi, amb comentaris explicatius quan sigui necessari.
- Les còpies seran penalitzades amb una puntuació de 0.

Entrega:
- Repositori Git privat, compartit amb l'usuari jpala4-ieti, que contingui el codi complet de la pràctica.
- Subdirectori "doc" dins del repositori amb el fitxer memoria.pdf.
- URL del repositori: S'ha de pujar a Moodle.

Recursos i materials:
- JDK de Java versió 21, Maven i Git instal·lats.
- Eina de programació: necessiteu una IDE per programar en Java (VS Code, IntelliJ, o una altra).
- Altres recursos: Podeu utilitzar Google per buscar tutorials que us ajudin a resoldre els exercicis.
- Repositori amb exemples d'SQLite: https://github.com/jpala4-ieti/DAM-M0486-Tema2-RA234-ExempleJavaSQLite-25-26

Punt de partida de la pràctica:

https://github.com/jpala4-ieti/DAM-M0486-Tema2-Esquelet-Basic-Projecte-Java-Maven-25-26

Objectiu del programa: Desenvolupar una aplicació Java que gestioni una base de dades de personatges i faccions basada en el videojoc "For Honor". Utilitza JDBC i SQLite.

Pots crear les classes que creguis necessàries per organitzar bé l'aplicació.

Exercici 0 - Aplicació "PR210Honor.java"

Inicialització de la Base de Dades:

- Verifica si la base de dades existeix.
- En cas contrari, inicialitza la base de dades amb les taules següents:
    - Taula Facció:
        - **id:** Clau primària, tipus enter.
        - **nom:** Cadena de text (varchar) de màxim 15 caràcters.
        - **resum:** Cadena de text (varchar) de màxim 500 caràcters.
    - Taula Personatge:
        - **id:** Clau primària, tipus enter.
        - **nom:** Cadena de text (varchar) de màxim 15 caràcters.
        - **atac:** Número real.
        - **defensa:** Número real.
        - **idFaccio:** Clau forània que enllaça amb l'id de la taula Facció.

Població de la Base de Dades:

Des del mateix programa `PR210Honor.java` insereix dades a la BD. Pots usar les comandes SQL de l'annex SQL Inserció de dades a la BD

Per tenir més informació sobre el joc pots consultar: For Honor - Wikipedia i fandom

Funcionalitats del Programa:

- **Mostrar una taula:** Permet l'usuari seleccionar quina taula vol visualitzar.
- **Mostrar personatges per facció:** Llista els personatges pertanyents a una facció específica. Indica el nom de la facció no només l'id.
- **Mostrar el millor atacant per facció*:** Indica quin personatge té l'atac més alt dins d'una facció seleccionada.
- **Mostrar el millor defensor per facció:** Indica quin personatge té la defensa més alta dins d'una facció seleccionada.
- **Sortir:** Tanca l'aplicació.

*En cas d'empat en atac o defensa, mostrar només un dels personatges.

Caldrà que trobis una forma de generar un diagrama de la base de dades.

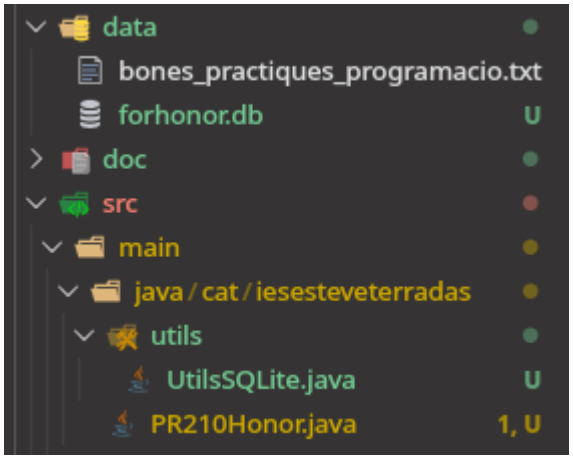
Enganxa en aquest quadre el diagrama de la base de dades

Faccio
- id (INTEGER, PRIMARY KEY, AUTOINCREMENT)
- nom (VARCHAR(15), NOT NULL)
- resum (VARCHAR(500))

Personatge
- id (INTEGER, PRIMARY KEY, AUTOINCREMENT)
- nom (VARCHAR(15), NOT NULL)
- atac (REAL)
- defensa (REAL)
- idFaccio (INTEGER, FOREIGN KEY REFERENCES Faccio(id))

Enganxa una captura de pantalla que mostri l'organització del teu programa (packages, classes) i explica les principals decisions de disseny que has pres.

ESTRUCTURA DE PAQUETS:



ESTRUCTURA DE CLASSES:

```
class PR210Honor {
  -Logger logger
  -String DB_PATH
  +main(String[] args) void
  +initDatabase() void
  -mostrarMenu() void
  -mostrarTaula(Connection, Scanner) void
  -seleccionarFaccio(Connection, Scanner) int
  -mostrarPersonatgesPerFaccio(Connection, Scanner) void
  -mostrarMillorAtacantPerFaccio(Connection, Scanner) void
  -mostrarMillorDefensorPerFaccio(Connection, Scanner) void
}
class UtilsSQLite {
  <<utility>>
  +connect(String dbPath) Connection
  +queryUpdate(Connection conn, String query) void
  +queryUpdatePS(Connection conn, String query, Object... params) void
  +querySelect(Connection conn, String query) ResultSet
  +querySelectPS(Connection conn, String query, Object... params) ResultSet
}
class Logger {
  <<interface>>
  +info(String message) void
  +error(String message) void
  +debug(String message) void
}
class Connection {
  <<interface>>
  +close() void
}
class ResultSet {
  <<interface>>
  +next() boolean
  +getInt(String column) int
  +getString(String column) String
  +getDouble(String column) double
}
class Scanner {
  +nextInt() int
  +nextLine() String
  +hasNextInt() boolean
  +close() void
}
class File {
  +exists() boolean
  +mkdirs() boolean
}
class SQLException {
  <<exception>>
  +getMessage() String
}
```

RELACIONS DE DEPÈNDENCIA

PR210Honor --> UtilsSQLite : utilitza
PR210Honor --> Logger : utilitza
PR210Honor --> Connection : utilitza
PR210Honor --> ResultSet : utilitza
PR210Honor --> Scanner : utilitza
PR210Honor --> File : utilitza
PR210Honor --> SQLException : maneja
UtilsSQLite --> Connection : retorna
UtilsSQLite --> ResultSet : retorna

EXPLICACIÓ DE DECISIÓ DE DISSENY

- **Separació en capes:** Si canviem de SQLite a MySQL, només cal modificar UtilsSQLite.java sense tocar la lògica del programa.
- **PreparedStatement:** Si un usuari introdueix "; DROP TABLE Faccio;-- com a entrada, el PreparedStatement ho escapa i no s'executa com a SQL.
- **Inicialització automàtica:** L'usuari no ha d'executar scripts SQL manualment; el programa està llest per funcionar des del primer moment.
- **JOIN en consultes:** En lloc de fer 2 consultes (una per personatge, altra per nom de facció), en fem una sola que és més eficient.
- **try-with-resources:** Si hi ha una excepció, la connexió es tanca igualment i no queden recursos oberts consumint memòria.
- **Logging amb SLF4J:** Podem veure exactament quines queries s'executen i quan, útil per trobar errors sense afegir System.out.println temporals.

Amplicació

Afegeix sistema de logging similar al del repositori d'exemple

Annexos

SQL Inserció de dades a la BD

```
INSERT INTO Faccio (nom, resum) VALUES ("Cavallers", "Though seen as a single group, the Knights are hardly
unified. There are many Legions in Ashfeld, the most prominent being The Iron Legion.");
INSERT INTO Faccio (nom, resum) VALUES ("Vikings",   "The Vikings are a loose coalition of hundreds of clans and
tribes, the most powerful being The Warborn.");
INSERT INTO Faccio (nom, resum) VALUES ("Samurais",  "The Samurai are the most unified of the three factions,
though this does not say much as the Daimyos were often battling each other for dominance.");

INSERT INTO Personatge (nom, atac, defensa, idFaccio) VALUES ("Warden",      1, 3, 1);
INSERT INTO Personatge (nom, atac, defensa, idFaccio) VALUES ("Conqueror",   2, 2, 1);
INSERT INTO Personatge (nom, atac, defensa, idFaccio) VALUES ("Peacekeep",   2, 3, 1);

INSERT INTO Personatge (nom, atac, defensa, idFaccio) VALUES ("Raider",    3, 3, 2);
INSERT INTO Personatge (nom, atac, defensa, idFaccio) VALUES ("Warlord",   2, 2, 2);
INSERT INTO Personatge (nom, atac, defensa, idFaccio) VALUES ("Berserker", 1, 1, 2);

INSERT INTO Personatge (nom, atac, defensa, idFaccio) VALUES ("Kensei",  3, 2, 3);
INSERT INTO Personatge (nom, atac, defensa, idFaccio) VALUES ("Shugoki", 2, 1, 3);
INSERT INTO Personatge (nom, atac, defensa, idFaccio) VALUES ("Orochi",  3, 2, 3);
```

FET