Name: _____

Abiturprüfung 2019

Informatik, Leistungskurs

Aufgabenstellung:

Das Unternehmen *Rent a Bike in a City* verleiht in einer Stadt an mehreren Verleihstationen Fahrräder an Kundinnen und Kunden.

Eine Kundin oder ein Kunde kann an einer Fahrradstation ein Fahrrad ausleihen und an einer beliebigen Fahrradstation zurückgeben.

Wenn ein Kunde oder eine Kundin ein Fahrrad ausleiht, dann wird der Ausleihzeitpunkt erfasst. Bei Rückgabe werden der Zeitpunkt und die gefahrenen Kilometer protokolliert. Ein Fahrrad ist nur an der Station, an der es steht, ausleihbar. Ein verliehenes Fahrrad ist somit nicht ausleihbar.

Zur Verwaltung der Daten möchte das Unternehmen eine relationale Datenbank aufbauen, die der folgenden Teilmodellierung entspricht:

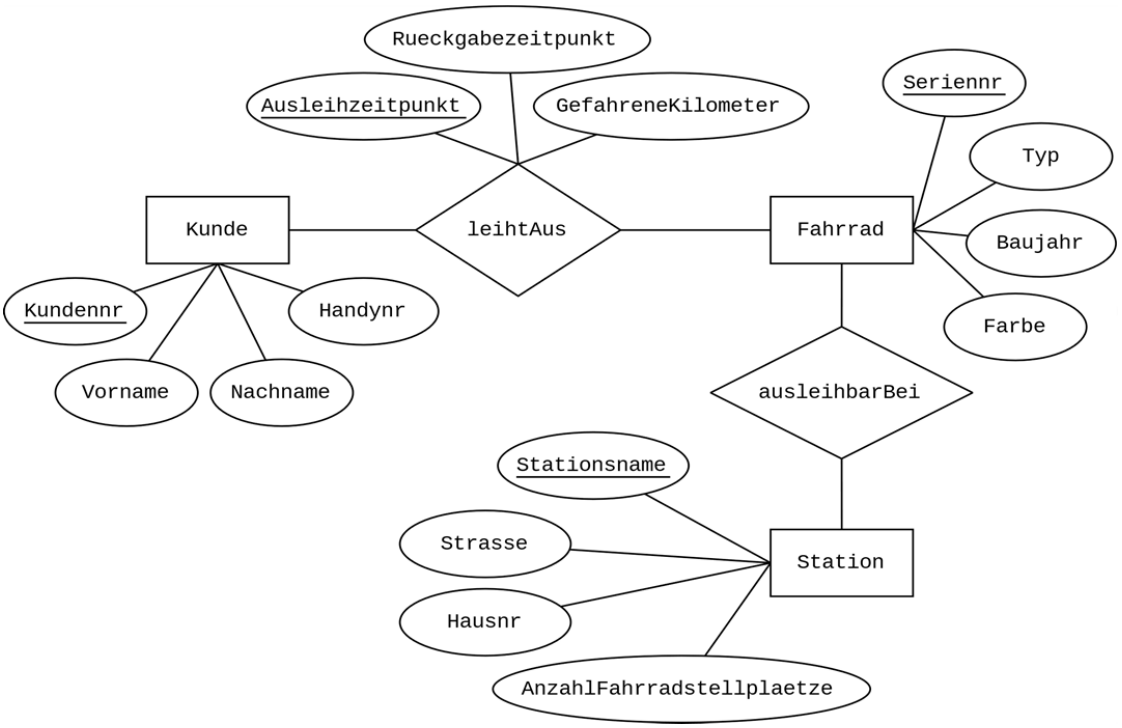

Abbildung 1: Teilmodellierung der Datenbank für *Rent a Bike in a City*

Name: _____

Im Folgenden soll mit dem in Abbildung 2 gegebenen Datenbankschema gearbeitet werden. Es stellt einen Ausschnitt der Gesamtdatenbank dar. Beispieldaten zu diesem Datenbankschema sind in der Anlage zu finden.

Kunde(Kundennr, Vorname, Nachname, Handynr)

leihtAus(↑Kundennr, ↑Seriennr, Ausleihzeitpunkt,
Rueckgabezeitpunkt, GefahreneKilometer)

Fahrrad(Seriennr, Typ, Baujahr, Farbe, ↑Stationsname)

Station(Stationsname, Strasse, Hausnr, AnzahlFahrradstellplaetze)

Abbildung 2: Datenbankschema eines Ausschnittes der Datenbank

a) *Beschreiben Sie die in Abbildung 1 gegebene Teilmodellierung.*

Ermitteln Sie die Kardinalitäten für das Entity-Relationship-Modell, so dass diese der Umsetzung im Datenbankschema entsprechen.

Begründen Sie anhand des Kontextes, warum die Attribute `Kundennr` *und* `Seriennr` *gemeinsam mit dem Attribut* `Ausleihzeitpunkt` *in der Relation* `leihtAus` *den kombinierten Primärschlüssel bilden.*

(12 Punkte)

b) Aus der Datenbank sollen folgende Informationen abgefragt werden:

i. Gesucht sind die Seriennummern – aufsteigend sortiert – aller roten Fahrräder im Bestand, die vor dem Jahr 2017 gebaut wurden.
ii. Gesucht sind die Vornamen, Nachnamen und Handynummern der Kunden, die ein Fahrrad ausgeliehen hatten und damit beim Ausleihvorgang mehr als 10 Kilometer gefahren sind.
iii. Gesucht sind die Seriennummern aller Fahrräder mit Baujahr, Summe der gefahrenen Kilometer und der Anzahl der Verleihvorgänge.
Das Ergebnis soll erstens nach dem Baujahr der Fahrräder aufsteigend und zweitens nach der Summe der gefahrenen Kilometer absteigend sortiert sein.

Entwerfen Sie für die obigen Anfragen i, ii und iii jeweils eine SQL-Anweisung.

(10 Punkte)

Name: _____

c) Folgende SQL-Anweisung ist gegeben:

```
 1  SELECT Station.Stationsname,
 2    Station.AnzahlFahrradstellplaetze,
 3    COUNT(*) AS X,
 4    Station.AnzahlFahrradstellplaetze - COUNT(*) AS Y,
 5    COUNT(*) / Station.AnzahlFahrradstellplaetze AS Z
 6  FROM Station
 7  INNER JOIN Fahrrad
 8    ON Station.Stationsname = Fahrrad.Stationsname
 9  GROUP BY Fahrrad.Stationsname
10
11  UNION
12
13  SELECT Station.Stationsname,
14    Station.AnzahlFahrradstellplaetze,
15    0 AS X,
16    Station.AnzahlFahrradstellplaetze AS Y,
17    0 AS Z
18  FROM Station
19  WHERE Station.Stationsname NOT IN (
20    SELECT DISTINCT Fahrrad.Stationsname
21    FROM Fahrrad
22  )
```

Hinweis: Ein SQL-Konstrukt der Form `0 AS spaltenbezeichner` füllt jeden Datensatz in der Spalte `spaltenbezeichner` mit dem Wert `0`.

Analysieren und erläutern Sie den ersten Teil der SQL-Anweisung von Zeile 1 – 9.

Analysieren und erläutern Sie den zweiten Teil der SQL-Anweisung von Zeile 13 – 22.

Erläutern Sie im Sachzusammenhang, welche Information die gesamte SQL-Anweisung ermittelt.

(8 Punkte)

Name: _____

d) Damit die Kunden bei einer Fahrradstation schnell das passende Fahrrad finden, sind alle Kinderfahrräder grün, alle Damenfahrräder rot und alle Herrenfahrräder blau.

Das in Abbildung 2 gegebene Datenbankschema befindet sich bereits in der ersten Normalform.

Überführen Sie das in Abbildung 2 gegebene Datenbankschema in ein neues Datenbankschema, das der dritten Normalform entspricht.

Erläutern Sie die Änderungen, die zur Überführung in die dritte Normalform notwendig sind.

Beurteilen Sie, ob Ihr entwickeltes Datenbankschema noch geeignet ist, wenn z. B. durch einen Zukauf noch mehrere gelbe Damen- und Herrenfahrräder hinzukämen.

(10 Punkte)

e) Da häufiger Fahrräder gestohlen werden, bekommt das Unternehmen *Rent a Bike in a City* regelmäßig eine Liste mit Seriennummern von gestohlenen Fahrrädern von der Polizei zur Verfügung gestellt, die wiedergefunden wurden. Die Seriennummer ist markenübergreifend eindeutig. Das Programm wird erweitert, um schnell überprüfen zu können, welche der gestohlenen Fahrräder dem Unternehmen gehören.

Abbildung 3 zeigt einen Ausschnitt des Implementationsdiagramms des Informatiksystems.

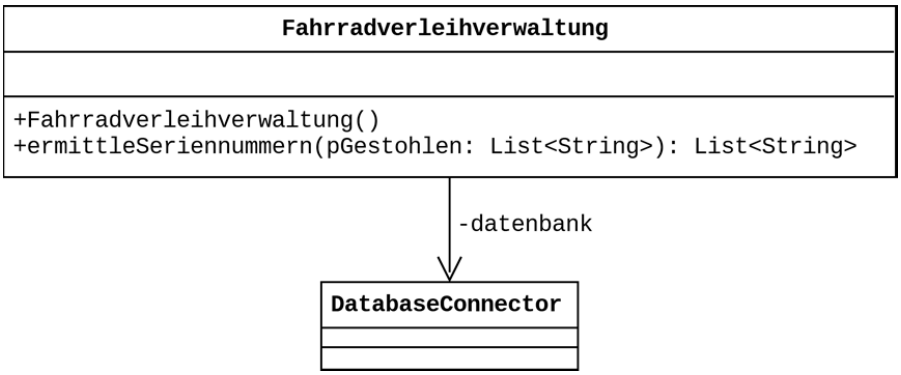

Abbildung 3: Teilmodellierung der Fahrradverleihverwaltung

Die Methode `ermittleSeriennummern` bekommt eine Liste mit Seriennummern der gestohlenen Fahrräder übergeben. Die Methode liefert eine neue Liste zurück, in der lediglich die Seriennummern der Fahrräder stehen, die zum eigenen Unternehmen gehören und gestohlen wurden.

Name: _____

Die Methode `ermittleSeriennummern` hat den folgenden Methodenkopf:

```
public List<String> ermittleSeriennummern(List<String>
                                                pGestohlen)
```

Entwerfen Sie einen Algorithmus für die beschriebene Methode `ermittleSeriennummern`.

Implementieren Sie die Methode `ermittleSeriennummern` *entsprechend Ihrem Algorithmus.*

Hinweis: Sie können davon ausgehen, dass die Datenbankverbindung bereits bei Objekterzeugung hergestellt wurde.

(10 Punkte)

Zugelassene Hilfsmittel:

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

Name: _____

Anlage

Beispieldaten zur Teilmodellierung aus Abbildung 2

Fahrrad				
Seriennr	**Typ**	**Baujahr**	**Farbe**	**Stationsname**
1	Kind	2018	grün	Bahnhof
2	Kind	2017	grün	See
3	Herren	2017	blau	*NULL*
4	Damen	2015	rot	Schule
5	Damen	2018	rot	Schwimmbad
6	Herren	2018	blau	Schwimmbad
7	Damen	2015	rot	See

Kunde			
Kundennr	**Vorname**	**Nachname**	**Handynr**
1	Ada	Lovelace	016012345678
2	John	von Neumann	016398765432
3	Alan	Turing	016001010101
4	Konrad	Zuse	016310101010

leihtAus			
Kundennr	**Seriennr**	**Ausleihzeitpunkt**	**Rueckgabezeitpunkt**
1	4	2016-06-16 15:00:00	2016-06-16 18:00:00
1	5	2017-04-13 10:30:00	2017-04-13 19:45:00
2	3	2018-12-24 08:00:00	2018-12-25 01:00:00
3	3	2019-04-16 12:08:00	*NULL*

GefahreneKilometer
7
12
20
NULL

Name: _____

Station			
Stationsname	**Strasse**	**Hausnr**	**AnzahlFahrradstellplaetze**
Bahnhof	Bahnhofsplatz	1	100
Park	Im Park	5	100
Schule	Schulstr.	3	40
Schwimmbad	Schwimmbadweg	1	50
See	Paradieser Weg	64	30

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse `DatabaseConnector`

Ein Objekt der Klasse **`DatabaseConnector`** ermöglicht die Abfrage und Manipulation einer MySQL-Datenbank.

Beim Erzeugen des Objekts wird eine Datenbankverbindung aufgebaut, so dass anschließend SQL-Anweisungen an diese Datenbank gerichtet werden können.

Dokumentation der Klasse `DatabaseConnector`

Konstruktor **`DatabaseConnector(String pIP, String pPort, String pDatabase, String pUsername, String pPassword)`**

Ein Objekt vom Typ `DatabaseConnector` wird erstellt, und eine Verbindung zur Datenbank wird aufgebaut. Mit den Parametern `pIP` und `pPort` werden die IP-Adresse und die Port-Nummer übergeben, unter denen die Datenbank mit Namen `pDatabase` zu erreichen ist. Mit den Parametern `pUsername` und `pPassword` werden Benutzername und Passwort für die Datenbank übergeben.

Auftrag **`void executeStatement(String pSQLStatement)`**

Der Auftrag schickt den im Parameter `pSQLStatement` enthaltenen SQL-Befehl an die Datenbank ab.
Handelt es sich bei `pSQLStatement` um einen SQL-Befehl, der eine Ergebnismenge liefert, so kann dieses Ergebnis anschließend mit der Methode `getCurrentQueryResult` abgerufen werden.

Anfrage **`QueryResult getCurrentQueryResult()`**

Die Anfrage liefert das Ergebnis des letzten mit der Methode `executeStatement` an die Datenbank geschickten SQL-Befehls als Objekt vom Typ `QueryResult` zurück.
Wurde bisher kein SQL-Befehl abgeschickt oder ergab der letzte Aufruf von `executeStatement` keine Ergebnismenge (z. B. bei einem INSERT-Befehl oder einem Syntaxfehler), so wird `null` geliefert.

Anfrage **`String getErrorMessage()`**

Die Anfrage liefert `null` oder eine Fehlermeldung, die sich jeweils auf die letzte zuvor ausgeführte Datenbankoperation bezieht.

Auftrag **`void close()`**

Die Datenbankverbindung wird geschlossen.

Name: _____

Die Klasse `QueryResult`

Ein Objekt der Klasse **`QueryResult`** stellt die Ergebnistabelle einer Datenbankanfrage mit Hilfe der Klasse `DatabaseConnector` dar. Objekte dieser Klasse werden nur von der Klasse `DatabaseConnector` erstellt. Die Klasse verfügt über keinen öffentlichen Konstruktor.

Dokumentation der Klasse `QueryResult`

Anfrage **`String[][] getData()`**

Die Anfrage liefert die Einträge der Ergebnistabelle als zweidimensionales Feld vom Typ `String`. Der erste Index des Feldes stellt die Zeile und der zweite die Spalte dar (d. h. `String[zeile][spalte]`).

Anfrage **`String[] getColumnNames()`**

Die Anfrage liefert die Bezeichner der Spalten der Ergebnistabelle als Feld vom Typ `String` zurück.

Anfrage **`String[] getColumnTypes()`**

Die Anfrage liefert die Typenbezeichnung der Spalten der Ergebnistabelle als Feld vom Typ `String` zurück. Die Bezeichnungen entsprechen den Angaben in der MySQL-Datenbank.

Anfrage **`int getRowCount()`**

Die Anfrage liefert die Anzahl der Zeilen der Ergebnistabelle als `int`.

Anfrage **`int getColumnCount()`**

Die Anfrage liefert die Anzahl der Spalten der Ergebnistabelle als `int`.

Name: _____

Die generische Klasse List<ContentType>

Objekte der generischen Klasse **`List`** verwalten beliebig viele, linear angeordnete Objekte vom Typ **`ContentType`**. Auf höchstens ein Listenobjekt, aktuelles Objekt genannt, kann jeweils zugegriffen werden. Wenn eine Liste leer ist, vollständig durchlaufen wurde oder das aktuelle Objekt am Ende der Liste gelöscht wurde, gibt es kein aktuelles Objekt. Das erste oder das letzte Objekt einer Liste können durch einen Auftrag zum aktuellen Objekt gemacht werden. Außerdem kann das dem aktuellen Objekt folgende Listenobjekt zum neuen aktuellen Objekt werden.

Das aktuelle Objekt kann gelesen, verändert oder gelöscht werden. Außerdem kann vor dem aktuellen Objekt ein Listenobjekt eingefügt werden.

Dokumentation der Klasse `List<ContentType>`

Konstruktor **`List()`**

Eine leere Liste wird erzeugt. Objekte, die in dieser Liste verwaltet werden, müssen vom Typ `ContentType` sein.

Anfrage **`boolean isEmpty()`**

Die Anfrage liefert den Wert `true`, wenn die Liste keine Objekte enthält, sonst liefert sie den Wert `false`.

Anfrage **`boolean hasAccess()`**

Die Anfrage liefert den Wert `true`, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert `false`.

Auftrag **`void next()`**

Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausführung des Auftrags kein aktuelles Objekt, d. h., `hasAccess()` liefert den Wert `false`.

Auftrag **`void toFirst()`**

Falls die Liste nicht leer ist, wird das erste Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

Auftrag **`void toLast()`**

Falls die Liste nicht leer ist, wird das letzte Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

Name: _____

Anfrage **`ContentType getContent()`**

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt zurückgegeben. Andernfalls (`hasAccess() == false`) gibt die Anfrage den Wert `null` zurück.

Auftrag **`void setContent(ContentType pContent)`**

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`) und `pContent` ungleich `null` ist, wird das aktuelle Objekt durch `pContent` ersetzt. Sonst bleibt die Liste unverändert.

Auftrag **`void append(ContentType pContent)`**

Ein neues Objekt `pContent` wird am Ende der Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Wenn die Liste leer ist, wird das Objekt `pContent` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt (`hasAccess() == false`).
Falls `pContent` gleich `null` ist, bleibt die Liste unverändert.

Auftrag **`void insert(ContentType pContent)`**

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird ein neues Objekt `pContent` vor dem aktuellen Objekt in die Liste eingefügt. Das aktuelle Objekt bleibt unverändert.
Falls die Liste leer ist und es somit kein aktuelles Objekt gibt (`hasAccess() == false`), wird `pContent` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt.
Falls es kein aktuelles Objekt gibt (`hasAccess() == false`) und die Liste nicht leer ist oder `pContent == null` ist, bleibt die Liste unverändert.

Auftrag **`void concat(List<ContentType> pList)`**

Die Liste `pList` wird an die Liste angehängt. Anschließend wird `pList` eine leere Liste. Das aktuelle Objekt bleibt unverändert. Falls es sich bei der Liste und `pList` um dasselbe Objekt handelt, `pList == null` oder eine leere Liste ist, bleibt die Liste unverändert.

Auftrag **`void remove()`**

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt gelöscht und das Objekt hinter dem gelöschten Objekt wird zum aktuellen Objekt. Wird das Objekt, das am Ende der Liste steht, gelöscht, gibt es kein aktuelles Objekt mehr (`hasAccess() == false`). Wenn die Liste leer ist oder es kein aktuelles Objekt gibt (`hasAccess() == false`), bleibt die Liste unverändert.

Unterlagen für die Lehrkraft

Abiturprüfung 2019

Informatik, Leistungskurs

1. Aufgabenart

Analyse, Modellierung und Abfrage relationaler Datenbanken

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2019

Die Aufgaben weisen vielfältige Bezüge zu den Kompetenzerwartungen und Inhaltsfeldern des Kernlehrplans bzw. zu den in den Vorgaben ausgewiesenen Fokussierungen auf.
Im Folgenden wird auf Bezüge von zentraler Bedeutung hingewiesen.

1. Inhaltsfelder und inhaltliche Schwerpunkte

Daten und ihre Strukturierung
- Datenbanken
- Objekte und Klassen
  - Lineare Strukturen

Algorithmen
- Analyse, Entwurf und Implementierung von Algorithmen

Formale Sprachen und Automaten
- Syntax und Semantik einer Programmiersprache
  - SQL
  - Java

2. Medien/Materialien
- entfällt

5. Zugelassene Hilfsmittel

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Modelllösungen

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Die in Abbildung 1 gegebene Teilmodellierung besteht aus drei Entitätstypen und zwei Beziehungstypen.

Im Entitätstyp `Kunde` wird als Primärschlüssel `Kundennr` verwendet.
Im Entitätstyp `Fahrrad` wird als Primärschlüssel `Seriennr` verwendet.
Im Entitätstyp `Station` wird als Primärschlüssel `Stationsname` verwendet.

Vom Entitätstyp `Kunde` werden die weiteren Attribute `Vorname`, `Nachname` und `Handynr` verwaltet.
Vom Entitätstyp `Fahrrad` werden die weiteren Attribute `Typ`, `Baujahr` und `Farbe` verwaltet.
Vom Entitätstyp `Station` werden die weiteren Attribute `Strasse`, `Hausnr` und `AnzahlFahrradstellplaetze` verwaltet.

Der Beziehungstyp `leihtAus` modelliert, welche Kundin oder welcher Kunde welches Fahrrad ausgeliehen hat. Zusätzlich verwaltet der Beziehungstyp die weiteren Attribute `Ausleihzeitpunkt`, `Rueckgabezeitpunkt` und `GefahreneKilometer`.

Der Beziehungstyp `ausleihbarBei` modelliert, an welcher Station ein zur Verfügung stehendes Fahrrad steht.

Kardinalitäten beim Beziehungstyp `leihtAus`:
Es handelt sich um einen `n:m`-Beziehungstyp. Ein Kunde kann mehrere Fahrräder ausleihen und ein Fahrrad kann von mehreren Kunden zu unterschiedlichen Zeitpunkten ausgeliehen werden.

Kardinalitäten beim Beziehungstyp `ausleihbarBei`:
Es handelt sich um einen `1:n`-Beziehungstyp. An einer Station können mehrere Fahrräder stehen. Ein Fahrrad steht an einer Station.
Im Datenbankschema wird diese Beziehung durch das Fremdschlüsselattribut `Stationsname` im Entitätstyp `Fahrrad` realisiert.

Der Primärschlüssel für den Beziehungstyp `leihtAus` setzt sich aus der `Kundennr` (Fremdschlüssel von `Kunde`) und der `Seriennr` (Fremdschlüssel von `Fahrrad`) und dem `Ausleihzeitpunkt` zusammen.

Weder die Fremdschlüssel noch der `Ausleihzeitpunkt` genügen, um einen Ausleihvorgang eindeutig bestimmen zu können, da zu einem Zeitpunkt mehrere Kunden unterschiedliche Fahrräder ausleihen könnten und ein Kunde mehrfach ein identisches Fahrrad ausleihen kann.

Teilaufgabe b)

Die folgenden SQL-Anweisungen realisieren die Anfragen:

(i)
```
SELECT Seriennr
FROM Fahrrad
WHERE Farbe = 'rot' AND Baujahr < 2017
ORDER BY Seriennr ASC
```

(ii)
```
SELECT Kunde.Vorname, Kunde.Nachname, Kunde.Handynr
FROM Kunde
  INNER JOIN leihtAus
    ON Kunde.Kundennr = leihtAus.Kundennr
WHERE leihtAus.GefahreneKilometer > 10
```

(iii)
```
SELECT Fahrrad.Seriennr, Fahrrad.Baujahr,
  SUM(leihtAus.GefahreneKilometer) AS Kilometersumme,
  COUNT(*) AS Verleihvorgaenge
FROM Fahrrad
  INNER JOIN leihtAus
    ON Fahrrad.Seriennr = leihtAus.Seriennr
GROUP BY Fahrrad.Seriennr
ORDER BY Fahrrad.Baujahr ASC, Kilometersumme DESC
```

Teilaufgabe c)

Die erste SQL-Anweisung (vgl. Zeilen 1 – 9) beinhaltet einen INNER JOIN von der Tabelle `Station` auf `Fahrrad` über den `Stationsnamen` (vgl. Zeilen 7 – 8). Somit tauchen in dieser Teilergebnismenge nur die Stationen auf, an denen mindestens ein Fahrrad steht. In Zeile 9 werden die Ergebnisse nach dem Stationsnamen gruppiert. Für jede Gruppe werden folgende Informationen ermittelt (vgl. Zeilen 1 – 5):

- Stationsname
- Anzahl der Fahrradstellplätze
- Anzahl der belegten Fahrradstellplätze
- Anzahl der noch freien Fahrradstellplätze
- (Belegungs-)Quote, wieviel Prozent der Fahrradstellplätze belegt sind

Die zweite SQL-Anweisung (vgl. Zeilen 13 – 22) beinhaltet eine Unterabfrage (vgl. Zeilen 19 – 22). Diese Unterabfrage liefert die Namen der Stationen, an denen (mindestens) ein Fahrrad steht.
Mithilfe der zweiten SQL-Anweisung werden folgende Informationen von den Stationen ermittelt, an denen kein Fahrrad steht (vgl. NOT IN in Zeile 19) (vgl. Zeilen 13 – 17):

- Stationsname
- Anzahl der Fahrradstellplätze
- Die Anzahl der belegten Fahrradstellplätze wird auf 0 gesetzt.
- Die Anzahl der noch freien Fahrradstellplätze wird auf die Anzahl der Fahrradstellplätze gesetzt.
- Die (Belegungs-)Quote wird auf 0 gesetzt.

Für die gesamte SQL-Anweisung werden zwei SQL-Anweisungen mit dem UNION-Befehl miteinander verknüpft.
Im Sachzusammenhang liefert die gesamte SQL-Anweisung eine Übersicht über alle Stationen mit den Informationen über den Stationsnamen, der Anzahl der Fahrradstellplätze, die Anzahl der noch freien Fahrradstellplätze und die (Belegungs-)Quote.

Teilaufgabe d)

Kunde(Kundennr, Vorname, Nachname, Handynr)

leihtAus(↑Kundennr, ↑Seriennr, Ausleihzeitpunkt,
Rueckgabezeitpunkt, GefahreneKilometer)

Fahrrad(Seriennr, ↑Typname, Baujahr, ↑Stationsname)

Typ(Typname, Farbe)

Station(Stationsname, Strasse, Hausnr, AnzahlFahrradstellplaetze)

Ein Datenbankschema ist in der 2. Normalform, wenn es in der 1. Normalform ist und zusätzlich jedes Attribut, das nicht selbst zum Schlüssel gehört, nur von allen Schlüsselattributen funktional abhängig ist und nicht bereits von einem Teil der Schlüsselattribute.
Ein Verstoß gegen die zweite Normalform könnte nur in einer Relation mit einem kombinierten Primärschlüssel auftreten. Da jedes Nichtschlüsselattribut der Relation `leihtAus` vom gesamten Schlüssel abhängt, ist das Datenbankschema auch in der zweiten Normalform.

Ein Datenbankschema ist in der 3. Normalform, wenn es in der 2. Normalform ist und es zusätzlich kein Nichtschlüsselattribut gibt, das transitiv von einem Schlüsselattribut abhängig ist. Es darf also keine funktionalen Abhängigkeiten von Attributen geben, die selbst nicht zum Schlüssel gehören.
Der Entitätstyp `Fahrrad` verstößt gegen die dritte Normalform, weil das Attribut `Farbe` funktional vom Attribut `Typ` abhängt. Dieser Verstoß wird gelöst, indem der neue Entitätstyp `Typ` mit dem Primärschlüssel `Typname` und dem Attribut `Farbe` eingerichtet wird. Im Entitätstyp `Fahrrad` werden die Attribute `Typ` und `Farbe` durch das Fremdschlüsselattribut `Typname` ersetzt.

Wenn durch einen Zukauf noch mehrere gelbe Damen- und Herrenfahrräder hinzukämen, wäre das entwickelte Datenbankschema wenig geeignet, weil im Entitätstyp `Typ` das Attribut `Farbe` nicht mehr vom Primärschlüssel `Typname` abhängen würde.

Teilaufgabe e)

Eine neue leere Ausgabeliste wird erzeugt.

Die Liste mit den Seriennummern der gestohlenen Fahrräder, die im Parameter übergeben wurde, muss einmal komplett durchlaufen werden:

- Für jede Seriennummer muss überprüft werden, ob diese aus dem Bestand des Unternehmens stammt. Falls ja, dann wird diese Seriennummer an die Ausgabeliste angehängt.

Die Ausgabeliste wird zurückgeliefert.

```
public List<String> ermittleSeriennummern(List<String>
                                                    pGestohlen) {
  List<String> ergebnis = new List<String>();
  pGestohlen.toFirst();
  while (pGestohlen.hasAccess()) {
    String aktuelleSeriennr = pGestohlen.getContent();
    datenbank.executeStatement("SELECT Seriennr " +
                               "FROM Fahrrad " +
                               "WHERE Seriennr = '" +
                               aktuelleSeriennr + "'");
    QueryResult qr = datenbank.getCurrentQueryResult();
    if (qr != null && qr.getRowCount() > 0) {
      ergebnis.append(aktuelleSeriennr);
    }
    pGestohlen.next();
  }
  return ergebnis;
}
```

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	beschreibt die gegebene Teilmodellierung.	4			
2	ermittelt die Kardinalitäten für das Entity-Relationship-Modell, so dass diese der Umsetzung im Datenbankschema entsprechen.	4			
3	begründet anhand des Kontextes, warum die Attribute `Kundennr` und `Seriennr` gemeinsam mit dem Attribut `Ausleihzeitpunkt` in der Relation `leihtAus` den kombinierten Primärschlüssel bilden.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe a)	**12**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft für die erste Anfrage `i` eine SQL-Anweisung.	3			
2	entwirft für die zweite Anfrage `ii` eine SQL-Anweisung.	3			
3	entwirft für die dritte Anfrage `iii` eine SQL-Anweisung.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe b)	**10**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert und erläutert den ersten Teil der SQL-Anweisung.	3			
2	analysiert und erläutert den zweiten Teil der SQL-Anweisung.	3			
3	erläutert im Sachzusammenhang, welche Information die gesamte SQL-Anweisung ermittelt.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe c)	**8**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	überführt das gegebene Datenbankschema in ein neues Datenbankschema, das der dritten Normalform entspricht.	4			
2	erläutert die Änderungen, die zur Überführung in die dritte Normalform notwendig sind.	3			
3	beurteilt, ob das entwickelte Datenbankschema noch geeignet ist, wenn z. B. durch einen Zukauf noch mehrere gelbe Damen- und Herrenfahrräder hinzukämen.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe d)	**10**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft einen Algorithmus für die beschriebene Methode `ermittleSeriennummern`.	4			
2	implementiert die Methode entsprechend dem Algorithmus.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe e)	**10**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	**EK**	**ZK**	**DK**
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der dritten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**150**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

Berechnung der Endnote nach Anlage 4 der Abiturverfügung auf der Grundlage von § 34 APO-GOSt

Die Klausur wird abschließend mit der Note _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	150 – 143
sehr gut	14	142 – 135
sehr gut minus	13	134 – 128
gut plus	12	127 – 120
gut	11	119 – 113
gut minus	10	112 – 105
befriedigend plus	9	104 – 98
befriedigend	8	97 – 90
befriedigend minus	7	89 – 83
ausreichend plus	6	82 – 75
ausreichend	5	74 – 68
ausreichend minus	4	67 – 60
mangelhaft plus	3	59 – 50
mangelhaft	2	49 – 41
mangelhaft minus	1	40 – 30
ungenügend	0	29 – 0