**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет   
імені Петра Могили**

**Факультет комп’ютерних наук**

**Кафедра інженерії програмного забезпечення**

**КУРСОВА РОБОТА**

з дисципліни «Організація баз даних»

на тему: **«БД «****JoJo's' Characters»»**

121 – КР.ПЗ.01 – 208.1710809

**Виконав: студент 2 курсу, групи 208**

**спеціальності 121 «Інженерія програмного забезпечення»**\_\_\_\_Зурілов І.М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис, ініціали та прізвище)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

**Керівник: професор,   
Фісун М.Т.**

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_\_

**м. Миколаїв – 2019 рік**

**Зміст**

[Вступ 3](#_Toc10444057)

[РОЗДІЛ 1. Опис предметної сфери та постановка задачі 5](#_Toc10444058)

[РОЗДІЛ 2. Моделі БД 6](#_Toc10444059)

[2.1. Концептуальна модель БД 6](#_Toc10444060)

[2.2. Інфологічна модель (Схема «Сутність – Зв'язок) 7](#_Toc10444061)

[2.3. Даталогічна модель даних 8](#_Toc10444062)

[2.4. Фізична модель БД 14](#_Toc10444063)

[РОЗДІЛ 3. Опис реалізації та склад проекту БД 15](#_Toc10444064)

[3.1. Опис реалізації 15](#_Toc10444065)

[3.2. Склад БД 23](#_Toc10444066)

[Висновки 33](#_Toc10444067)

[Список використаних джерел 34](#_Toc10444068)

[Додаток А. Перевірка на оригінальність 35](#_Toc10444069)

[Додаток Б. Опис програмного коду 36](#_Toc10444070)

# Вступ

Людство, за весь час його існування, встигло вигадати безліч повчальних історій. Але через таку кількість інформації у ній дуже легко заплутатись. Сучасні технології дозволяють подолати цю проблему шляхом створення об’єктних баз даних. Це дозволяє створювати застосунки, що допомагають орієнтуватися та пізнавати великі та комплексні твори мистецтва та проводити їх аналіз легше та логічніше. Така мета була поставлена перед проектуванням бази даних цієї курсової роботи.

Першочергове завдання курсової роботи з об’єктних баз даних є проектування та створення системи управління базою даних(СУБД), що дозволить переглядати та модифікувати систему персонажів всесвіту «JoJo’s Bizzare Adventures» і допоможе познайомитися із цим вигаданим всесвітом ближче. Застосунок має показувати таблиці, у яких чітко виведені дані не тільки про головних героїв, а і про їх оточення (країни, здібності, фракції та, щонайголовніше, - посилання на твори мистецтва, на честь яких були названі персонажі та/або їх здібності).

Системи управління була розроблена за допомогою мови програмування Java, графічний інтерфейс коритувача - Swing та локальна база даних на основі SQLite 3.

Мова програмування Java – це одна із об’єктно-орієнтованих мов програмування. Її характерними особливостями є строга типізація (кожна змінна має бути оголошена із своїм типом) та кросплатформеність. В офіційній реалізації Java-програми компілюються у так званий байт-код, що при виконанні інтерпретується віртуальною машиною Java для вибраної платформи.

База даних(БД) – сукупність даних, організованих відповідно до схеми, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування (за стандартом ISO/IEC 2382:2015). В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

В загальному випадку базою даних можна вважати будь-який впорядкований набір даних.

Мета дослідження: навчитися працювати з Java та частиною мови, що відповідає за взаємодію із базами даних SQLite та мовою запитів SQL, GUI Swing та методами обробки даних за допомогою влаштованих методів та компонентів, локальною базою даних SQLite та мовою запитів SQL, навчитися створювати зрозумілий та приємний користувацький інтерфейс взаємодії користувача та бази даних, створювати зв’язки між таблицями та сутностями.

Завдання дослідження:

1. Створити графічний застосунок для зручного керування базою даних;

2. Наявність у застосунку необхідних функціональних можливостей.

3. Створити стабільну базу даних.

Застосунок було розроблено у середовищі Apache NetBeans IDE 10.0.

# РОЗДІЛ 1. Опис предметної сфери та постановка задачі

*Об’єктом дослідження* курсової роботи є БД JoJo's' Characters (або ж Персонажі ДжоДжо). Цей всесвіт багатий на персонажів та сюжети, в який дуже легко заплутатися новачку, але, тим не менш, цю історію дуже цікаво досліджувати. Все зав’язане на персонажах, їх здібностях та іменах, що посилаються на відомих діячів мистецтва.

*Предметом дослідження* курсової роботи є дані, що додаються, супроводжуються і використовуються користувачем для створення уявлення про всесвіт та дослідження зв’язків між персонажами та діячами мистецтва.

Дана БД містить декілька таблиць: персонажі, станди (надздібності персонажів), фракції, раси, країни, посилання тощо. Персонажі можуть бути на одному боці або ж ні, мають імена та/або здібності, що посилаються на різних/однакових діячів, події кожен раз відбуваються у новій країні.

Безпосереднє перетворення даних здійснюється за допомогою операцій введення, корегування та видалення інформації.

Результат отримується за допомогою запитів у вигляду результатів, які виводяться у таблицю.

Система повинна:

- завантажувати БД;

- відображувати БД у вигляді таблиці;

Дана версія програми має наступні функціональні можливості:

- робити сортування за кожним полем;

- шукати у вибраному полі за вказаним шаблоном;

- вставляти нове поле;

- редагувати вже наявне поле;

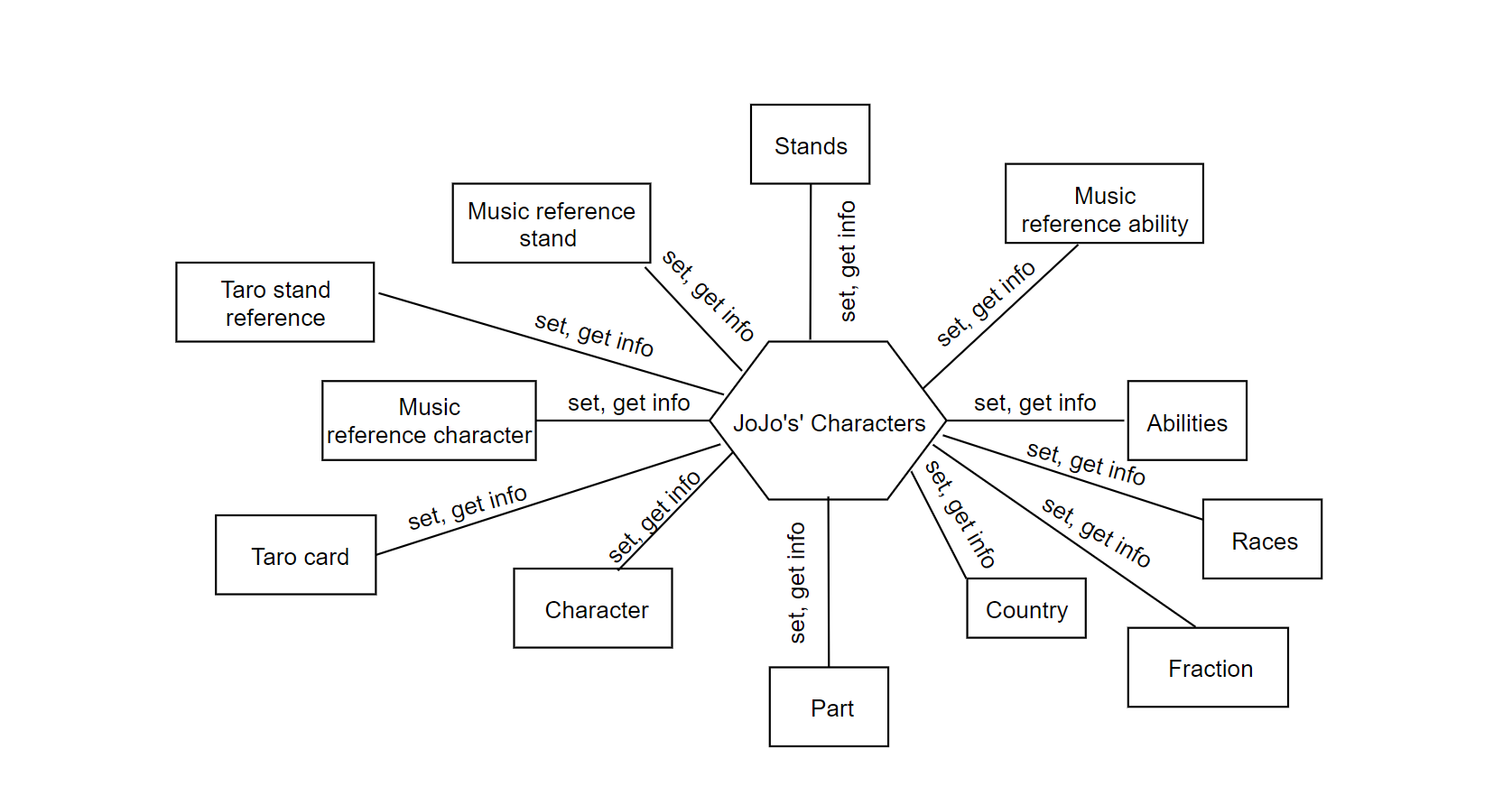
- видаляти поле;

Вимоги до програмного забезпечення ПК: операційна система – Windows 2000/NT/XP/Vista/7/8.1/8/10; віртуальна машина Java 1.8.0, (опціонально, SQLite Studio – для перевірки правильності взаємодії бази даних та програми).

# РОЗДІЛ 2. Моделі БД

2.1. Концептуальна модель БД

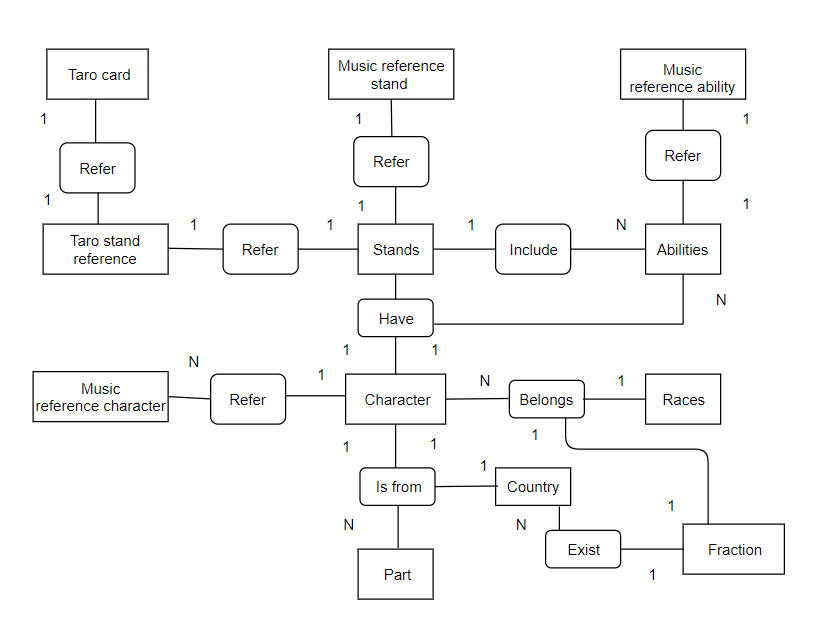
Концептуальна модель (КМ) – модель предметної області, що складається з переліку взаємопов’язаних понять, що використовуються для опису цієї області, разом з властивостями й характеристиками, класифікацією цих понять, за типам ситуацій, ознаками в даній області і законів протікання процесів в ній.

 За допомогою КМ користувачі БД визначають вимоги, які оформлені у вигляді таблиць, схем інформаційних потоків, схем взаємозв’язків між показниками технічно-службової інформації та порядку їх розрахунку та коментарів до них. Таким чином, проектування КМ засновано на аналізі задач обробки даних, що мають розв’язуватися в інформаційній системі.

*Рис. 2.1. Концептуальна модель даних*

## **2.2. Інфологічна модель (Схема «Сутність – Зв'язок)**

Сутність-зв'язок – модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій-блоків, є результатом процесу, який описує предметну область, тобто візуалізує її за допомогою певних зв'зків, які виражають залежності і вимоги між даними.



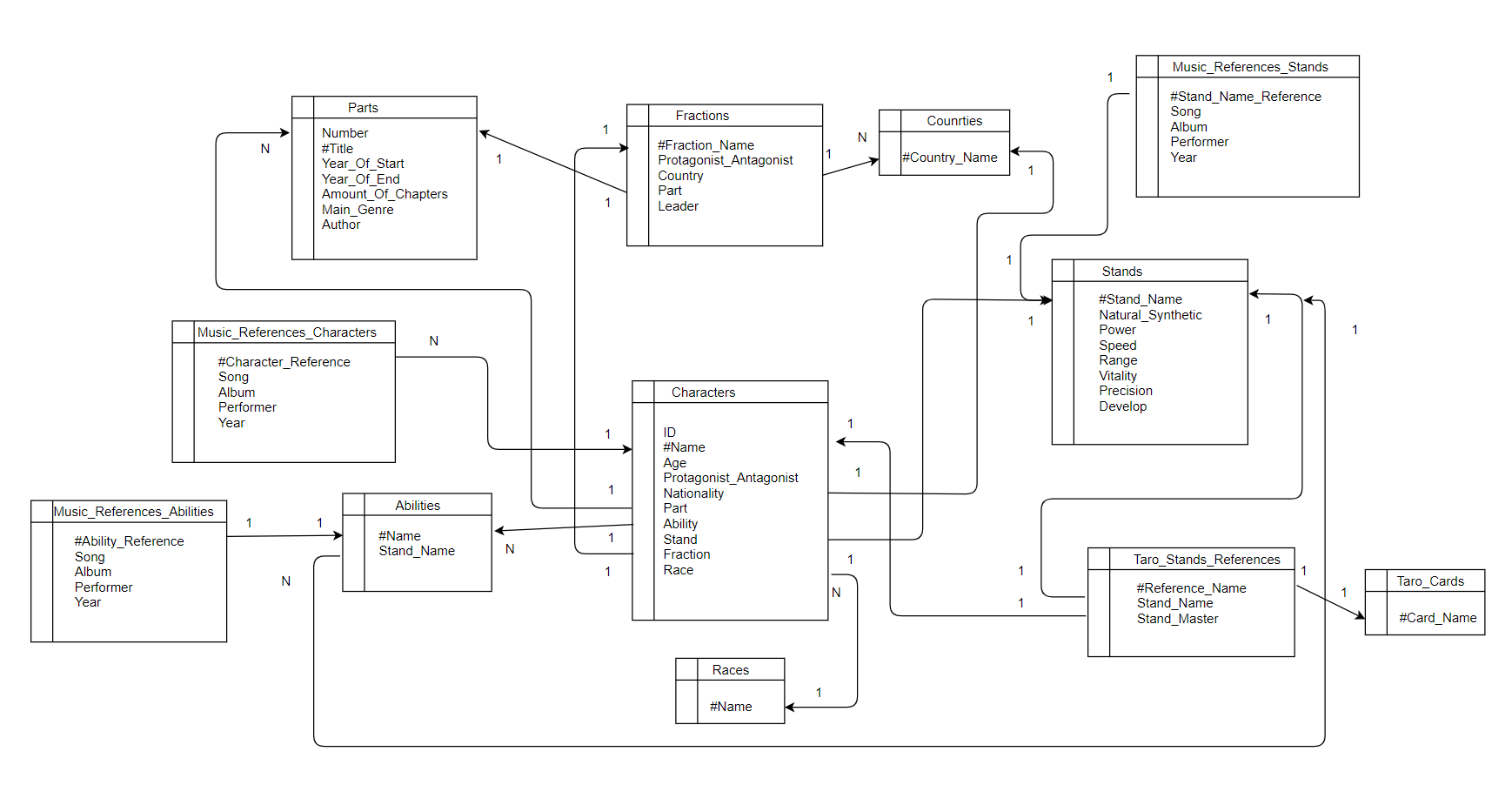
*Рис. 2.2. Інфологічна модель БД персонажів ДжоДжо*

## **2.3. Даталогічна модель даних**

Даталогічна модель є моделлю логічного рівня і являє собою відображення логічних зв’язків між елементами даних не враховуючи їхній зміст та середовище зберігання. Такий тип моделі будується за допомогою інформаційних одиниць, що використовуються та актуальні для конкретної СКБД, у середовищі якої ми проектуємо базу даних. Етап створення даталогічної моделі називається даталогічним проектуванням. Опис логічної структури бази даних мовою СКБД називається схемою.

Хоча даталогічне проектування є логічною структурою бази даних, на нього впливають можливості фізичної організації даних, що представляються конкретною СКБД. Тому знання особливостей фізичної організації даних є корисним при проектуванні логічної структури.

Логічна структура бази даних і сама заповнена даними база даних є відображенням реальної предметної області. Тому на вибір проектних розв’язків найбезпосередніший вплив виявляє специфіка відображуваної предметної області, яка показана в інфологічній моделі. Нижче наведений *рис. 2.3. «Даталогічна модель даних»*, що відображає взаємозв’язки між таблицями бази даних Персонажі ДжоДжо.



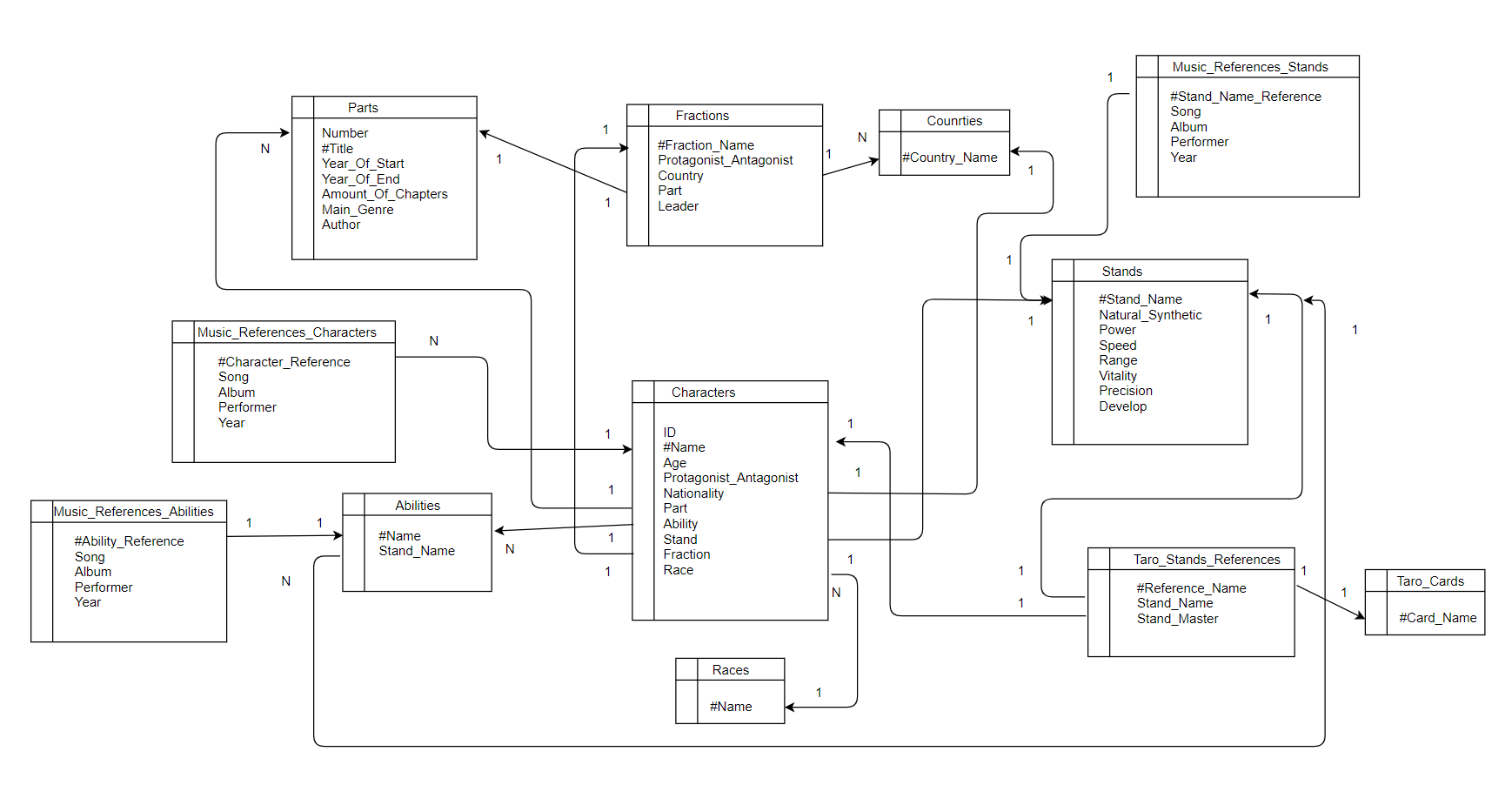
Розглянемо табличні дані та атрибути таблиць, що використовуються в курсовій работі (табл 2.1.):

*Таблиця 2.1. Дані та атрибути таблиць*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Сутність** | **Назва атрибуту** | **Ключ: PK, FK** | **Обов’язкове**  **значення** | **Тип даних** |
| **Таблиця Characters** | | | | |
| ID | Ідентифікатор персонажа |  | Так | Текстовий |
| Name | Ім’я персонажа | PK | Так | Текстовий |
| Age | Вік персонажа |  | Так | Цілий |
| Protagonist\_Antagonist | Протагоніст чи антагоніст |  | Так | Текстовий |
| Nationality | Національність персонажа (походження) | FK | Ні | Текстовий |
| Part | Частина історії | FK | Так | Цілий |
| Ability | Здібності персонажа | FK | Так | Текстовий |
| Stand | Станд персонажа | FK | Ні | Текстовий |
| Fraction | Фракція, до якої належить персонаж | FK | Ні | Текстовий |
| Race | Раса персонажа | FK | Так | Текстовий |
| **Таблиця Stands** | | | | |
| Stand\_Name | Ім’я станду | РК | Так | Текстовий |
| Natural\_Synthetic | Походження станду |  | Так | Текстовий |
| Power | Сила станду |  | Так | Текстовий |
| Speed | Швидкість станду |  | Так | Текстовий |
| Range | Дальність станду |  | Так | Текстовий |
| Vitality | Витривалість станду |  | Так | Текстовий |
| Precision | Точність станду |  | Так | Текстовий |
| Develop | Потенціал розвитку станду |  | Так | Текстовий |
| **Таблиця Fractions** | | | | |
| Fraction\_Name | Назва фракції | PK | Так | Текстовий |
| Protagonist\_Antagonist | На чиєму боці фракція |  | Так | Текстовий |
| Country | Країна, в якій фракція має вплив | FK | Так | Текстовий |
| Part | Частина історії | FK | Так | Цілий |
| Leader | Керівник |  | Ні | Текстовий |
| **Таблиця Abilities** | | | | |
| Name | Назва здібності | РК | Так | Текстовий |
| Stand\_Name | Ім’я станду, якому властива здібність | FK | Ні | Текстовий |
| **Таблиця Parts** | | | | |
| Number | Номер частини |  | Так | Цілий |
| Title | Назва частини | РК | Так | Текстовий |
| Year\_Of\_Start | Дата виходу першої глави частини |  | Так | Цілий |
| Year\_Of\_End | Дата виходу останньої глави частини |  | Так | Цілий |
| Amount\_Of\_Chapters | Кількість глав |  | Так | Цілий |
| Main\_Genre | Основний жанр написання |  | Так | Текстовий |
| Author | Автор |  | Так | Текстовий |
| **Таблиця Music\_References\_Characters** | | | | |
| Character\_Reference | Назва посилання, що походить від ім’я персонажа | PK, FK | Так | Текстовий |
| Song | Назва пісні |  | Ні | Текстовий |
| Album | Назва альбому |  | Ні | Текстовий |
| Performer | Ім’я (назва гурту) виконавця |  | Так | Текстовий |
| Year | Дата виходу пісні (виконавця) |  | Так | Цілий |
| **Таблиця Music\_References\_Stands** | | | | |
| Stand\_Name\_Reference | Назва посилання, що походить від ім’я станда | PK, FK | Так | Текстовий |
| Song | Назва пісні |  | Ні | Текстовий |
| Album | Назва альбому |  | Ні | Текстовий |
| Performer | Ім’я (назва гурту) виконавця |  | Так | Текстовий |
| Year | Дата виходу пісні (виконавця) |  | Так | Цілий |
| **Таблиця Music\_References\_Abilities** | | | | |
| Ability\_Reference | Назва посилання, що походить від ім’я здібності | PK, FK | Так | Текстовий |
| Song | Назва пісні |  | Ні | Текстовий |
| Album | Назва альбому |  | Ні | Текстовий |
| Performer | Ім’я (назва гурту) виконавця |  | Так | Текстовий |
| Year | Дата виходу пісні (виконавця) |  | Так | Цілий |
| **Таблиця Taro\_Stands\_References** | | | | |
| Reference\_Name | Назва посилання, що походить від назви карти Таро | PK,FK | Так | Текстовий |
| Stand\_Name | Ім’я станда, що посилається на карту Таро | FK | Так | Текстовий |
| Stand\_Master | Ім’я власника цього станда | FK | Так | Текстовий |
| **Таблиця Taro\_Cards** | | | | |
| Card\_Name | Назва карти Таро | PK | Так | Текстовий |
| **Таблиця Races** | | | | |
| Name | Назва раси | PK | Так | Текстовий |
| **Таблиця Counrties** | | | | |
| Country\_Name | Назва країни | PK | Так | Текстовий |

## **2.4. Фізична модель БД**

*Фізичне проектування*– це проектування БД у конкретній СУБД.

 На цьому етапі моделювання модель сутностей і зв’язків перетворюється на схему БД. Визначаються зв’язки, комунікації. За допомогою візуалізації можна зрозуміти логіку програми, які запити будуть використовуватися. Також важливо надавати інформативну назву полям таблиці для зрозумілості та зручного використання надалі. Завершена фізична модель даних включатиме всі умовності та правила БД, необхідні для створення відношень між таблицями чи для досягнення мети продуктивності.

*Рис. 2.4. Фізична модель даних.*

# РОЗДІЛ 3. Опис реалізації та склад проекту БД

## **3.1. Опис реалізації**

Перш за все, було створено з’єднання з локальною базою даних за допомогою наведених команд :

String url = "jdbc:sqlite:JoJosCharacters.db";

conn = DriverManager.getConnection(url);



*Рис. 3.1. Вигляд вікна головного меню*

За допомогою головного меню можна переміщуватися по усіх таблицях бази даних. Кожна форма містить однаковий набір функцій управління таблицями БД, а саме :

- Add new – вставка елементу до таблиці;

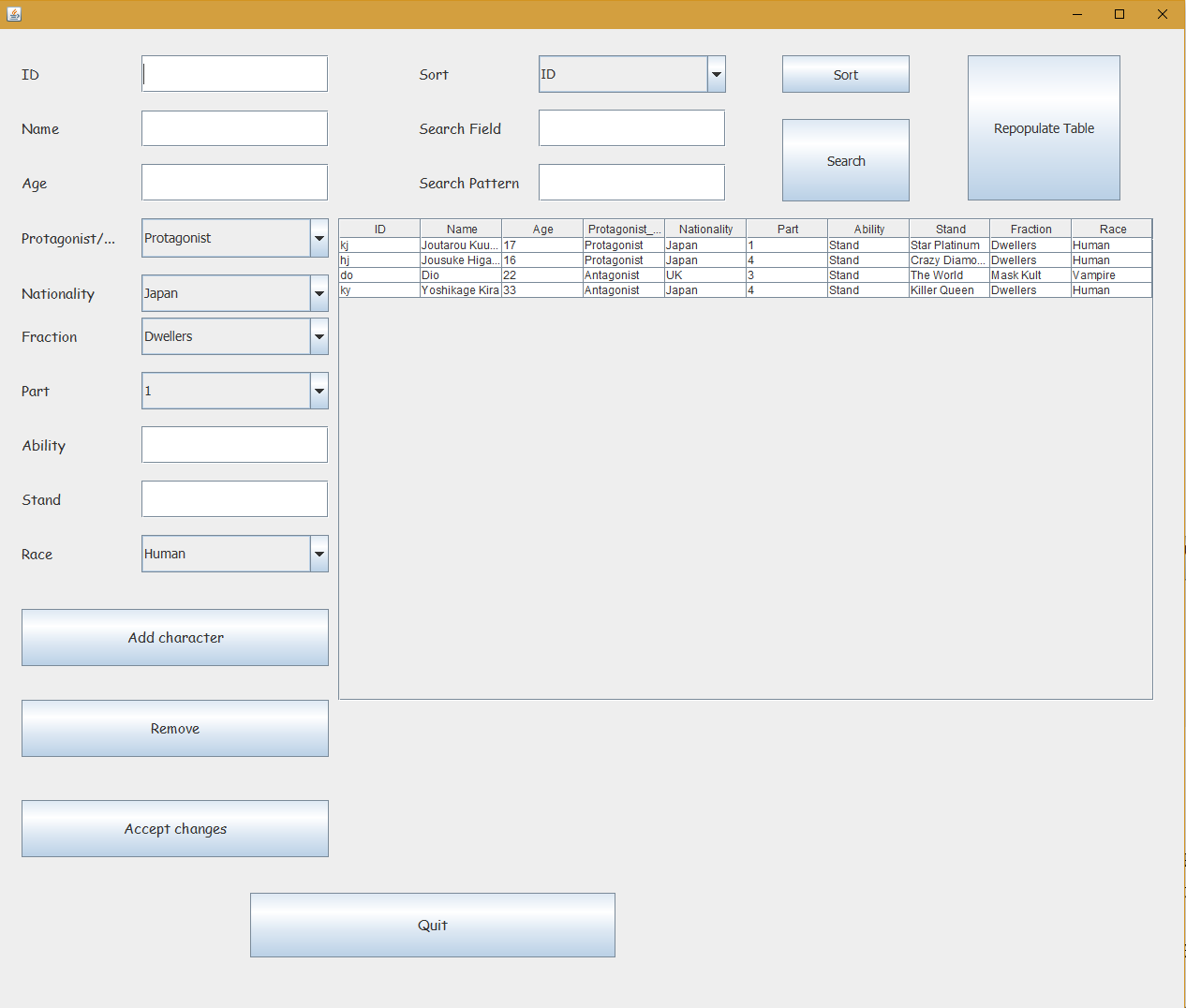
- Remove– видалення елементу з таблиці;

- Accept changes – оновлення елементу в таблиці;

- Repopulate table – вивести дані з БД у таблицю у початковому форматуванні (без сортування або пошуку);

- Sort – сортувати за вибраним полем;

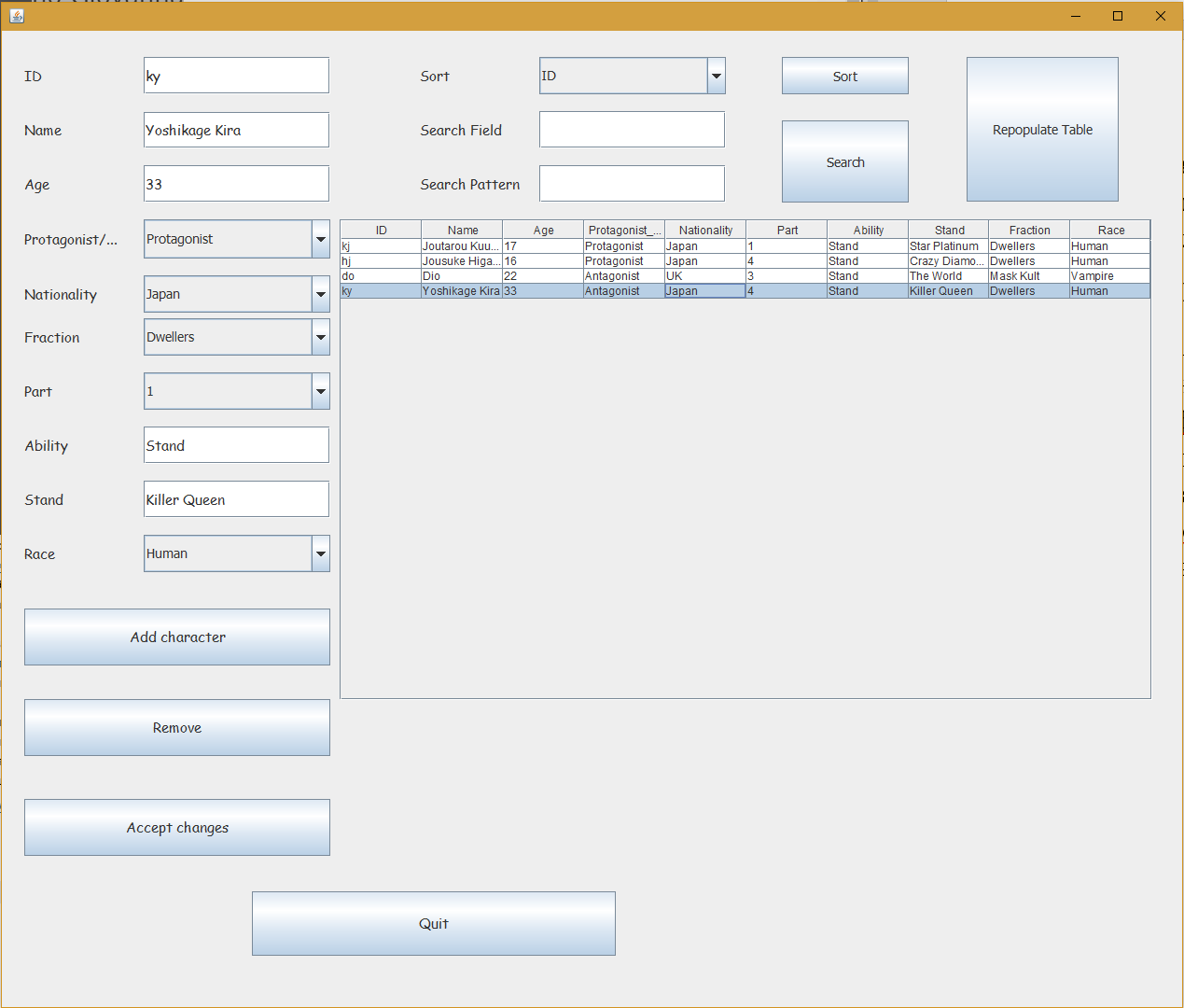
- Search – шукати за вибраним полем;

- Quit – закрити поточне вікно (якщо це вікно гол. меню, програма закриється повністю).

*Рис. 3.2. Вигляд вікна меню персонажів*

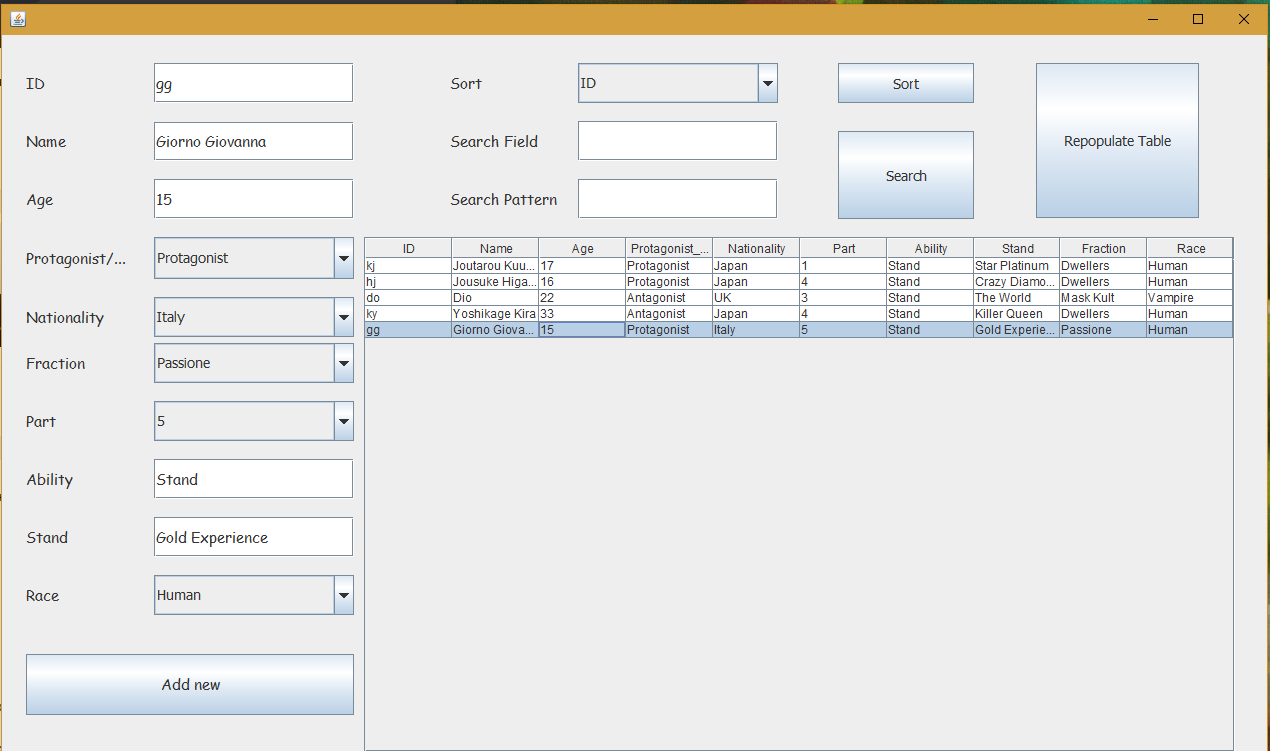
Для виводу неформатованої інформації, як показано на рис. 3.2, при створенні форми та спрацьовуванні кнопки Repopulate table використовується запит виду :

"SELECT \* FROM [ім’я таблиці]"

Також цей запит додатково використовується після видалення, оновлення або додавання нового об’єкту до таблиці для виведення усіх неформатованих даних з БД у форму. При натисканні ЛКМ на рядок таблиці данні цього рядка переносяться у поля вводу задля можливості їх подальшого легшого змінення. Елементи випадаючих меню при цьому залишаються незмінними. Така функція зберігається для усіх форм.

*Рис. 3.3. Заповнення текстових полів при натисканні ЛКМ на рядок таблиці*

При введенні даних у поля, виборі опцій із меню та натисканні кнопки Add new виконується додавання нового об’єкта у БД та миттєве оновлення таблиці форми. При додаванні нового об’єкту використовується запит типу :

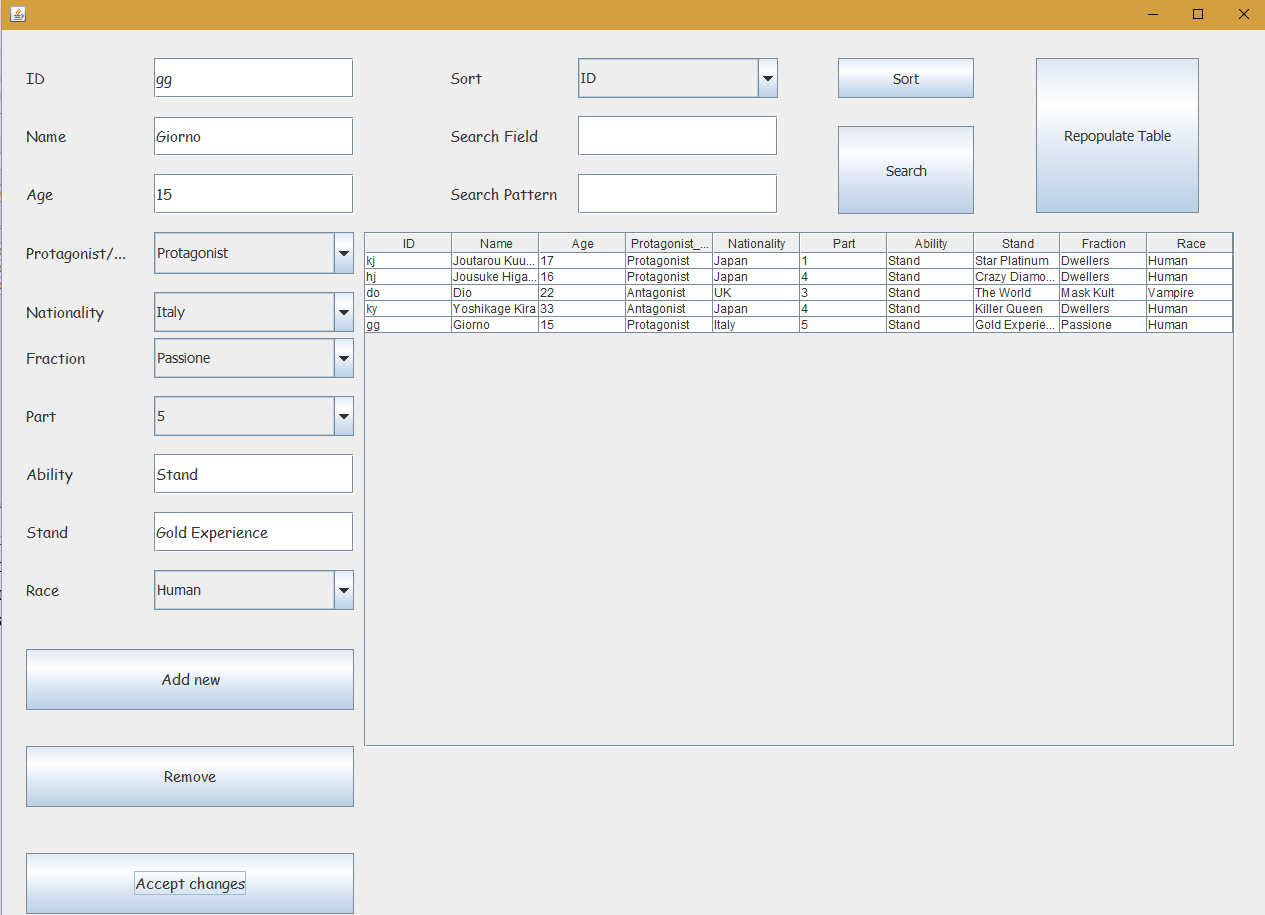
INSERT INTO Characters(ID, Name, Age, Protagonist\_Antagonist, Nationality, Part, Ability, Stand, Fraction, Race) VALUES(?,?,?,?,?,?,?,?,?,?)

*Рис. 3.4. Результат виконання роботи кнопки Add new*

Якщо виникає потреба оновити данні в таблиці, слід ввести нові данні у поля та/або вибрати нові пункти випадаючого меню та натиснути кнопку Accept changes. Головна умова – не змінювати поле ID, адже за цим полем ідентифікується строка таблиці. У такому випадку программа не знайде відповідності у БД та не зможе модифікувати рядок.

Результат – ім’я Giorno Giovanna було змінено на Giorno. Запит для модифікації данних має наступний вигляд :

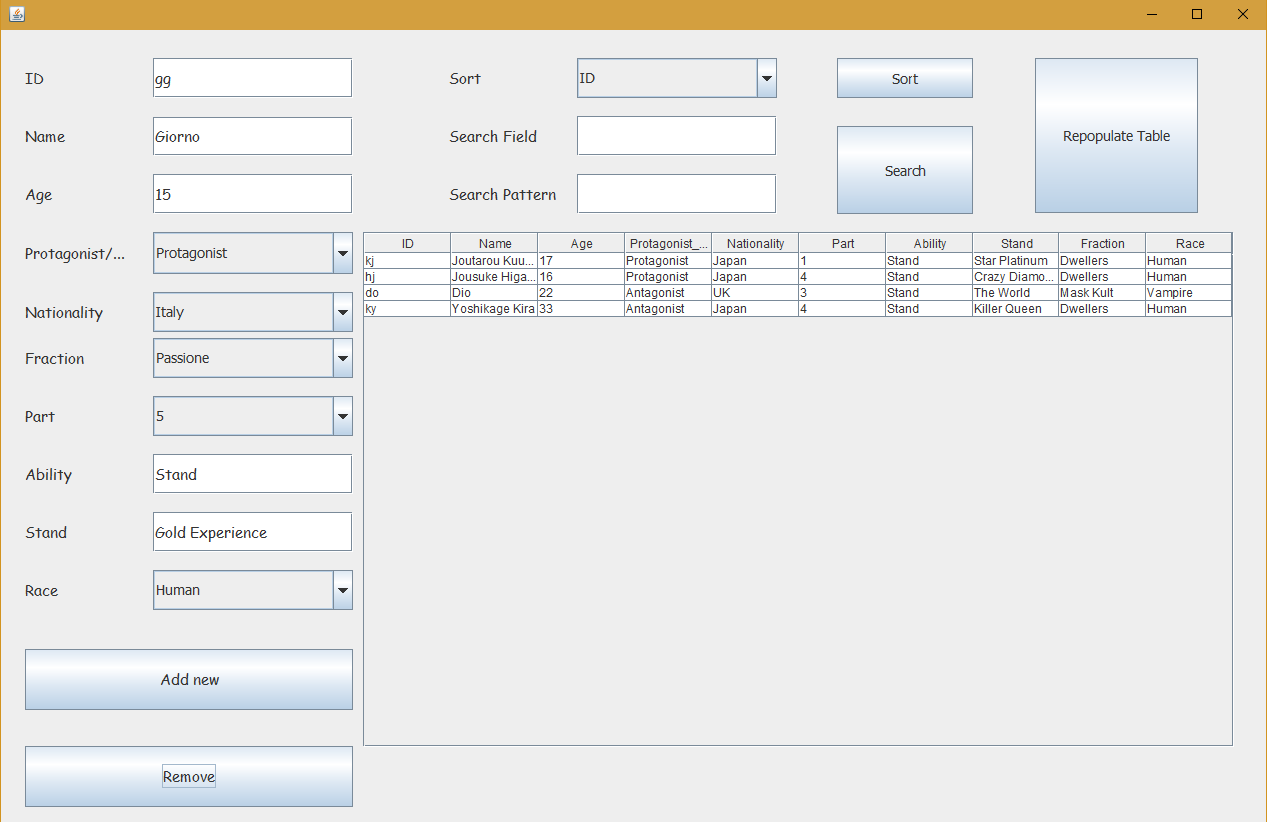
UPDATE Characters SET ID = ? , Name = ? , Age = ? , Protagonist\_Antagonist = ? , Nationality = ? ,"

+ "Part = ? ,Ability = ? ,Stand = ? ,Fraction = ? , Race = ?  + "WHERE ID = ?

*Рис. 3.5. Результат виконання роботи кнопки Accept changes*

При натисканні кнопки Remove видаляється рядок, але тільки той, що був вибраний. Запит для цієї функціїї має наступний вигляд :

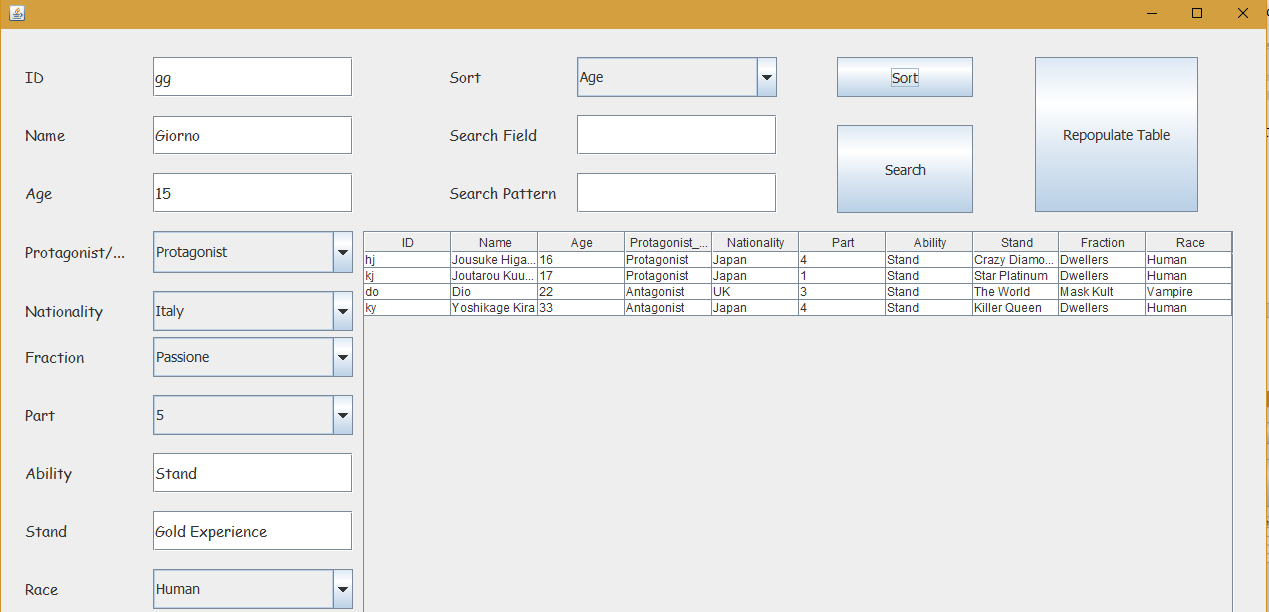
DELETE FROM Characters WHERE Name = ?

*Рис. 3.6. Результат виконання роботи кнопки Remove*

Над таблицею розташовані додаткові поля та кнопки обробки данних. Випадаюче меню Sort містить назви полів, за якими можна відсортувати данні в таблиці та вивести їх на головний екран. Сортування відбудеться після вибору поля сортування та натискання кнопки Sort. Запит має наступний вигляд :

"SELECT \* FROM Characters GROUP BY " + selectedColumn,

де selectedColumn – поле, за яким ми хочемо сортувати. Результат – на рис.3.7.

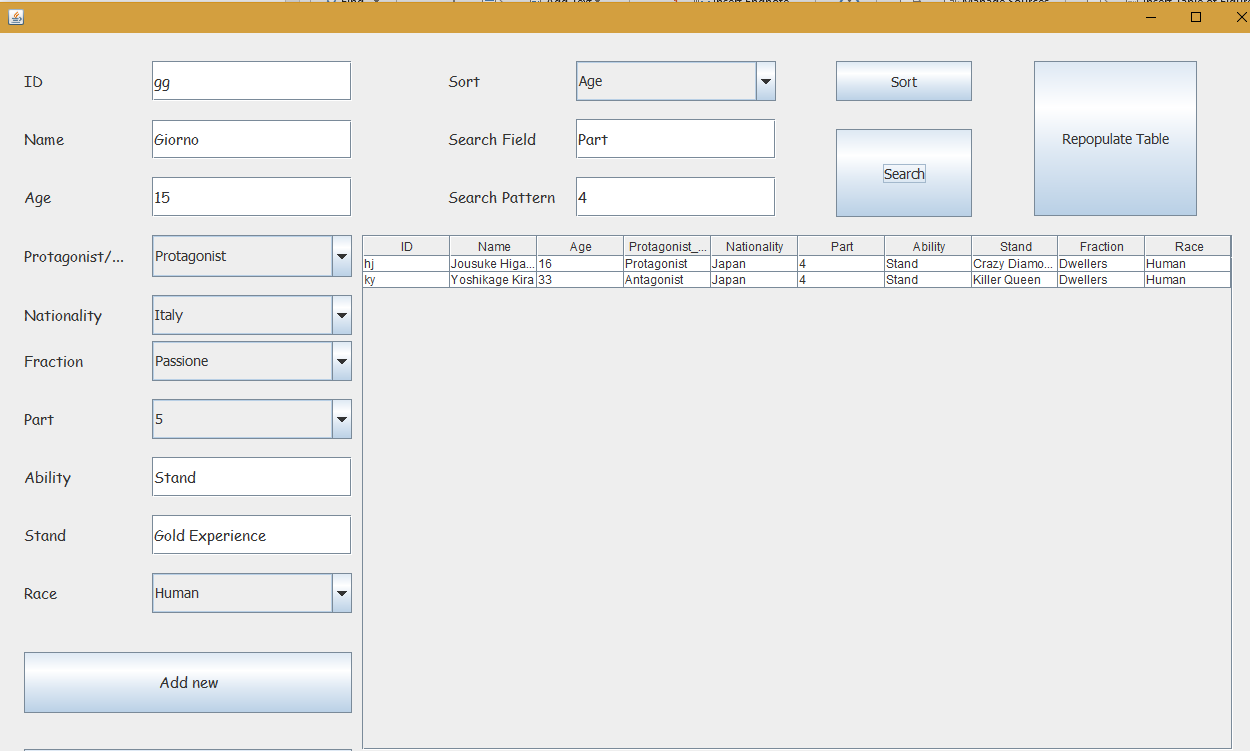
*Рис. 3.7. Результат виконання роботи кнопки Sort*

Нижче розташовані два поля : Search Field та Search Pattern. За допомогою цих текстових полів можна вибрати бажане поле, за

яким ми будемо шукати та шаблон для пошуку. Для прикладу, візьмемо поле та шаблон, що відображені на рис.3.8 та розглянемо результат роботи кнопки. Запит, що реалізує цю функцію має вигляд :

"SELECT \* FROM Characters WHERE " + searchedColumn + " LIKE ?"

де searchedColumn – поле, за яким відбувається пошук. Деталі – у додатку.

*Рис. 3.7. Результат виконання роботи кнопки Search*

Іноді користувач не зовсім розуміє, як правильно використовувати ту, чи іншу кнопку, як саме повинен працювати додаток. СУБД «Автосалон» зроблена з інтуїтивно – зрозумілим інтерфейсом, щоб кожен користувач міг логічно розуміти сенс програми та її використовувати.

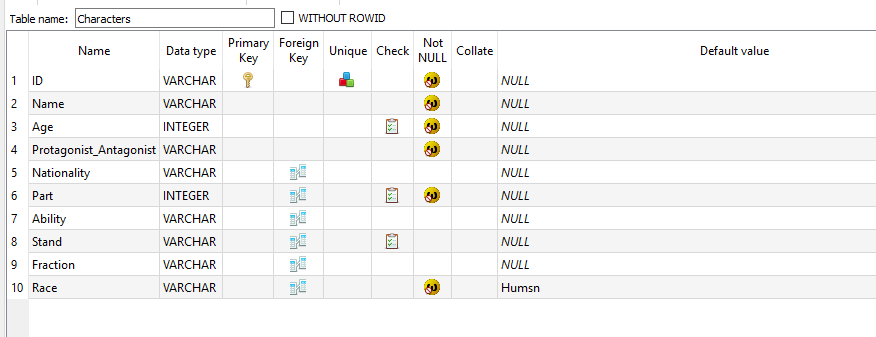
Жодних непередбачуваних ситуацій не може виникнути, тому що за допомогою блоку коду:

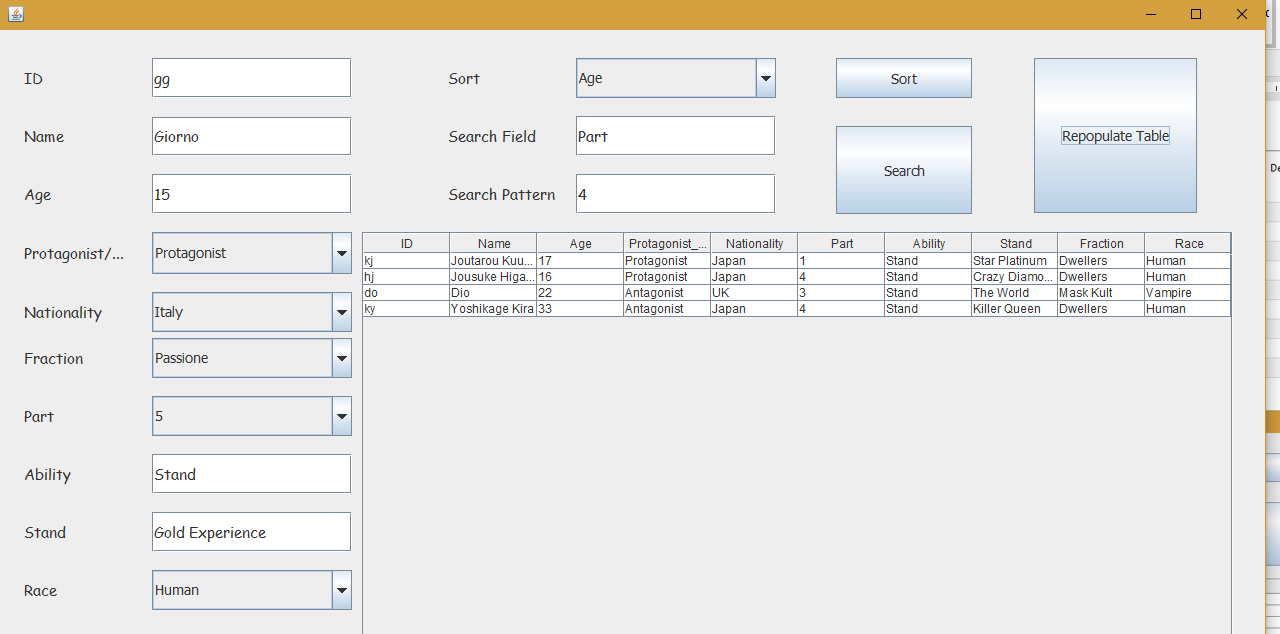
try{}

catch(){}

передбачаються всі некоректні дії користувача, після цього на екрані з’являється повідомлення з текстом інструкції, щодо щойно скоєної помилки чи неправильного користування додатком.

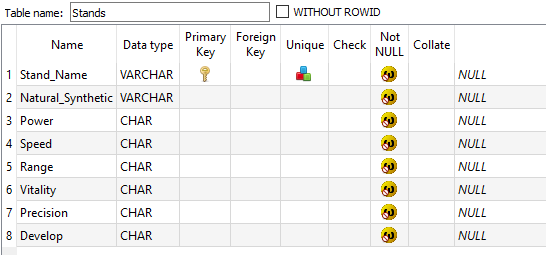
## **3.2. Склад БД**

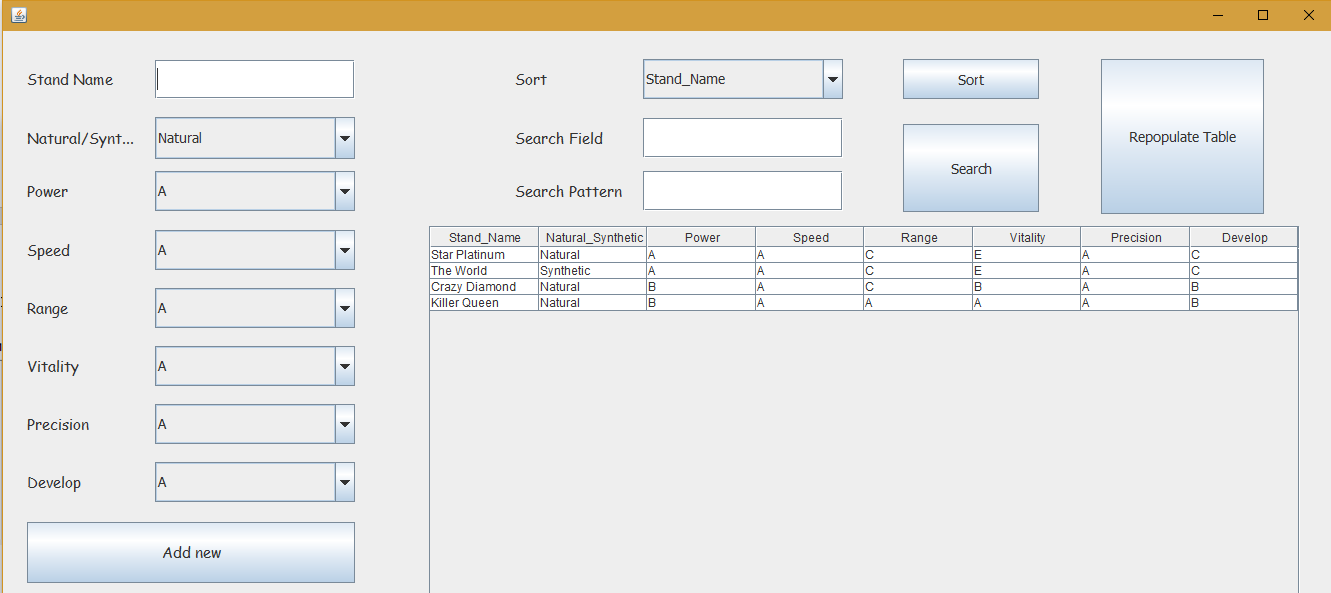
**Таблиця** **Characters**

*******Рис 3.8. Таблиця Characters*

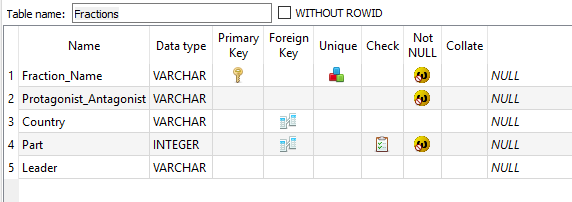
*Рис 3.8. Данні таблиці Characters*

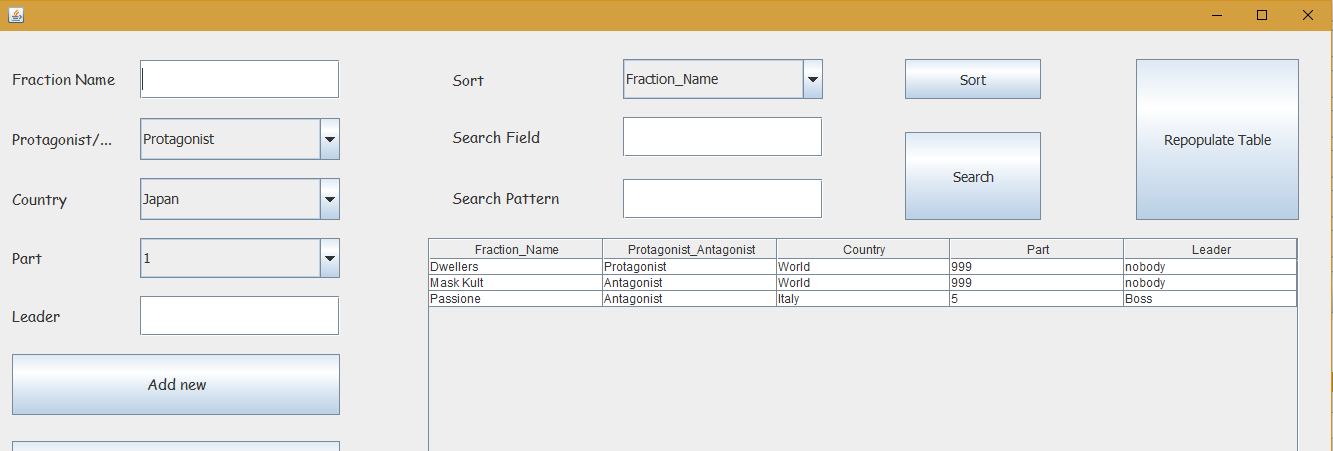
**Таблиця Stands**



*Рис 3.9. Таблиця Stands*

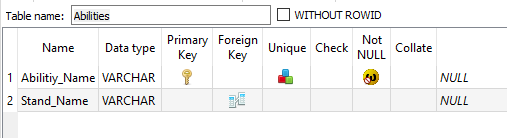
*Рис 3.10. Данні таблиці Stands*

**Таблиця Fractions**

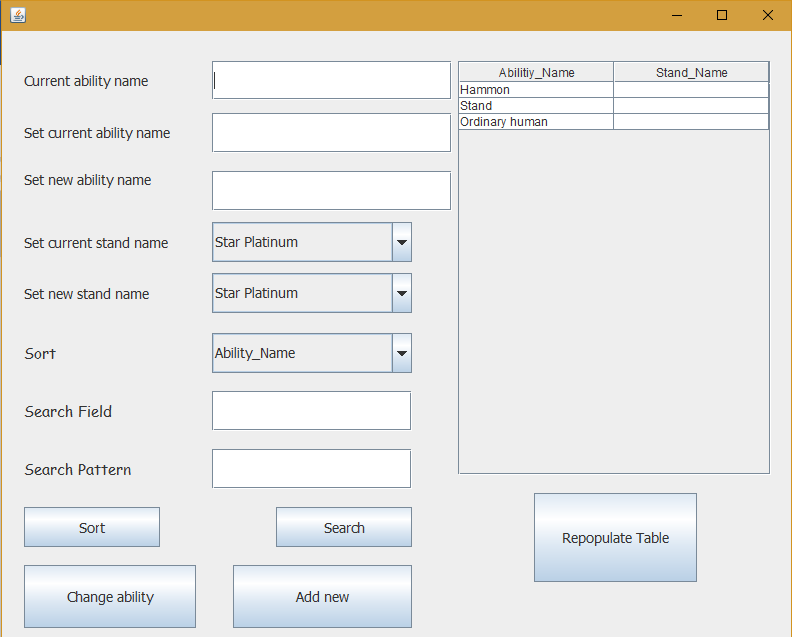
*Рис 3.11. Таблиця Fractions*

*Рис 3.12. Данні таблиці Fractions*

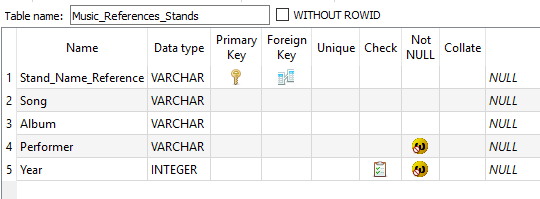
**Таблиця Abilities**



*Рис 3.13. Таблиця Abilities*

*Рис 3.14. Данні таблиці Abilities*

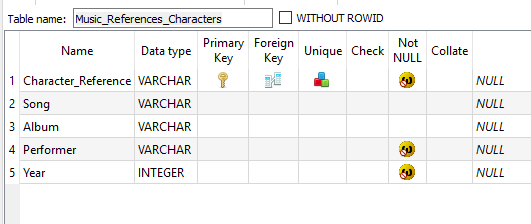
**Таблиця Music\_References\_Stands**

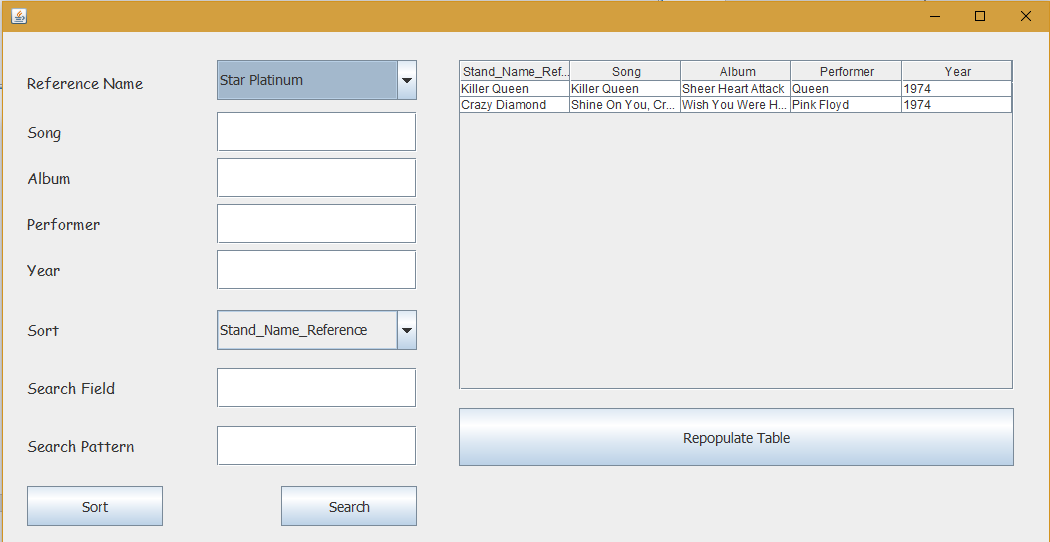


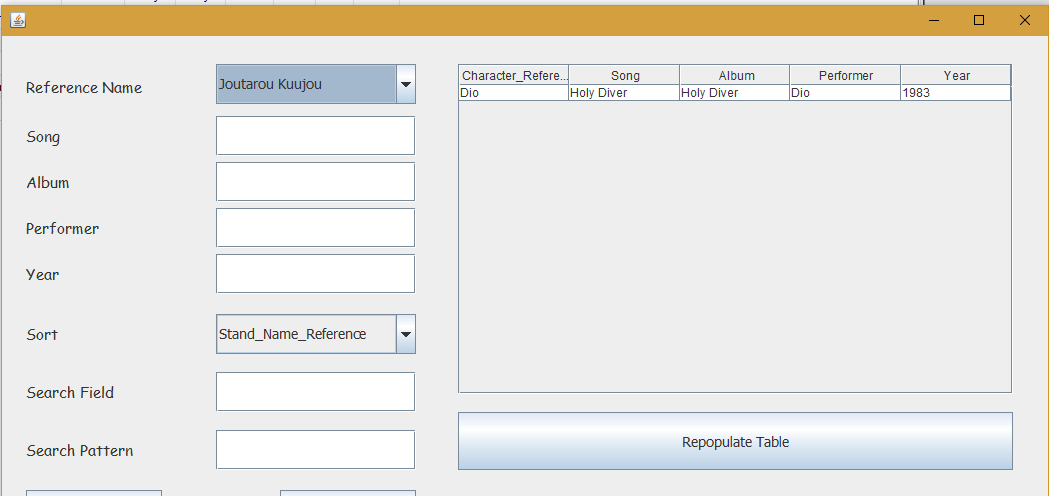
*Рис 3.15. Таблиця Music\_References\_Stands*

*Рис 3.16. Данні таблиці Music\_References\_Stands*

**Таблиця Music\_References\_Characters**

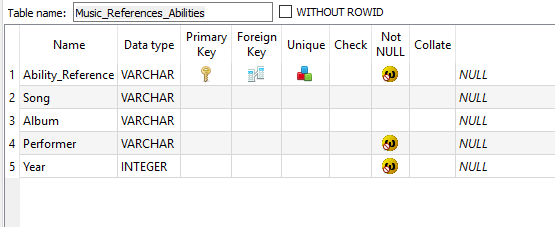


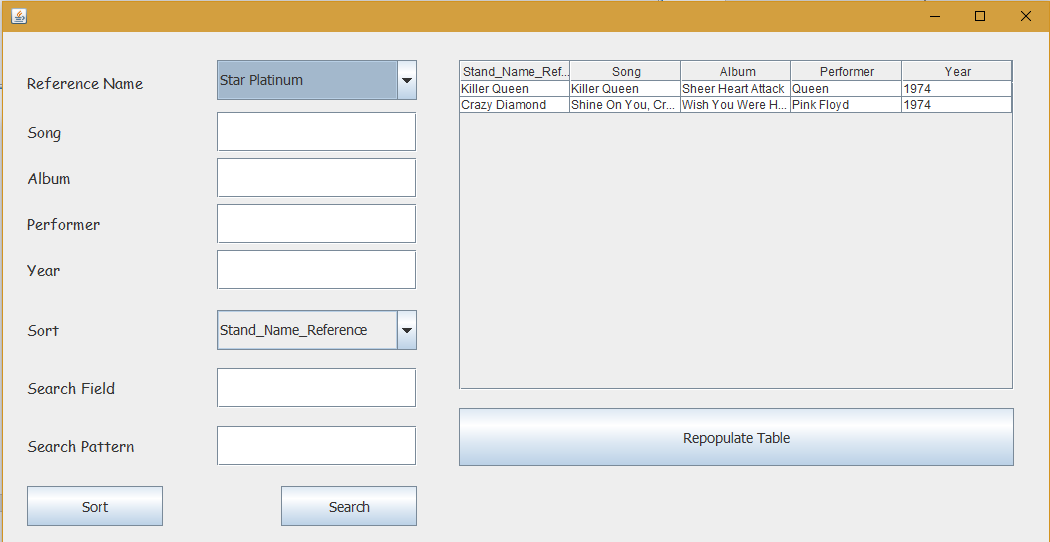
*Рис 3.17. Таблиця Music\_References\_Characters *

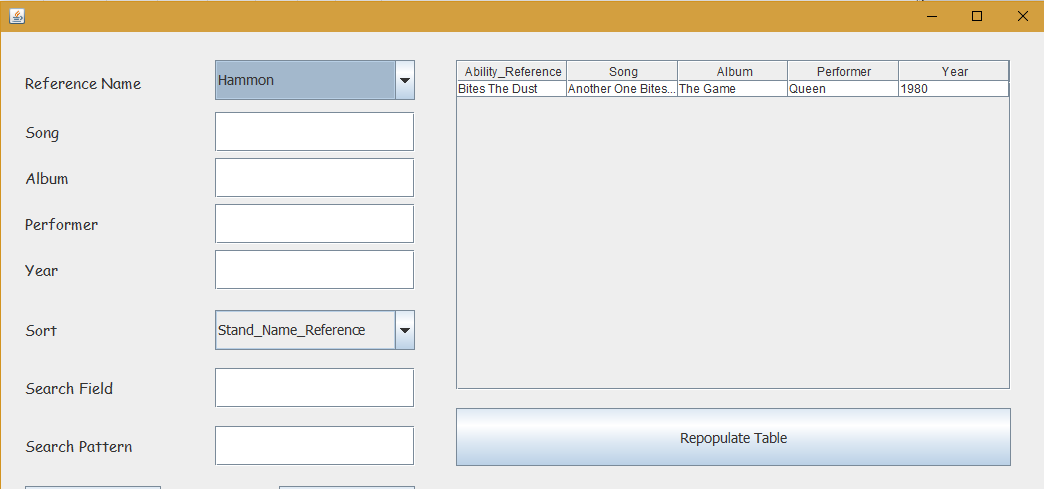


*Рис 3.18. Данні таблиці Music\_References\_Characters*

**Таблиця Music\_References\_Abilities**

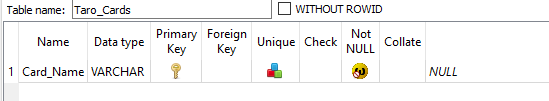


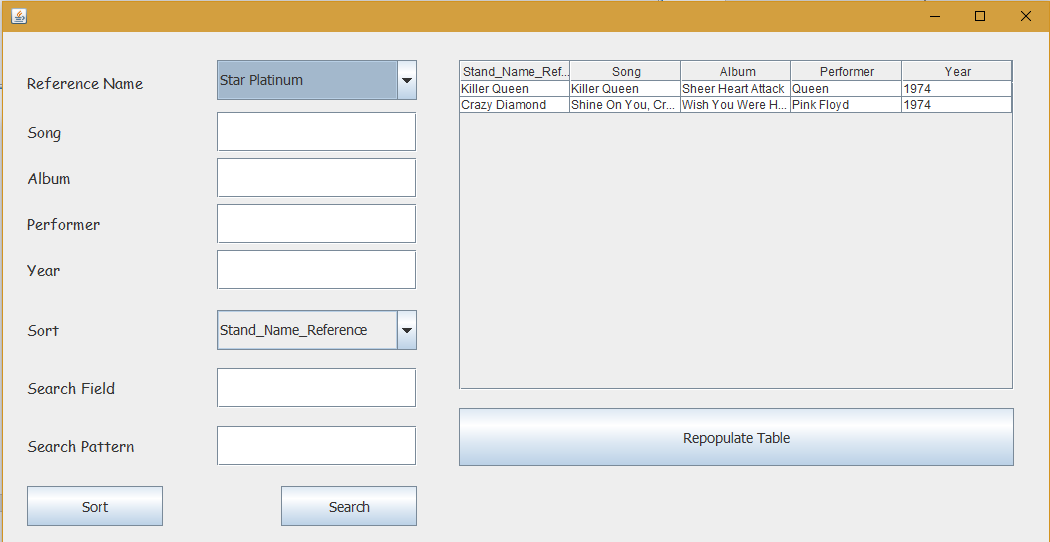
*Рис 3.19. Таблиця Music\_References\_Abilities*



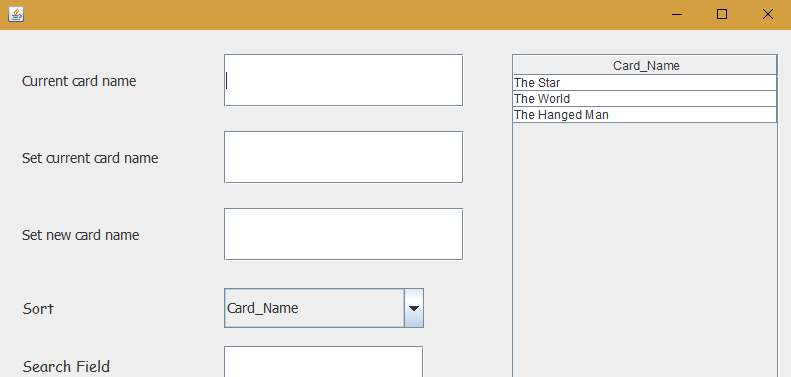
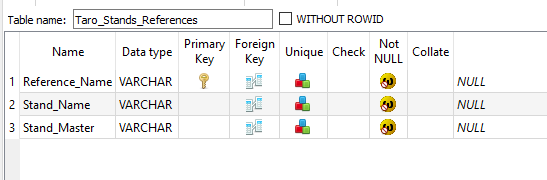
*Рис 3.20. Данні таблиці Music\_References\_Abilities*

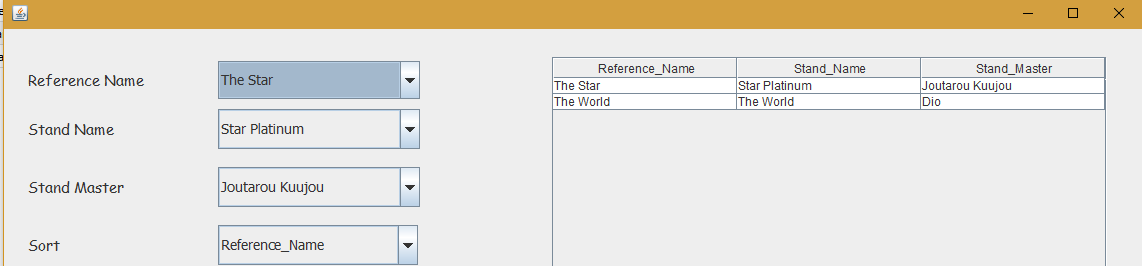
**Таблиця Taro\_Cards**



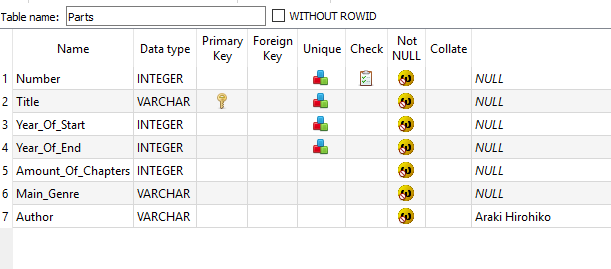
*Рис 3.21. Таблиця Taro\_Cards*

*Рис 3.22. Данні таблиці Taro\_Cards*

 **Таблиця Taro\_Stands\_References**

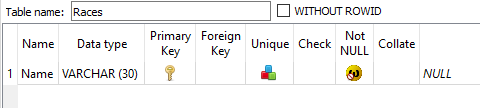
*Рис 3.23. Таблиця Taro\_Stands\_References*

*Рис 3.24. Данні таблиці Taro\_Stands\_References*

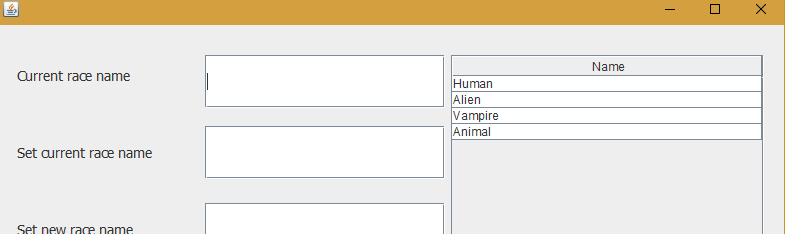
**Таблиця Parts**

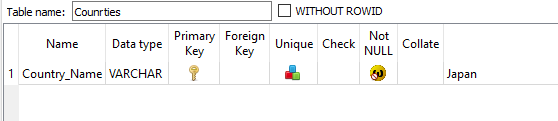
*Рис 3.25. Таблиця Parts*

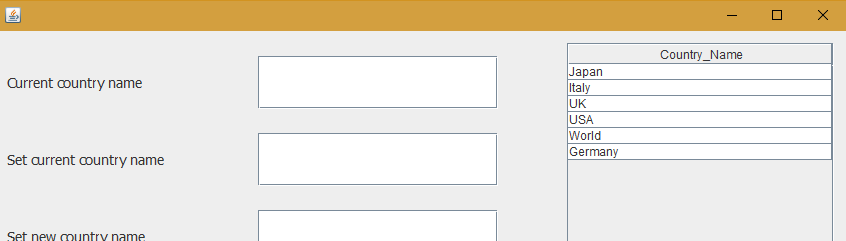
*Рис 3.26. Данні таблиці Parts*

**Таблиця Races**

*Рис 3.27. Таблиця Races*

*Рис 3.28. Данні таблиці Races*

**Таблиця Counrties**

*Рис 3.29. Таблиця Counrtiesі*

*Рис 3.30. Данні таблиці Counrties*

# Висновки

В результаті виконання курсової роботи був розроблений застосунок СУБД "Персонажі ДжоДжо" з для використання будь-яким дослідником всесвіту цього твору. СУБД дає змогу вивчити історію та особливості персонажів та їх оточення не порушуючи логічні зв’язки між ними. База даних являє собою реляційну базу даних.

Дана БД містить інформацію про персонажів, здібності, станди, фракції, музичні посилання, посилання на карти Таро, списки частин твору, рас та країн.

Застосунок, який є результатом виконання курсової роботи виконує всі поставлені перед ним завдання.

Були пройдені такі етапи створення застосунку:

* Концепція;
* Створення інтерфейсу;
* Створення функціоналу;
* Тестування.

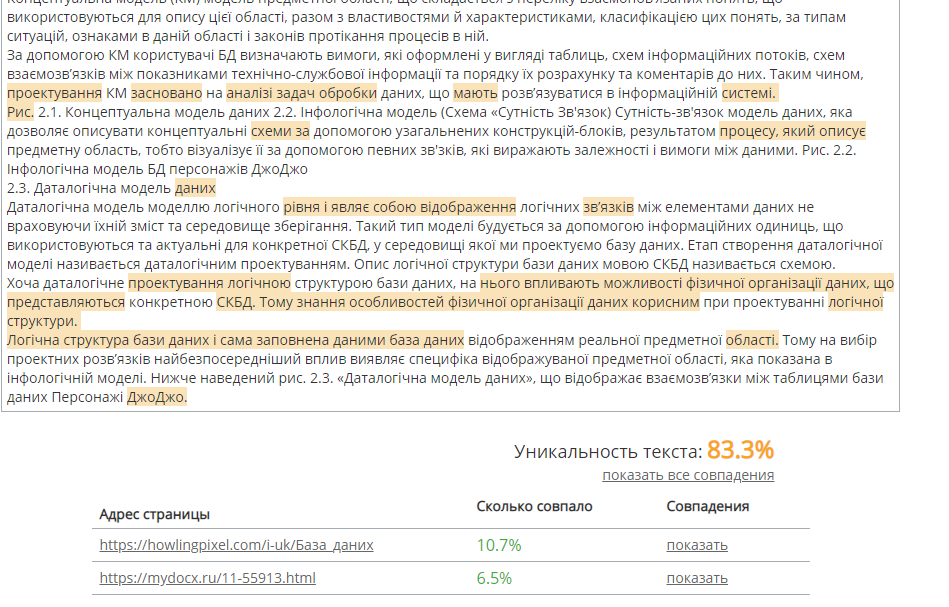
Система була протестована на предмет різноманітних виключень та помилок. Інтерфейс зрозумілий будь-якому користувачу та зручний у використанні.

Під час роботи над застосунком були вдосконалені вміння у мові Java та набуто багато нових навичок в роботі з Swing та базами даних.

# Список використаних джерел

1. Батрак Ю.А., Донченко М.В., Журавська І.М., Фісун М.Т. Методичні вказівки до оформлення звітної текстової документації та кваліфікаційних робіт з дисциплін, закріплених за факультетом комп'ютерних наук. – Миколаїв: видавн. ЧДУ ім. Петра Могили, 2009. – 42 с.
2. Дейт К. Дж. Введение в системы баз данных. - Киев\*Москва: Диалектика, 1998. -787 с.
3. Teorey, T.; Lightstone, S. and Nadeau, T. Database Modeling & Design: Logical Design, 4th edition, Morgan Kaufmann Press, 2005. ISBN 0-12-685352-5, 448 c.
4. Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, 1097 c.
5. Блог для програмістів – Підключення до БД – режим доступу: <https://habr.com/ru/post/326614/>
6. Java Swing | JTable режим доступу: <https://www.geeksforgeeks.org/java-swing-jtable/>
7. Stack Overflow - Where Developers Learn, Share, & Build Careers – режим доступу: https://stackoverflow.com/
8. Java. Работа с jdbc – режим доступу: <https://www.youtube.com/watch?v=BJ2SCZEkKoQ>

# Додаток А. Перевірка на оригінальність



*Рис. 1. Перевірка на унікальність*

# Додаток Б. Опис програмного коду

public Connection connect() {

String url = "jdbc:sqlite:JoJosCharacters.db"; Connection conn = null;

try {conn = DriverManager.getConnection(url);

System.out.println("Connection success");

} catch (SQLException e) { System.out.println("Connection failure"); }

return conn;}

private void populateTableMethod() {String populateTableSql = "SELECT \* FROM Characters";

try {prepparedStatObj = connectionObj.prepareStatement(populateTableSql);

resultSetObj = prepparedStatObj.executeQuery();

charactersTable.setModel(DbUtils.resultSetToTableModel(resultSetObj)); }

catch (SQLException ex)

{ Logger.getLogger(CharactersForm.class.getName()).log(Level.SEVERE, null, ex); }}

private void selectNationality()

{try {String sql = "SELECT \* FROM Counrties";

prepparedStatObj = connectionObj.prepareStatement(sql);

resultSetObj = prepparedStatObj.executeQuery();

while (resultSetObj.next())

{String raceName = resultSetObj.getString("Country\_Name").toString();

nationalityComboBox.addItem(raceName); }}

catch (SQLException e) { System.out.println("Cannot fill the combobox"); }}

private void selectPart()

{try {String sql = "SELECT \* FROM Parts";

prepparedStatObj = connectionObj.prepareStatement(sql);

resultSetObj = prepparedStatObj.executeQuery();

while (resultSetObj.next())

{int partNumber = resultSetObj.getInt(1);

partComboBox.addItem(partNumber); }}

catch (SQLException e)

{ System.out.println("Cannot fill the combobox");}}

private void selectFractions()

{try {String sql = "SELECT \* FROM Fractions"; prepparedStatObj = connectionObj.prepareStatement(sql);

resultSetObj = prepparedStatObj.executeQuery();

while (resultSetObj.next())

{String partNumber = resultSetObj.getString("Fraction\_Name").toString();

fractionsComboBox.addItem(partNumber); }}

catch (SQLException e)

{ System.out.println("Cannot fill the combobox");

private void selectRace()

{try {String sql = "SELECT \* FROM Races"; prepparedStatObj = connectionObj.prepareStatement(sql);

resultSetObj = prepparedStatObj.executeQuery();

while (resultSetObj.next())

{String partNumber = resultSetObj.getString("Name").toString();

racesComboBox.addItem(partNumber); }}

catch (SQLException e)

{ System.out.println("Cannot fill the combobox");

private void addCharacterActionPerformed(java.awt.event.ActionEvent evt) {

String characterId = setIdTextField.getText();

String characterName = setNameTextField.getText();

int characterAge = Integer.parseInt(setAgeTextField.getText());

String characterProtagAntag = protagAntagComboBox.getSelectedItem().toString();

String characterNationality = nationalityComboBox.getSelectedItem().toString();

int characterPart = Integer.parseInt(partComboBox.getSelectedItem().toString());

String characterAbility = abilityTextField.getText();

String characterStand = standTextField.getText();

String characterFraction = fractionsComboBox.getSelectedItem().toString();

String characterRace = racesComboBox.getSelectedItem().toString();

String sqlInsertNewCharacter = "INSERT INTO Characters(ID, Name, Age, Protagonist\_Antagonist, Nationality, Part, Ability, Stand, Fraction, Race) VALUES(?,?,?,?,?,?,?,?,?,?)";

try {PreparedStatement pstmt = connectionObj.prepareStatement(sqlInsertNewCharacter);

pstmt.setString(1, characterId);

pstmt.setString(2, characterName);

pstmt.setInt(3, characterAge);

pstmt.setString(4, characterProtagAntag);

pstmt.setString(5, characterNationality);

pstmt.setInt(6, characterPart);

pstmt.setString(7, characterAbility);

pstmt.setString(8, characterStand);

pstmt.setString(9, characterFraction);

pstmt.setString(10, characterRace);

pstmt.executeUpdate();

populateTableMethod();}

catch (SQLException e) { System.out.println("addin went wrong"); }}

private void acceptChangesButtonActionPerformed(java.awt.event.ActionEvent evt) { String characterId = setIdTextField.getText(); String characterName = setNameTextField.getText();

int characterAge = Integer.parseInt(setAgeTextField.getText());

String characterProtagAntag = protagAntagComboBox.getSelectedItem().toString();

String characterNationality = nationalityComboBox.getSelectedItem().toString();

int characterPart = Integer.parseInt(partComboBox.getSelectedItem().toString());

String characterAbility = abilityTextField.getText();

String characterStand = standTextField.getText();

String characterFraction = fractionsComboBox.getSelectedItem().toString();

String characterRace = racesComboBox.getSelectedItem().toString();

String sql = "UPDATE Characters SET ID = ? , Name = ? , Age = ? , Protagonist\_Antagonist = ? , Nationality = ? ,"

+ "Part = ? ,Ability = ? ,Stand = ? ,Fraction = ? , Race = ? "

+ "WHERE ID = ?";

try {prepparedStatObj = connectionObj.prepareStatement(sql);

prepparedStatObj.setString(1, characterId);

prepparedStatObj.setString(2, characterName);

prepparedStatObj.setInt(3, characterAge);

prepparedStatObj.setString(4, characterProtagAntag);

prepparedStatObj.setString(5, characterNationality);

prepparedStatObj.setInt(6, characterPart);

prepparedStatObj.setString(7, characterAbility);

prepparedStatObj.setString(8, characterStand);

prepparedStatObj.setString(9, characterFraction);

prepparedStatObj.setString(10, characterRace);

prepparedStatObj.setString(11, characterId);

prepparedStatObj.executeUpdate();

populateTableMethod(); }

catch (SQLException e)

{ System.out.println("updating went wrong"); }}

private void quitShipManagementFormActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose(); }

private void removeCharacterButtonActionPerformed(java.awt.event.ActionEvent evt) {

String sql = "DELETE FROM Characters WHERE Name = ?";

String selectedRow = charactersTable.getModel().getValueAt(charactersTable.getSelectedRow(), 1).toString();

try {prepparedStatObj = connectionObj.prepareStatement(sql);

prepparedStatObj.setString(1, selectedRow);

prepparedStatObj.executeUpdate();

populateTableMethod();}

catch (SQLException e) { System.out.println(e.getMessage()); }}

private void charactersTableMouseClicked(java.awt.event.MouseEvent evt) {

try {int row = charactersTable.getSelectedRow();

String tableClick = (charactersTable.getModel().getValueAt(row, 1).toString()); String sql = "SELECT \* FROM Characters WHERE Name = '"+tableClick+"' ";

prepparedStatObj = connectionObj.prepareStatement(sql);

resultSetObj = prepparedStatObj.executeQuery();

if(resultSetObj.next())

{String characterId = resultSetObj.getString("ID");

String characterName = resultSetObj.getString("Name");

String characterAge = resultSetObj.getString("Age");

String characterAbility = resultSetObj.getString("Ability");

String characterStand = resultSetObj.getString("Stand");

setIdTextField.setText(characterId);

setNameTextField.setText(characterName);

setAgeTextField.setText(characterAge);

abilityTextField.setText(characterAbility);

standTextField.setText(characterStand); }}

catch (SQLException e) { System.out.println("table fill went wrong");} }

private void sortButtonActionPerformed(java.awt.event.ActionEvent evt) {

String selectedColumn = sortComboBox.getSelectedItem().toString();

String sqlSort = "SELECT \* FROM Characters GROUP BY " + selectedColumn;

try {prepparedStatObj = connectionObj.prepareStatement(sqlSort);

resultSetObj = prepparedStatObj.executeQuery();

charactersTable.setModel(DbUtils.resultSetToTableModel(resultSetObj)); }

catch (Exception e) {System.out.println("sorting went wrong"); } }

private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {

String searchedColumn = searchFieldTextField.getText();

String searchPattern = searchPatternTextField.getText();

String sqlSearch = "SELECT \* FROM Characters WHERE " + searchedColumn + " LIKE ?";

try {prepparedStatObj = connectionObj.prepareStatement(sqlSearch)

prepparedStatObj.setString(1, "%" + searchPattern + "%");

resultSetObj = prepparedStatObj.executeQuery();

charactersTable.setModel(DbUtils.resultSetToTableModel(resultSetObj));

} catch (Exception e) { System.out.println("seraching went wrong"); }