P2: Datenbanken (Olympia)

Abgabetermin: 24. April 2025

Punktzahl: 35 + 5* + 5 (Einhaltung der Coding Richtlinien)

In diesem Projekt sollen Sie eine Applikation vervollständigen, in der es darum geht, Daten zu den olympischen Spielen der Neuzeit (1896 bis 2022)[1] zu verarbeiten und verschiedene Informationen (wie z.B. die Ergebnisse eines Wettbewerbs) auszufiltern. Screenshots hierzu finden Sie am Ende des Dokuments.

Folgende Aufgaben sind in diesem Projekt zu bearbeiten

- Füllen von Datenbanktabellen mit Daten aus mehreren CSV-Dateien
- Generieren von Datenbankanfragen und Auswerten der Ergebnisse

Vorgaben

Das Eclipse Projekt in der Datei `P2-Vorgaben.zip` definiert bereits eine Grundstruktur, die Sie für Ihre Implementierung verwenden sollen. Neben einem `resources` Unterverzeichnis sind drei Packages mit insgesamt neun Klassen vorgegeben.

resources

In diesem Verzeichnis finden Sie folgende Dateien:

- die Library `sqlite-jdbc-3.46.1.3.jar`. Der Build Path ist im Projekt bereits entsprechend angepasst.
- drei CSV-Dateien mit den Daten für die von Ihnen zu füllenden Tabellen.
- die Datenbank `Olympia.db` mit 5 Tabellen, von denen zwei mit Daten zu den Ländern und der Medaillen Historie bereits gefüllt sind. Die restlichen drei sind von Ihnen in Aufgabe 1 zu füllen.
- die Text Datei `solution.txt`, welche exemplarische Lösungen zu den von Ihnen in Aufgabe 2 zu erstellenden Datenbankabfragen enthält.

[1] Modifiziert von https://www.kaggle.com/datasets/muhammadehsan02/126-years-of-historical-olympic-dataset (Stand: 20.09.2024)

src

Im Verzeichnis für die Quelldateien sind die folgenden vier Packages definiert:

data

In diesem Package sind die Klassen `AthleteEventResult` und `Result` definiert. Die Klasse `AthleteEventResult` repräsentiert einen Athleten und dessen Ergebnis (in Bezug auf eine Medaille) in einer Edition der Olympischen Spiele in einem einzelnen Event. Hierzu sind neben den Attributen ein Konstruktor und die Getter Methoden zu den Attributen definiert.
Die Klasse `Result` repräsentiert ein einzelnes Ergebnis mit der genauen Position eines Athleten aus einem bestimmten Land. Hierzu sind neben den Attributen und einem Konstruktor Getter Methoden zu den Attributen definiert. Außerdem ist die `compareTo` Methode überschrieben, welche dazu dient zwei Positionen zu vergleichen. Hierzu sind auch weitere Hilfsmethoden definiert.

db

Dieses Package beinhaltet alle Klassen, die auf die Datenbank zugreifen:

- Die Klasse `Database` bildet eine abstrakte Basisklasse für den Zugriff auf die Datenbank. Die Klasse hat hierzu das Attribut `connection` für die Verbindung zu der Datenbank, welches in der vorgegebenen `connect` Methode initialisiert und in den Unterklassen verwendet werden kann. Die `connect` Methode wird dabei im Konstruktor der Klasse aufgerufen. Die vorgegebene Methode `disconnect` schließt die Verbindung wieder. Um das Öffnen und Schließen der Datenbank-Verbindung müssen Sie sich hier nicht explizit kümmern, da dies in den Unterklassen `CreateDatabase, FillDatabase und FilterDatabase` sowie in der Klasse `GUI` (aus dem Package `gui`) bereits vorgegeben ist.
- Die Klasse `CreateDatabase` dient dazu, die Tabellen in der Datenbank (neu) zu erstellen.
- Mit Hilfe der Klasse `FillDatabase` werden die Tabellen in der Datenbank gefüllt.
- Die Klasse `FilterDatabase` beinhaltet mehrere Methoden, um Informationen aus der Datenbank zu filtern.

gui

Die einzige Klasse in diesem Package, `GUI`, stellt eine graphische Oberfläche zum Testen der von Ihnen zu entwickelnden Funktionalität zur Verfügung. Über die Buttons `Athleten, Ergebnisse und Medaillenspiegel` gelangen Sie zu verschiedenen Panels, in der Sie dann ggf. noch einige Parameter einstellen müssen. Sofern Sie eine sinnvolle Kombination eingegeben haben, werden verschiedene Methoden in anderen Klassen aufgerufen und die zugehörigen Informationen angezeigt.

io

Die einzige Klasse in diesem Package, `CSVReader`, dient dazu, die Daten aus den CSV-Dateien einzulesen. Mit der Methode `read` wird die CSV-Datei zeilenweise eingelesen. Die erste Zeile mit den Spaltenüberschriften wird übersprungen. Jede weitere Zeile wird in ein Array von Strings aufgeteilt. Die resultierenden String-Arrays werden in einem Listen-Objekt gesammelt und zurückgegeben. Diese Listen benötigen Sie später, um in Aufgabe 1 die Tabellen zu füllen. Sie dürfen davon ausgehen, dass die Länge der Felder jeweils der Anzahl der Spalten in den zugehörigen CSV-Dateien entspricht und fehlende Werte in durch leere Strings repräsentiert werden.

Aufgabenstellung

Zunächst sollen Sie die Datenbank vollständig füllen und anschließend verschiedene Informationen ausfiltern. Hierzu benötigen Sie u.a. Objekte der Klassen `Statement` und `ResultSet`. Sorgen Sie dafür, dass diese immer korrekt geschlossen werden. Falls möglich, verwenden Sie eine *Try-With-Resources* Anweisung.

Aufgabe 1: Datenbank füllen — 10 Punkte

Im ersten Schritt sollen Sie die Datenbank mit den Daten aus den CSV-Dateien füllen. In der Klasse `FillDatabase` ist das Attribut `reader` vorgegeben, mit welchem die entsprechenden CSV-Dateien gelesen werden. Die Klasse stellt bereits folgende Methoden zur Verfügung:

- Die Methode `fillTables` ruft die Methoden zum Füllen der einzelnen Tabellen auf. Um das Befüllen effizient zu gestalten, wird der Autocommit Status zu Beginn auf `false` gesetzt. Anschließend werden die ausgeführten Änderungen explizit comitted und die Prepared Statements am Ende (durch die Methode `closeStatements`) geschlossen.
- Die Methode `clearTable` löscht den Inhalt der als Parameter übergebenen Tabelle. Diese Methode wird jeweils vor dem Füllen jeder Tabelle aufgerufen, um gegebenenfalls alte Daten zu entfernen.
- Die Methode `closeStatements` schließt die als Attribute vorgegebenen Prepared Statements falls nötig/möglich.
- In der `main` Methode wird die Verbindung zur Datenbank geöffnet und alle Tabellen werden befüllt. Am Ende wird die Verbindung zur Datenbank wieder geschlossen.

Die von Ihnen zu füllenden Tabellen, welche bereits in der Klasse `createDatabase` erstellt wurden, haben folgende Schemata:

In der Tabelle ***athletes*** sind die allgemeinen Informationen zu allen Athleten, die bei den olympischen Spielen teilgenommen haben, enthalten. Das Schema für die Tabelle *athletes* enthält die ID, den Namen, das Geschlecht, das Geburtsdatum, Größe und Gewicht sowie das Herkunftsland der Athleten. Anmerkung: Der hier angegebene Datentyp `DATE` existiert in SQLite nicht, wird aber verwendet um einen Text (String) einzutragen, der ein Datum repräsentiert.

athlete_id	name	sex	born	height	weight	country
INTEGER PRIMARY KEY NOT NULL	TEXT NOT NULL	TEXT NOT NULL	DATE	NUMERIC	INTEGER	TEXT NOT NULL

Die Tabelle ***event_ details*** enthält die Ergebnisse aller ausgetragenen Wettbewerbe. Ein einzelner Eintrag repräsentiert dabei die Teilnahme eines einzelnen Athleten in einem Wettbewerb (der sich aus einer Sportart und dem Namen des Events zusammensetzt) in einer Edition der Olympischen Spiele und dem zugehörigen Ergebnis (Position und ggf. Medaille). Das Schema für die Tabelle *event_ details* enthält die Fremdschlüssel `edition_id`, welcher auf den entsprechenden Primärschlüssel der Tabelle *game_ summary* verweist und `athlete_id`, welcher auf den entsprechenden Primärschlüssel der Tabelle *athletes* verweist. Außerdem beinhaltet das Schema die jeweilige Sportart (z.B. Athletics), das Event (z.B. 100m Sprint), die Position, die der Athlet erreicht hat sowie die Medaille (Gold,Silber,Bronze).

edition_id	athlete_id
INTEGER NOT NULL REFERENCES game_summary (edition_id)	INTEGER NOT NULL REFERENCES athletes (athlete_id)

sport	event	pos	medal
TEXT NOT NULL	TEXT NOT NULL	TEXT NOT NULL	TEXT

Die Tabelle ***game_ summary*** enthält Informationen zum jeweiligen Austragungsort der Spiele. Das Schema für die Tabelle *game_ summary* enthält die ID, die Edition (z.B. Sommerspiele), das Austragungsjahr und den Austragungsort, das Land in dem der Austragungsort liegt sowie eine Anmerkung, falls die Spiele nicht stattfanden (z.B wegen Krieg).

edition_id	edition	year	city	country	reason_not_held
INTEGER PRIMARY KEY NOT NULL	TEXT NOT NULL	TEXT NOT NULL	TEXT NOT NULL	TEXT NOT NULL	TEXT

Die im Folgenden beschriebenen Tabellen ***countries*** und ***medal_history*** sind bereits in der Datenbank vorhanden und müssen `nicht` von Ihnen gefüllt werden.

Die Tabelle *countries* enthält alle Länder. Das Schema enthält die ID sowie den Namen des Landes.

country_noc	country
TEXT PRIMARY KEY NOT NULL	TEXT NOT NULL

Die Tabelle *medal_history* enthält die Informationen dazu, wie viele Medaillen ein Land bei den einzelnen Spielen gewonnen hat. Das Schema der Tabelle *medal_history* enthält den Fremdschlüssel `edition_id`, welcher auf den entsprechenden Primärschlüssel der Tabelle *game_summary* verweist, das Land sowie die entsprechende Anzahl an Gold-, Silber- und Bronzemedaillen.

edition_id	country	gold	silver	bronze
INTEGER NOT NULL REFERENCES game_summary (edition_id)	TEXT NOT NULL	INTEGER NOT NULL	INTEGER NOT NULL	INTEGER NOT NULL

Um die Tabellen `athletes`, `event_details` und `game_summary` zu füllen, müssen Sie die Daten aus den CSV Dateien `athletes.csv`, `event_details.csv` und `game_summary.csv` verwenden. Die Methode `read` aus der Klasse `io.CSVReader` liefert Ihnen für jede der Dateien eine Liste von Feldern, welche jeweils die Spaltenwerte einer einzelnen Zeile aus der zugehörigen Datei beihalten. Schauen Sie sich jeweils die erste Zeile in den drei Dateien an, um die Spalteneinträge den Tabellenattributen zuzuordnen.
Ergänzen Sie die Klasse `FillDatabase` nun wie folgt:

- Ergänzen Sie die Methode `prepareStatements` so, dass die Attribute `pstmAthletes`, `pstmEventDetails` und `pstmGameSummary` mit Prepared Statements initialisiert werden, mit denen die Daten in die Tabellen `athletes`, `event_details` und `game_summary` eingefügt werden können.

- Durchlaufen Sie in der Methode `fillAthlethesTable` die Liste `athletes` und fügen Sie mit Hilfe des zugehörigen Prepared Statements eine entsprechende Zeile in die Tabelle *athletes* ein. Überprüfen Sie dabei, ob Werte für das Geburtsdatum, die Größe und das Gewicht vorhanden sind (falls nicht wird ein leerer String übergeben) und setzen sie die Einträge, für die es keine Werte gibt, explizit auf einen Nullwert (des entsprechenden Typs).

- Durchlaufen Sie in der Methode `fillEventDetailsTable` die Liste `eventDetails` und fügen Sie mit Hilfe des zugehörigen Prepared Statements eine entsprechende Zeile in die Tabelle *event_details* ein.

- Durchlaufen Sie in der Methode `fillGameSummaryTable` die Liste `gameSummary` und fügen Sie mit Hilfe des zugehörigen Prepared Statements eine entsprechende Zeile in die Tabelle *game_summary* ein. Überprüfen Sie dabei, ob für den Tabelleneintrag `reason_not_held` ein Wert vorhanden ist (falls nicht, wird ein leerer String übergeben) und setzen sie die Einträge, für die es keine Werte gibt, explizit auf einen Nullwert eines passenden Datentyps (aus der Klasse `java.sql.Types`).

Hinweis: Führen Sie die Klasse `FillDatabase` aus und überprüfen Sie anschließend mit einem Tool Ihrer Wahl (z.B dem `DB Browser for SQLite`), ob die Datenbank korrekt gefüllt wurde (mit 155.861 Einträgen in der Tabelle *athletes*, 314.408 Einträgen in der Tabelle *event_details* und mit 61 Einträgen in der Tabelle *game_summary*). Aufgrund der Datenmenge dauert das Einfügen ein paar Sekunden. Sobald die Daten fertig eingefügt wurden erscheint in der Konsole der Text "Successfully disconnected from database".

Aufgabe 2: Datenbank filtern $12 + 10 + 8 = 30$ Punkte

In dieser Aufgabe sollen Sie mit Hilfe von SQL-Statements verschiedene Informationen aus der Datenbank filtern, indem Sie mehrere Methoden in der Klasse `FilterDatabase` entsprechend vervollständigen. Nach jeder Aufgabe können Sie das Ergebnis in der GUI testen und mit den Screenshots (die Sie am Ende dieses Dokuments finden) vergleichen. Da die Aufgaben teilweise aufeinander aufbauen, müssen Sie i.d.R aufeinanderfolgende Aufgaben nacheinander korrekt gelöst haben, um die nachfolgenden Aufgaben in der GUI testen zu können. Sie können aber auch jede Methode einzeln testen, indem Sie alle bis auf den zugehörigen Methodenaufruf in der `main`-Methode der Klasse auskommentieren. Die entsprechenden Ergebnisse finden Sie im `resources` Ordner in der Datei `solution.txt`.

Sie benötigen hier Objekte der Klassen `PreparedStatement`, `Statement` und `ResultSet`. Sorgen Sie dafür, dass `Statement` und `ResultSet` Objekte nach Verwendung stets geschlossen werden. Tritt eine `SQLException` auf, soll eine passende Fehlermeldung in der Konsole ausgegeben werden.

Aufgabe 2(a) Ergebnisse filtern **12 P**

Vervollständigen Sie die folgenden Methoden um die Ergebnisse der einzelnen Wettkämpfe herauszufinden.

- Die Methode `getOlympicEditions` soll eine Liste von Strings zurückgeben, die alle Editionen der olympischen Spiele repräsentieren. Jede Edition soll dabei nur einmal im Ergebnis vorkommen. Filtern Sie die Werte mit Hilfe eines Statement Objekts aus der Tabelle `game_summary`. In der GUI sollte nun die in Abb. 1 zu sehende Liste erscheinen, wenn Sie auf den Button *Choose Edition* klicken.

- Die Methode `getOlympicYears` soll eine Liste mit allen Jahren, in denen olympische Spiele einer ausgewählten Edition stattfanden, zurückgeben. Die Ergebnisse sollen aufsteigend sortiert sein.
  Initialisieren Sie hierzu zunächst in der Methode `prepareStatementOlympicYears` das Prepared Statement `pstmOlympicYears` entsprechend. Implementieren Sie dann die Methode `getOlympicYears`, in der Sie die als Parameter übergebene Edition verwenden, um die Variable in dem Prepared Statement `pstmOlympicYears` zu setzen und stellen Sie die Anfrage an die Datenbank. Filtern Sie die Jahreswerte aus dem Ergebnis der Anfrage und speichern Sie diese in der vorgegebenen Liste. In der GUI sollte nun die in Abb. 2 zu sehende Liste erscheinen, wenn Sie auf den Button *Choose Year* klicken und bei der Edition "Summer Olympics" ausgewählt haben.
  Hinweis: Spiele fanden immer dann statt, wenn `reason_not_held` einen NULL-Wert enthält.

- Die Methode `getOlympicEditionId` soll zu einer vorgegebenen Edition und einem dazugehörigen Jahr, welche der Methode als Parameter übergeben werden, die entsprechende Edition-ID zurückgeben. Die Edition-ID wird durch Edition und Jahr eindeutig bestimmt und ist in der Tabelle `game_summary` zu finden. Initialisieren Sie hierzu zunächst das Prepared Statement `pstmOlympicEditionIds` in der Methode `prepareStatementOlympicEditionId` entsprechend. Verwenden Sie dann in der Methode `getOlympicEditionId` die übergebenen Parameter, um die Variablen in dem Prepared Statement `pstmOlympicEditionIds` zu setzen. Stellen Sie nun die Anfrage an die Datenbank und bestimmen die ID mit Hilfe des Ergebnisses der Anfrage.

- Die Methode `getOlympicSports` soll eine Liste mit allen Sportarten zurückgeben, die bei einer vorgegebenen Edition der Olympischen Spiele vertreten waren. Die Edition wird dabei durch die der Methode übergebenen Edition-ID bestimmt. Jede Sportart soll hierbei nur einmal vorkommen und die Ergebnisse sollen lexikalisch aufsteigend sortiert sein. Verwenden Sie hierzu ein passenden Statement, welche die Sportarten aus der Tabelle `event_details` filtert und speichern Sie die Ergebnisse in der vorgegebenen Liste.
  In der GUI sollte nun die in Abb. 3 zu sehende Liste erscheinen, wenn Sie auf den

Button *Choose Sport* klicken und bei der Edition "Summer Olympics" und beim Jahr 2012 ausgewählt haben.

- Die Methode `getOlympicEvents` soll eine Liste mit allen Events (Wettbewerben) einer Sportart bei einer bestimmten Edition der Olympischen Spiele aus der Tabelle `event_details` filtern und in einer Liste zurückgeben. Die Sportart und die Edition-ID werden der Methode dabei als Parameter übergeben. Jedes Event soll hierbei nur einmal vorkommen und die Ergebnisse sollen lexikalisch aufsteigend sortiert sein.
  Initialisieren Sie hierzu zunächst das Prepared Statement `pstmOlympicEvents` in der Methode `prepareStatementOlympicEvents` entsprechend. Verwenden Sie dann in der Methode `getOlympicEvents` die übergebenen Parameter, um die Variablen in dem Prepared Statement `pstmOlympicEvents` zu setzen und stellen Sie die Anfrage an die Datenbank. Speichern Sie die Ergebnisse in der vorgegebenen Liste.
  In der GUI sollte nun die in Abb. 4 zu sehende Liste erscheinen, wenn Sie auf den Button *Choose Event* klicken und bei der Edition "Summer Olympics", beim Jahr 2012 und bei der Sportart "Athletics" ausgewählt haben.

- Die Methode `getOlympicResults` soll eine Liste mit Objekten der Klasse `Result` (zu finden im Package `data`) zurückgeben. Die Rückgabe soll die Ergebnisse einer bestimmten Edition der Olympischen Spiele von einem Event einer Sportart enthalten (die entsprechenden Werte werden der Methode als Parameter übergeben). Ein Objekt der Klasse `Result` besteht aus der Position, dem Namen und dem Herkunftsland des Athleten. Initialisieren Sie hierzu zunächst das Prepared Statement `pstmOlympicResults` in der Methode `prepareStatementOlympicResult` um die entsprechenden Daten aus der Datenbank zu filtern. Verwenden Sie die der Methode `getOlympicResults` übergebenen Parameter, um die Variablen in dem Prepared Statement `pstmOlympicResults` zu setzen. Werten Sie dieses dann aus, erstellen ein entsprechendes Result Objekt und fügen Sie dieses der Liste hinzu.
  In der GUI sollte nun die in Abb. 5 gezeigten Ergebnisse erscheinen, wenn Sie bei der Edition "Summer Olympics", beim Jahr 2012, bei der Sportart "Athletics" und beim event "1,500 metres, Women" ausgewählt haben.
  Hinweis: Um die benötigten Werte zu erhalten müssen Sie einen JOIN über die beiden Tabellen *athletes* und *event_ details* durchführen, so dass sie zu jedem Eintrag in der Tabelle *event_ details* die entsprechenden Daten aus der Tabelle *athletes* bekommen. Die JOIN-Bedingung müssen Sie jeweils über die *athlete_ id* formulieren. Sie müssen die Ergebnisse [2] `nicht` selbst sortieren: das wird in der GUI mit Hilfe der in der Klasse `Result` überschriebenen `compareTo` Methode gemacht.

[2] Die Bedeutungen der Abkürzungen finden Sie unter: https://www.olympedia.org/static/faq

Aufgabe 2(b) Informationen zu den Athleten filtern **10 P**

Vervollständigen Sie die im Folgenden genannten Methoden, um Informationen über die Athleten, die an den olympischen Spielen teilgenommen haben zu erhalten.

- Die Methode `getCountries` soll eine Liste mit den Namen aller Länder zurückgeben. Das Ergebnis soll lexikalisch aufsteigend sortiert sein. Verwenden Sie hierzu eine `Statement` Instanz und speichern Sie die Ergebnisse in der vorgegebenen Liste. Wenn Sie in der GUI nun zum Fenster `Athleten` wechseln und den Button *Choose Country* klicken, sollten die in Abb. 6 zu sehenden Ergebnisse erscheinen.

- Die Methode `getOlympicSports2` soll eine Liste mit allen Sportarten zurückgeben, an denen Athleten aus einem bestimmten Land (in mindestens einer Edition der Olympischen Spiele in mindestens einem Event) teilgenommen haben. Das Land wird als Parameter übergeben. Jede Sportart soll hierbei nur einmal vorkommen und die Ergebnisse sollen lexikalisch aufsteigend sortiert sein. Initialisieren Sie hierzu zunächst das Prepared Statement `pstmOlympicSports` in der Methode `prepareStatementOlympicSports` entsprechend. Verwenden Sie dann in der Methode `getOlympicSports2` den übergebenen Parameter, um die Variable im Prepared Statement `pstmOlympicSports` zu setzen und speichern Sie die Ergebnisse in der vorgegebenen Liste.
  In der GUI sollten nun die in Abb. 7 zu sehenden Ergebnisse erscheinen, wenn Sie als Land `Germany` auswählen und anschließend den Button *chooseSport* anklicken.
  Hinweis: Sie müssen die Tabellen *athletes* und *event_ details* wieder, wie in der letzten Teilaufgabe von Aufgabe 2a beschrieben, miteinander verbinden.

- Die Methode `getOlympicAthletes` soll eine Liste mit Objekten der Klasse `AthleteEventResult` (zu finden im Package *data*) zurückgeben. Ein Objekt der Klasse `AthleteEventResult` besteht aus der Medaille, dem Namen, dem Geburtsdatum, Größe und Gewicht, sowie den entsprechenden Spielen. Die Spiele setzten sich aus der Edition und dem Jahr zusammen. Das Land, aus dem die Athleten kommen sollen, die Sportart, die sie ausüben und ihr Geschlecht werden der Methode als Parameter übergeben.
  Initialisieren Sie zunächst das Prepared Statement `pstmtOlympicAthlethes` in der Methode `prepareStatementOlympicAthletes`, um alle Informationen, die Sie benötigen, um die `AthleteEventResult` Objekte zu erstellen, aus der Datenbank zu filtern. Sortieren Sie die Ergebnisse absteigend nach dem Jahr. Verwenden Sie anschließend in der Methode `getOlympicAthletes` die übergebenen Parameter, um die Variablen in dem Prepared Sattement `pstmtOlympicAthlethes` zu setzen und werten Sie dieses aus. Erzeugen Sie die entsprechenden `AthleteEventResult` Instanzen und speichern Sie diese in der vorgegebenen Liste.
  In der GUI sollten nun die in Abb. 8 zu sehenden Ergebnisse erscheinen, wenn Sie als Land "Germany", als Sportart "Alpine Skiing" und als Geschlecht "Female" auswählen.
  Hinweis: Sie müssen die Tabelle *event_ details* mit den Tabellen *athletes* und *game_ summary* so verbinden, dass Sie zu jedem Eintrag in der Tabelle *event_ details*

sowohl die entsprechenden Informationen über die Athleten als auch über die Spiele erhalten. Die JOIN-Bedingung müssen Sie für die Tabelle *athletes* über die *athlete_id* und für die Tabelle *game_summary* über die *edition_id* formulieren.
Weitere Hinweise zum Joinen mehrerer Tabellen finden Sie unter:
https://learnsql.de/blog/wie-man-3-tabellen-oder-mehr-in-sql-verbindet/

Aufgabe 2(c) Fakten zu den einzelnen Spielen filtern **8 P**

Vervollständigen Sie die im Folgenden genannten Methoden, um ein paar interessante, allgemeine Fakten über die olympischen Spiele zu erfahren. Wenn Sie in der GUI wieder zum Fenster `Ergebnisse` wechseln und dort als Edition "Summer Olympics" und als Jahr 1896 auswählen erscheint auf der linken Seite ein noch unvollständiger Text. Nach dem Bearbeiten jeder Aufgabe wird eine der Lücken ausgefüllt. Die vollständige Lösung für die gegebenen Werte finden Sie in Abb. 9.

- Die Methode `getOlympicVenue` soll mit Hilfe eines Statements den Austragungsort einer Edition der Olympischen Spiele aus der Tabelle `game_summary` filtern und zurückgeben. Die entsprechende *edition_id* wird der Methode als Parameter übergeben.

Hinweis: Bei den folgenden zwei Methoden müssen Sie Inline-Tabellen/ Unterabfragen verwenden und wieder die Tabellen *athletes* und *event_details* so miteinander verbinden, dass Sie für jeden Eintrag in der Tabelle *event_details* die entsprechenden Informationen über den jeweiligen Athleten erhalten.

- Die Methode `getOldestOrYoungestAthlete` soll ein String-Array mit dem Namen (erster Eintrag im Array) und dem Geburtsdatum (zweiter Eintrag im Array) des ältesten (übergebener Parameter oldest=true) oder des jüngsten (übergebener Parameter oldest=false) Athleten einer bestimmten Edition der Spiele (durch den der Methode übergebenen Parameter `edition_id` spezifiziert) zurückgeben.
  Hinweis: Finden Sie hierzu zuerst das kleinste bzw. größte Geburtsdatum des Athleten, der an dem entsprechenden Spiel teilgenommen hat. Diese Anfrage müssen Sie nun, ähnlich wie bei Inline-Tabellen, weiterverwenden, allerdings in der `where`-Klausel. Filtern Sie damit dann die entsprechenden Informationen zum Athleten aus.
  Ihre Anfrage sollte nun folgende Form haben:
  SELECT ... WHERE born = (*Hier Ihre Anfrage zum kleinsten/größten Geburtsdatum*).
  Dies funktioniert, da die Aggregatfunktion max (min) nur ein Ergebnis zurückgibt und dieses somit in der `where`-Klausel verwendet werden kann.

- Die Methode `getMaleFemaleCount` soll die Anzahl der männlichen (übergebener Parameter male=true) bzw. der weiblichen (übergebener Parameter male=false) Teilnehmer eines bestimmten Spiels (durch den übergebenen Parameter `edition_id` bestimmt) zurückgeben.

Hinweis: Zählen Sie in der Inline-Tabelle alle IDs der Athleten des gesuchten Geschlechts bei den gesuchten Spielen.

Bonus: Ewigen Medaillenspiegel erstellen **5 P**

In dieser Aufgabe sollen Sie mit den Informationen aus der Tabelle *medal_history* den ewigen Medaillenspiegel, also wie viele Medaillen ein Land bei allen Spielen zusammengezählt gewonnen hat, erstellen. Hierzu benötigen Sie ein `group by` Statement. Wie dieses funktioniert können Sie dem folgenden Beispiel mit der Tabelle *students* entnehmen:

name	aufgabe	punkte	maxpunkte
Emily	a1	5	5
Emily	a2	12	15
Torben	a1	3	5
Torben	a2	1	15

Mit folgendem SQL-Statement wird die Summe aller erreichten Punkte ermittelt:

```
Select name,sum(punkte) as punkte
from students
group by name
```

name	punkte
Emily	17
Torben	4

Nach dem `group by` kann man noch eine `having` Klausel einfügen, um nur bestimmte Ergebnisse zu filtern. Fügen wir im obigen Beispiel nach der Gruppierung noch `having sum(punkte) >= 10` ein, werden nur noch Studenten, die in Summe über alle Aufgaben zehn oder mehr Punkte haben, im Ergebnis angezeigt.

Vervollständigen Sie nun die folgende Methode:

- Die Methode `getOlympicMedalTally` soll eine Liste mit String-Arrays zurückgeben. Jedes Array soll dabei die folgenden vier Einträge (in dieser Reihenfolge) enthalten: den Namen des Landes, die Anzahl an Goldmedaillen, die Anzahl an Silbermedaillen und die Anzahl an Bronzemedaillen. Filtern Sie dazu die entsprechenden Werte aus der Tabelle *medal_history* und gruppieren Sie diese nach dem Land. Wählen sie dabei nur solche Länder aus die mindestens eine Goldmedaille, eine Silbermedaille und eine Bronzemedaille erhalten haben. Die Ergebnisse sollen absteigend nach der Anzahl an Goldmedaillen, bei Gleichstand nach Anzahl der Silbermedaillen und bei erneutem Gleichstand nach der Anzahl der Bronzemedaillen sortiert sein.

Wenn Sie in der GUI nun zum Fenster `Medaillenspiegel` wechseln, sollten die in Abb. 10 zu sehenden Ergebnisse erscheinen.

Abb. 1

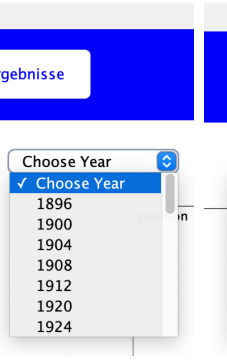


Abb. 2

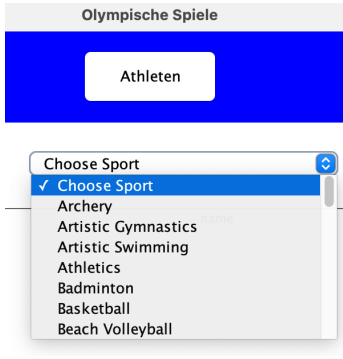


Abb. 3

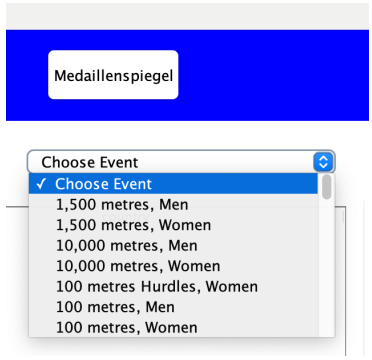


Abb. 4

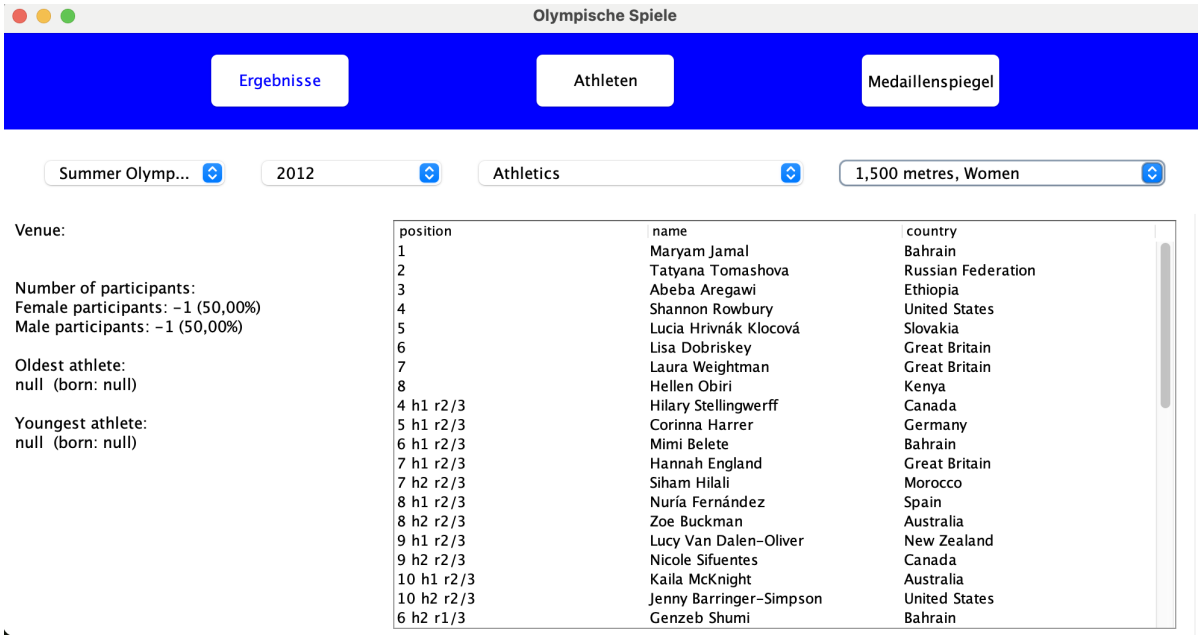


Abb. 5

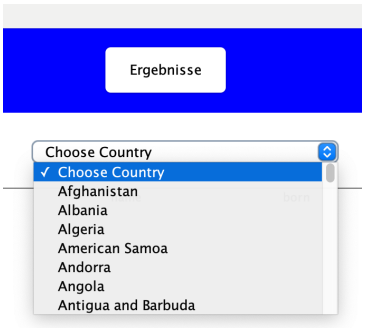


Abb. 6

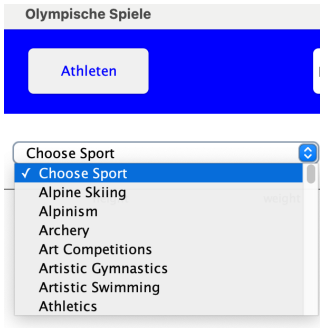


Abb. 7

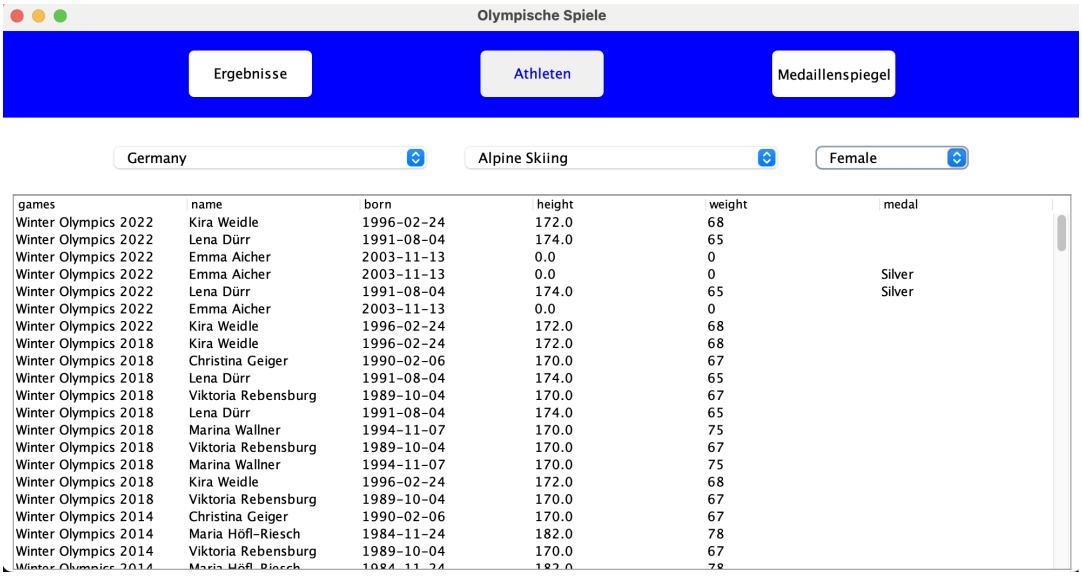


Abb. 8

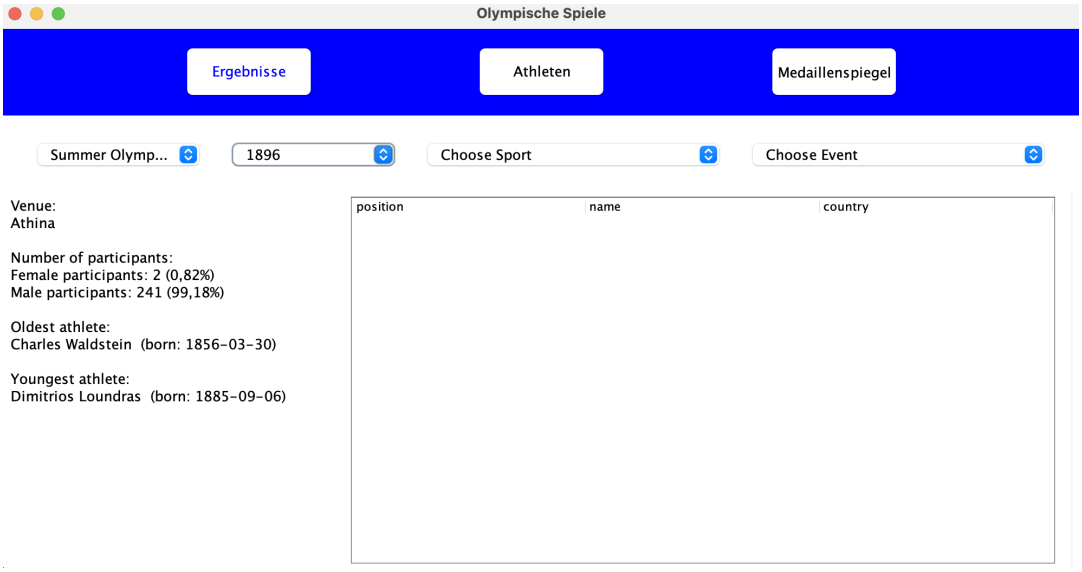
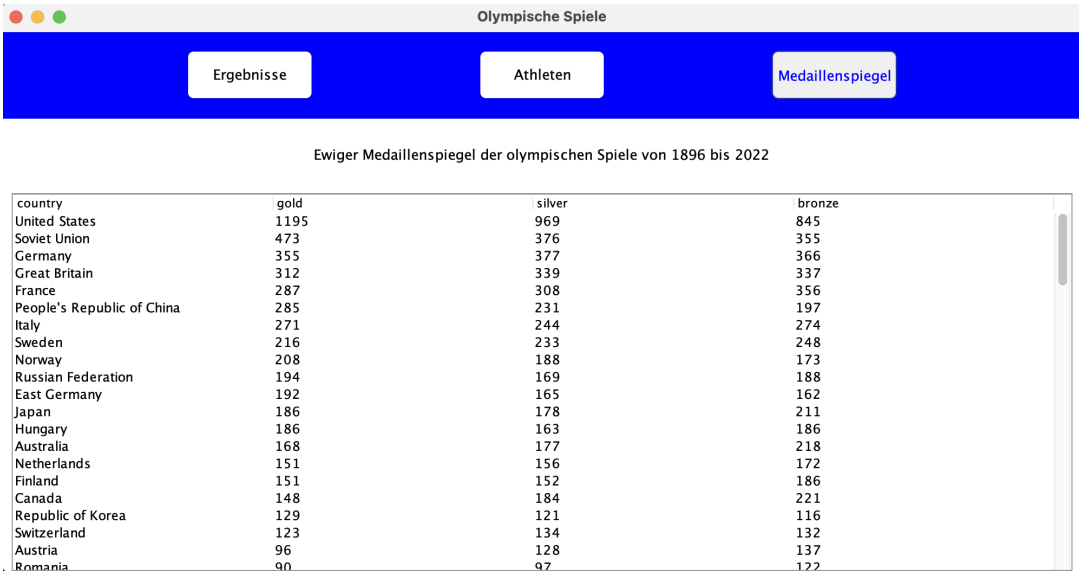


Abb. 9



country	gold	silver	bronze
United States	1195	969	845
Soviet Union	473	376	355
Germany	355	377	366
Great Britain	312	339	337
France	287	308	356
People's Republic of China	285	231	197
Italy	271	244	274
Sweden	216	233	248
Norway	208	188	173
Russian Federation	194	169	188
East Germany	192	165	162
Japan	186	178	211
Hungary	186	163	186
Australia	168	177	218
Netherlands	151	156	172
Finland	151	152	186
Canada	148	184	221
Republic of Korea	129	121	116
Switzerland	123	134	132
Austria	96	128	137
Romania	90	97	122

Abb. 10