P3: Datenbanken, XML

Abgabetermin: 8. Februar 2021

Punktzahl: 35

In diesem Projekt sollen Sie eine Applikation entwickeln, welche einem Spieler ermöglicht ein Quiz zu spielen. Um ein Quiz zu starten muss der Spieler seinen Namen eingeben sowie eine Kategorie und einen Schwierigkeitsgrad wählen. Dem Spieler stehen jeweils drei Kategorien von Fragen und Schwierigkeitsgraden zur Verfügung. Die Kategorien sind Geografie, Filme und Musik. Die Schwierigkeitsgrade sind leicht, mittel und schwer. Jedes Quiz enthält zehn Fragen.

In Bezug auf den Schwierigkeitsgrad nehmen wir an, dass in der Kategorie Geografie Fragen mit Länder aus der EU als leicht eingestuft werden, Fragen mit Länder aus Nordamerika als mittel und die restlichen Fragen als schwer. In der Kategorie Filme werden nur Komödien, Abenteuer und Fantasie-Filme abgefragt. Dabei werden Komödien als leicht eingestuft, Abenteuer-Filme als mittel und Fantasie-Filme als schwer. In der Kategorie Musik richtet sich der Schwierigkeitsgrad nach dem Erscheinungsjahr. Ein Lied mit einem Erscheinungsjahr zwischen 2000 und 2007 (jeweils inklusive) wird als schwer eingestuft, ein Erscheinungsjahr zwischen 2008 und 2014 wird als mittel eingestuft und ein Erscheinungsjahr zwischen 2015 und 2020 als leicht.

Vorgaben

Die Vorgaben unter `P3.zip` definieren bereits eine Grundstruktur, die Sie für Ihre Implementierung verwenden sollen. Darin sind mehrere Packages vorgegeben, die nachfolgend kurz erläutert werden.

data

In diesem Package finden Sie die Klassen: Frage, Antwort und Spieler, welche die zentralen Datenobjekte in dieser Applikation repräsentieren.
Die Klasse Frage repräsentiert eine Frage vom Quiz, die Klasse Antwort repräsentiert eine Antwortmöglichkeit einer Frage. Die Klasse Spieler repräsentiert einen Spieler des Spiels.
Die drei Klassen enthalten jeweils zwei Konstruktoren, Attribute für alle Eigenschaften,

die in der Datei data.xml genannt sind, sowie auch Getter-Methoden. Die Klasse Spieler enthält noch zwei Hilfsmethoden. Die Methode generateNumber generiert und weist dem aktuellen Spieler eine zufällige Spielernummer zwischen 2050 und 2150 zu. Die Methode increaseScore erhöht die Punktzahl des Spielers um einen Punkt.

db

In diesem Package sind alle Klassen, die mit der von Ihnen zu generierenden Datenbank zu tun haben.

- Die Klasse Database beinhaltet alle Methoden, die für das Öffnen und Schließen der Verbindung zu einer Datenbank notwendig sind. Der Konstruktor der Klasse initialisiert alle Attribute und öffnet eine Verbindung. Des weiteren besitzt die Klasse ein Attribut connection, welches in den Unterklassen verwendet werden kann um SQL Statements zu generieren. Die Methode disconnect schließt die Verbindung mit einer Datenbank.

- Die Klasse CreateDatabase dient dazu eine Datenbank mit den drei Tabellen Frage, Antwort und Spieler anzulegen. Beim Aufruf der Klasse wird eine Verbindung zur Datenbank aufgebaut. Mithilfe der Methode dropTables können alle Tabellen, mit den Namen Frage, Antwort und Spieler, sofern diese in der Datenbank bereits existieren, gelöscht werden. Die Methode createTables, erzeugt die drei Tabellen, indem diese die Methoden aufruft, welche später von Ihnen zu implementieren sind.

- Die Klasse FillDatabase dient dazu, die von Ihnen angelegt Datenbank mit Daten zu füllen. In der Klasse sind bereits drei Prepared-Statements deklariert, welche von Ihnen später zu verwenden sind. Mithilfe der Methode clearTable, wird der Inhalt der Tabelle mit dem als Parameter übergebenen Namen, sofern diese existiert, gelöscht. Die Methode closeStatements schließt alle Prepared-Statements, welche noch offen sind. Die Methoden prepareStatements, fillFrage, fillAntwort und fillPlayer sind teilweise vorgegeben und von Ihnen zu implementieren.
  Beim Ausführen der Klasse werden die von Ihnen generierten Tabellen, mithilfe einer XML_Reader-Instanz mit Daten aus der XML-Datei gefüllt.

- Die Klasse QuizDB dient dazu, Anfragen an die Datenbank, die für das Spiel benötigt werden, durchzuführen. In der Klasse sind bereits sieben Prepared-Statements deklariert, welche von Ihnen verwendet werden müssen. Alle Methoden sind bereits leer vorgegeben oder geben einen Dummy-Wert zurück und sind später zu implementieren.

general

In diesem Package befindet sich eine Klasse Parameter, in der einige globale String-Konstanten definiert sind. Dazu gehören verschiedene Pfade zu Dateien aus dem resources-Verzeichnis und eine Variable, die die Länge der Applikation festlegt.

gui

In diesem Package finden Sie Klassen, die die Graphische Oberfläche definieren und die Interaktion damit über entsprechende Listener steuern.

Die Klasse MainGui definiert den Haupt-Frame der Applikation und verwaltet die anzuzeigenden Panels. Nach Aufrufen dieser Klasse wird der Name des Spielers, die Kategorie und der Schwierigkeitsgrad erfasst. Die Klasse enthält ein Menu mit den Menüpunkten Neues Quiz starten, Statistik zeigen und Schließen. Das Menu und die benötigten Listener werden in der Methode createMenu initialisiert und der Applikation hinzugefügt. Die Methode showStatistics zeigt das Statistik-Panel. Die Methode openQuiz zeigt das Quiz-Panel und fügt dem Button einen Listener, welcher nach Ende eines Spiels die Statistik und die Spielerleistung zeigt.

Die Klasse QuizPanel erzeugt ein Panel, welches für das tatsächliche Spiel benötigt wird. Der Konstruktor der Klasse initialisiert alle Komponente des Panels. Mithilfe der Methode updatePanel, wird das Panel aktualisiert, indem drei weitere Methoden aufgerufen werden. Die drei Methoden zeigen eine neue Frage, die dazugehörigen Antwortmöglichkeiten und passen den Fortschritt des Spiels in der ProgressBar entsprechend an. Ein Button, welcher mit einem Listener bereits ausgestattet ist, ruft eine Aktualisierung des Panels hervor.

Die Klasse StatisticsPanel dient dazu, dem Spieler die Möglichkeit zu geben, sich mit ehemaligen Spieler zu vergleichen. Der Konstruktor der Klasse erzeugt ein Panel, welches die Leistung des Spielers und eine Tabelle enthält. Die Tabelle enthält alle ehemalige Spieler absteigend sortiert nach ihrer Leistung und wird mithilfe der Methode fillTable gefüllt.

io

Das Package io enthält alle Klassen, die dazu dienen, die Daten aus der, im resources-Verzeichnis befindliche XML-Datei, einzulesen und zu parsen.
Die Klasse XML-Reader enthält die Methode parseFile sowie drei Getter-Methoden und wird zum Lesen der XML-Datei benötigt. Die Methode generiert einen XML-Reader und mithilfe einer QuizContentHandler-Instanz wird die Datei durchlaufen. Am Ende des Parsedurchlaufs werden den Listen fragen, antworten und spieler die Daten aus der XML zugewiesen.
Die Klasse QuizContentHandler bildet einen ContentHandler, den Sie später vervollständigen sollen. Die Klasse enthält drei Listen, die später mit Frage, Antwort und Spieler-Instanzen aus dem XML-Dokument gefüllt werden sollen.

resources

In diesem Verzeichnis ist folgendes vorgegeben:

- ein Unterverzeichnis images mit den Bildern der Kategorien.
- eine DTD, welche das XML-Format der Datei data.xml beschreibt.

- eine Datei data.xml, mit allen verfügbaren Fragen, Antworten und Spieler.
- Die Datenbank quiz.db
- Die Library sqlite-jdbc-3.34.0.jar
- Die Library jdom-2.0.6.jar

Der Java Build Path des Projekts wurde bereits mit den beiden Libraries für JDOM und JDBC entsprechend ergänzt.

Benutzeroberfläche

Die folgenden Abbildungen zeigen wie die Benutzeroberfläche nach Implementierung der einzelnen Aufgaben aussehen soll.

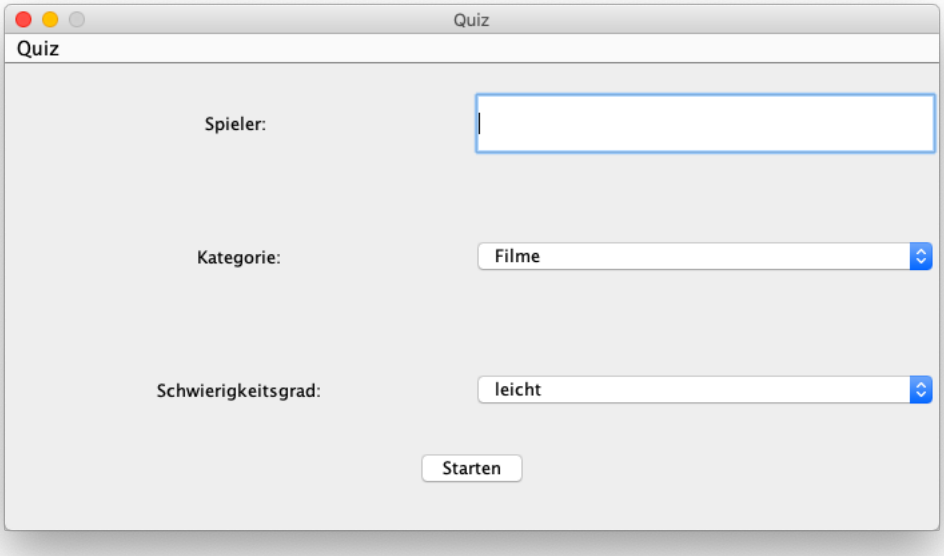

Abbildung 1: Nach der erfolgreichen Implementierung der Aufgaben (a) - (d) wird die Oberfläche Kategorien und Schwierigkeitsgrade enthalten.

Abbildung 2: Nach der erfolgreichen Implementierung der Aufgabe (e) wird die Oberfläche je nach gewählter Kategorie nach dem Starten des Spiels Fragen und Antwortmöglichkeiten anzeigen. Das Bild neben der Frage kennzeichnet die Kategorie der Fragen.

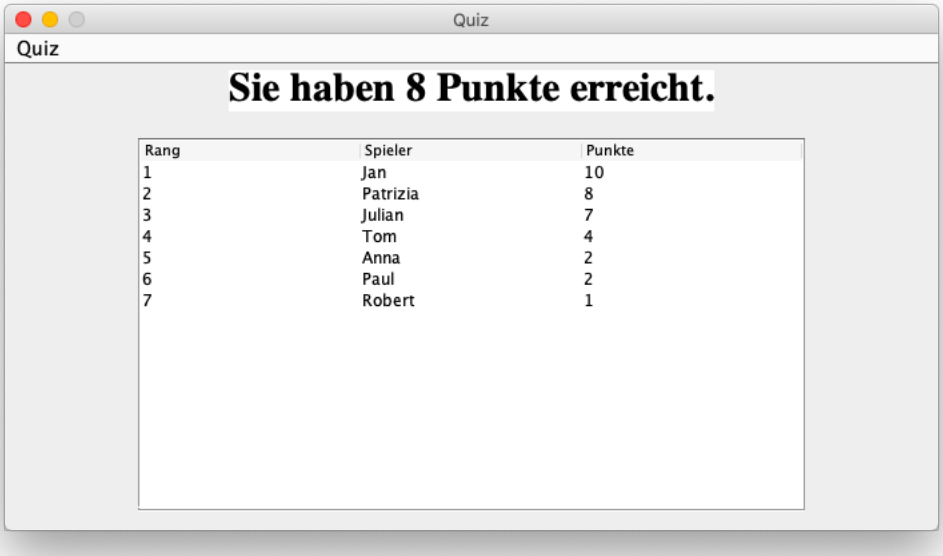

Abbildung 3: Nach erfolgreicher Implementierung der Aufgabe (g) wird Ihnen am Ende eines Spiels Ihre Leistung und die Statistik angezeigt.

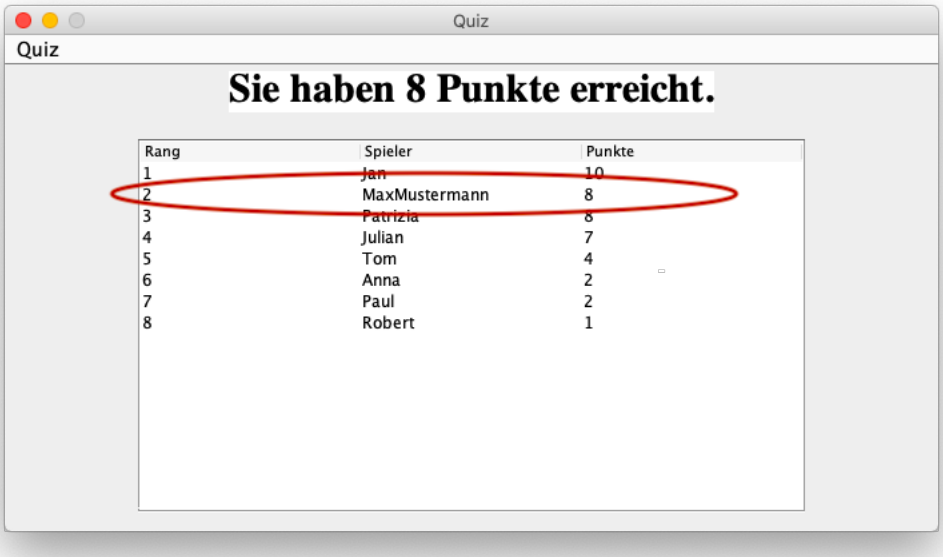

Abbildung 4: Nach erfolgreicher Implementierung der Aufgabe (h) werden Sie in der Tabelle ihre Daten sehen.

Aufgabenstellung

(a) (6 P) Tabellen erstellen

In diesem Schritt sollen Sie die drei Methoden createTableFrage, createTableAntwort und createTableSpieler in der Klasse db.CreateDatabase implementieren. Die Tabellen müssen gemäß der folgenden Schemadefinitionen in der Datenbank angelegt werden. Das Attribut Antwort soll dabei einen Fremdschlüssel für das Attribut AntwortNr in der Tabelle Frage bilden. Das Attribut Frage in der Tabelle Antwort bildet einen Fremdschlüssel, der das Attribut FrageNr in der Tabelle Frage referenziert.

Frage	
FrageNr	INTEGER PRIMARY KEY
Frage	TEXT NOT NULL
Antwort	INTEGER NOT NULL
Schwierigkeitsgrad	INTEGER NOT NULL
Kategorie	TEXT NOT NULL

Antwort	
AntwortNr	INTEGER PRIMARY KEY
Antwort	TEXT NOT NULL
Frage	INTEGER NOT NULL

Spieler	
SpielerNr	INTEGER NOT NULL
Name	TEXT NOT NULL
Punkte	INTEGER NOT NULL

(b) (4 P) XML-Dokument filtern

In diesem Schritt sollen Sie die Daten aus dem XML-Dokument vom resources-Verzeichnis, mithilfe eines SAX-Parsers parsen.

Ergänzen Sie dazu die Klasse io.QuizContentHandler, indem Sie die Methoden startElement und endElement implementieren. Das XML-Format können Sie aus der DTD aus dem resources-Verzeichnis entnehmen. Die Klasse enthält drei Listen, die später verwendet werden um die Datenbank mit Daten zu füllen. Sie können Hilfsvariablen nutzen um die benötigten Instanzen in den Listen hinzuzufügen.
Sie können davon ausgehen, dass jede Frage im XML-Dokument genau einmal vorkommt.

(c) (6 P) Datenbank füllen

Im dritten Schritt sollen die Tabellen Frage, Antwort und Spieler gefüllt werden. Ergänzen Sie dazu die Klasse db.FillDatabase wie folgt:

- prepareStatements: Hier sollen Sie die drei Prepared-Statements, die als Attribute in der Klasse deklariert sind, mit je entsprechenden SQL-Statements initialisieren.
- Implementieren Sie die Methoden fillFrage, fillAntwort und fillSpieler. Hier müssen Sie die Variablen in dem zugehörigen Prepared-Statement setzen und ausführen. Nutzen Sie hierzu die als Parameter übergebenen Instanzen aus der ArrayList.

(d) (4 P) Basisinformationen generieren

Um nun noch verschiedene Daten aus der Datenbank zu filtern, die beispielsweise zum Füllen der Comboboxen in der Oberfläche benötigt werden, müssen Sie die folgenden Methoden in der Klasse db.QuizDB implementieren:

- getKategorien: Diese Methode soll eine Sammlung von Kategorien generieren und zurückgeben, die in der Tabelle Frage enthalten sind. Sorgen Sie dafür, dass alle Kategorien nur einmal in der Ergebnisliste erfasst werden und dass die Kategorien lexikalisch aufsteigend sortiert werden.
- getSchwierigkeitsgrad: Diese Methode soll eine Sammlung von Zeichenketten generieren und zurückgeben, die alle Schwierigkeitsgrade aus der Tabelle Frage enthält. Schreiben Sie ein SQL-Statement, das alle Schwierigkeitsgrade ohne Duplikate zurückgibt und diese lexikalisch aufsteigend sortiert. Da die Schwierigkeitsgrade in der Datenbank als Zahlen erfasst werden, müssen diese in geeignete Zeichenkette umgewandelt werden. Gehen Sie davon aus, dass ein Schwierigkeitsgrad drei einer schweren Aufgabe entspricht, der Schwierigkeitsgrad zwei einer mittleren Aufgabe und der Schwierigkeitsgrad eins einer leichten Aufgabe entspricht.

Nutzen Sie für die beiden Methoden jeweils ein in der Klasse bereits deklariertes Prepared-Statement.

(e) (6 P) Fragen und Antworten generieren

Um die passenden Fragen und Antworten zu generieren, die der selektierten Kategorie und dem selektierten Schwierigkeitsgrad entsprechen, müssen Sie nun die Methoden getAntworten und getFragen in der Klasse db.QuizDB implementieren:

- getFragen: Diese Methode soll eine Sammlung von Fragen generieren und zurückgeben. Es sollen nur Fragen einer bestimmten Kategorie ausgewählt werden, dessen Name als Parameter übergeben wird, und eines bestimmten Schwierigkeitsgrades, deren Wert ebenfalls als Parameter übergeben wird. Denken Sie daran, die passenden Datentypen beim Erzeugen der neuen Frage-Instanzen zu verwenden.
- getAntworten: Diese Methode soll eine Sammlung von vier Antworten generieren und zurückgeben, die für eine bestimmte Frage als Antwortmöglichkeiten angezeigt werden. Initialisieren Sie ein, der bereits in der Klasse deklarierten Prepared-Statement an, welches die Antwort und die Antwortnummer einer Antwort mit einer bestimmten Frage aus der Datenbank holt und sortieren Sie das Ergebnis lexikalisch aufsteigend nach der Antwort.

(f) (3 P) Korrekte Antwort auswählen

Implementieren Sie nun die Methode getKorrekteAntwort in der Klasse db.QuizDB. Hier sollen Sie die korrekte Antwort einer Frage zurückgeben, deren Nummer als Parameter übergeben ist. Nutzen Sie hierzu eine Inline-Tabelle, die die Antwort einer Frage zurückgibt, die der als Parameter übergebenen Fragenummer entspricht.

(g) (3 P) Spielerstatistik entwickeln

Hier sollen Sie eine Statistik entwickeln, die dem aktuellen Spieler erlaubt, sich mit anderen Spielern zu vergleichen. Implementieren Sie die Methode getStatistik und initialisieren Sie ein bereits in der Klasse deklariertes Prepared-Statement. Implementieren Sie die Methode wie folgt:

- Generieren Sie eine Sammlung von Spielern, die mindestens eine Frage richtig beantwortet haben.
- Sortieren Sie die Spieler absteigend nach ihrer erreichten Punktzahl und lexikalisch aufsteigend nach ihren Namen.
- Denken Sie daran, die passenden Datentypen beim Erzeugen der neuen Spieler-Instanzen zu generieren.

(h) (3 P) Spieler hinzufügen

Hier sollen Sie schließlich die Änderung der Datenbank durchführen, die durch das Hinzufügen eines Spielers notwendig werden. Implementieren Sie hierzu die Methode addSpieler in der Klasse db.QuizDB. Gehen Sie wie folgt vor:

- Filtern Sie mithilfe der als Parameter übergebenen Spieler-Instanz, den Name, die Nummer und die Punktzahl.
- Schreiben Sie ein SQL-Statement, welches den Spieler in der Tabelle Spieler hinzufügen soll.
- Stellen Sie sicher, dass Ihre Änderungen permanent in der Datenbank gespeichert werden.