Projekt: Dienstwagendatenbank Programmierung mit Java

Aufgabe
Es soll eine Anwendung in Java entstehen, welche sich mit den Dienstwagen und deren Fahrten beschäftigt. Die genauen Anforderungen an das Programm werden im folgenden Dokument beschrieben. Es soll auf ein objektorientiertes Datenmodell geachtet werden und es sollen Tests für die wichtigsten Funktionen vorhanden sein. Es dürfen außer JUNIT und dem JDK keine weiteren Bibliotheken verwendet werden. Es sind ebenfalls keine anderen Abhängigkeiten wie Builds-Tools oder ähnliches erlaubt.
Zusätzlich zu dem Programm soll ein Klassendiagramm angefertigt werden, welche sowohl die Beziehung der Klassen beschreibt, wie auch deren Methoden. Dieses Klassendiagramm darf nicht generiert werden und muss im PDF oder PNG Format abgegeben werden. Im Klassendiagramm müssen alle im Projekt verwendeten Klassen aufgeführt werden.
Unnötig komplexer Code muss vermieden werden.

Abgabe
Bitte sendet euer Projekt als Zip-Datei (.zip!) an unterstein@me.com. In dieser Zip-Datei müssen sowohl Quelltext, als auch Klassendiagramm enthalten sein. Der Name der Zip-Datei muss dem Muster "Dienstwagenprojekt_Kurs_Matrikelnummer.zip" entsprechen, wobei Kurs und Matrikelnummer durch den realen Wert ersetzt werden muss. Abgabeende ist der 03.08.2025 23:59:59.

Datenmodell
Das Datenmodell dieser Anwendung enthält Dienstwagen, Fahrer und Fahrten.
Dienstwagen haben eine ID, einen Hersteller, ein Modell und ein Kennzeichen. Ein Fahrer hat eine ID, einen Vornamen, einen Nachnamen und eine Führerscheinklasse. Eine Fahrt hat einen Start Kilometerstand, End Kilometerstand. und Startzeit und Endzeit.

Daten laden
Es muss möglich sein die Datei https://wwwlehre.dhbw-stuttgart.de/~unterstein/dienstwagenprojekt2025.db zu importieren und in das Datenmodell des Programmes zu überführen. Eine Zeile, welche mit 'New_Entity:' beginnt, leitet ein neues Objekt des Datenmodelles ein. Es gibt folgende 3 Datensätze:
1. Fahrer
2. Dienstwagen
3. Fahrt

In der Zeile, die mit 'New_Entity:' beginnt, sind Beschreibungen für die Spalten der folgenden Zeilen enthalten. Die Werte sind jeweils mit Komma von einander getrennt und jeweils von Anführungszeichen eingegrenzt. Die trennenden Kommas und Anführungszeichen sollen nicht ins Datenmodell importiert werden.
Der Import soll beim Starten des Programmes erfolgen, kann aber von einer lokalen Datei laden und muss nicht die URL jedes mal neu abfragen.
Beim importieren der Daten sollen überflüssige Leerzeichen am Anfang und Ende von Feldern (z.B. Name des Fahrers) entfernt werden. Das Datenmodell kann Dubletten enthalten: z.B. ein Dienstwagen der mehrfach vorkommt, zum Beispiel Kennzeichen oder FahrzeugID.

Fahrer suchen
Für den Benutzer des Programmes muss es möglich sein die ID und Details eines Fahrers anzuzeigen. Dabei soll zum Beispiel die Suche nach ‘Hoff”, alle Fahrer “Hoff” und “Hoffmann” liefern. Das Ausgabeformat ist nicht näher definiert, es müssen aber alle Details angezeigt werden.

Fahrzeug suchen
Für den Benutzer des Programmes muss es möglich sein die ID und Details eines Fahrzeugs anzuzeigen. Dabei soll zum Beispiel die Suche nach ‘Volkswagen’ unter anderem den Golf “S-DE-7839” liefern. Die Suche nach, zum Beispiel, “S-DE-7839” oder “Golf” muss auch möglich sein. Das Ausgabeformat ist nicht näher definiert, es müssen aber alle Details angezeigt werden.

Feature 1: Fahrzeug wurde geblitzt und Fahrer muss gefunden werden.
Für den Benutzer des Programmes muss es möglich sein, den Fahrer zu ermitteln für ein gegebenes Kennzeichen und ein definierten Zeitpunkt. Sollte es keinen Fahrer zu diesem Zeitpunkt geben, muss die Ausgabe lauten “Kein Fahrer gefunden”.

Die Eingabe sieht so aus: “${Kennzeichen};${Datum}”
Beispiel: "S-MN-9932;2024-02-14T13:57:43"
Die Ausgabe muss in folgendem Format geschehen:
Paul Becker

Feature 2: Fahrer hat etwas im Fahrzeug liegen lassen und sucht andere Fahrer
Für Benutzer des Programm muss es möglich sein, für einen gegeben Tag und einen gegebenen Fahrer, alle Fahrzeuge dieses Fahrers an diesem Tag zu finden und dann alle anderen Fahrer dieser Fahrzeuge am gleichen Tag auszugeben. Der initial suchende Fahrer darf nicht Teil der Ausgabe sein. Die Ausgabeliste muss Alphabetisch aufsteigend sortiert sein. Wenn Fahrer für mehrere Fahrzeuge gefunden werden, kommen diese pro Fahrzeug ein mal in der Ausgabe vor.

Die Eingabe sieht so aus: “${Fahrer};${Datum}"
Beispiel: “F003;2024-08-13”
Ausgabe muss in folgendem Format sein:
Ben Wagner (S-GH-3277), Mia Hoffmann (S-GH-3277).

Statischer Modus
Es wird keine Oberfläche für dieses Programm benötigt. Die Kommunikation mit dem Programm erfolgt mittels Übergabe von Argumenten auf der Kommandozeile. Argumente können keine Listen sein, sondern immer nur konkrete Suchbegriffe bzw. Ids. Das Programm soll folgende Argumente verstehen können:

Fahrersuche: z.B.: --fahrersuche="Hoff"
Fahrzeugsuche: z.B.: --fahrzeugsuche="Volkswagen"
Geblitzt: z.B.: --fahrerZeitpunkt="S-MN-9932;2024-02-14T13:57:43"
Liegen gelassen: z.B.: --fahrerDatum="F003;2024-08-13"

Unittests und Test auf Korrektheit
Die wichtigste Funktionalität (zum Beispiel die genannten vier Anforderungen, Import und so weiter) muss in einem Unittest geprüft werden. Zusätzlich zu den eigenen Unittests werden von mir Tests auf Korrektheit des Projektes durchgeführt. Der Projekttester.java wird im Laufe der Projektzeit bereit gestellt.

Bewertung

Aufgabe	Prozentpunkte
OO Datenmodell	20
UML Diagramm	15
Import aus File	20
Unittests	5
Fahrzeugsuche	5
Fahrersuche	5
Fahrzeug wurde geblitzt	7
Fahrer hat etwas liegen lassen	7
Code verständlich und sinnvolle Docs	10
Test auf Korrektheit	6
	100

Regeln für die Bewertung
1.) Kleine Abweichungen in der Bewertungsmatrix sind später noch möglich, z.B. falls ich was vergessen habe.
2.) Viele Punkte setzen voraus, dass das Programm kompiliert.
3.) Es gibt Abzüge falls gegen die oben genannten Anforderungen verstoßen wird, z.B. externe Abhängigkeiten, unnötig komplexer Code oder wenn das Dateiformat mißachtet wird.
4.) Der Projekttester sind keine Unit-Tests! Unit tests dürfen nicht den Standard-Out oder Standard-Err Stream abgreifen.