

Sie Aufgaben 1 bis 4 beziehen sich auf die folgende Ausgangssituation:

In Kiel soll das Nahverkehrssystem (KVAG) um eine neue Stadtbahn erweitert werden. In Zusammenhang mit dieser Erweiterung sollen die IT-Systeme der KVAG erweitert und erneuert werden. Dazu werden verschiedene Teilprojekte festgelegt. Diese betreffen unter anderem die Fahrplanauskunft, die Buchungssysteme, die Anzeigen und die Vernetzung. Alle Teilprojekte sollen auch hinsichtlich der Anforderungen von Datenschutz und Datensicherheit untersucht werden.

1. Aufgabe (25 Punkte)

Die KVAG möchte ihre Fahrpläne besser an den tatsächlich vorhandenen Fahrzeiten ausrichten. Dazu wurden die Abfahrtszeiten einer Bahn an den einzelnen Haltestellen für eine Fahrt einer Bahnlinie an mehreren Tagen gemessen.

Für jeden Tag ist festgehalten, wann die Bahn die nachfolgenden Haltestellen verlassen hat (siehe Array `zeiten`).



Es existiert eine Klasse **Abfahrtszeit**, mit deren Attributen ein Datum, eine Haltestellennummer, die planmäßige Abfahrtszeit (in Minuten seit Mitternacht) und die tatsächliche Abfahrtszeit (ebenfalls in Minuten) gespeichert wird. Für die Endstation sind hier die planmäßigen bzw. tatsächlichen Ankunftszeiten gespeichert.

Klasse **Abfahrtszeit**

Abfahrtszeit	
- datum : Date	
- haltestellenNr : Integer	
- planAbfahrt : Integer	// planmäßige Abfahrtszeit in Minuten seit Mitternacht
- istAbfahrt : Integer	// tatsächliche Abfahrtszeit in Minuten seit Mitternacht

Für jedes Attribut ist eine öffentliche Get-Methode vorhanden.

In dem eindimensionalen Array `zeiten` vom Typ **Abfahrtszeit** sind die Daten des untersuchten Zeitraums gespeichert. In einer Zeile sind die Daten eines **Abfahrtszeit**-Objektes aufgeführt.

Auflistung der Daten des Arrays `zeiten`

Datum	Haltestellen Nummer	Planmäßige Abfahrt	Tatsächliche Abfahrt
1.9.2024	0	480	480
1.9.2024	1	483	483
1.9.2024	2	485	486
...			
1.9.2024	15	579	583
2.9.2024	0	480	480
2.9.2024	1	483	484
...			
2.9.2024	15	579	582
...			

Die Fahrzeit auf einer Strecke ist die Differenz zweier Abfahrtszeiten aufeinanderfolgender Haltestellen.

Eine Verspätung liegt vor, wenn die Fahrzeit auf einer Strecke mehr als zwei Minuten länger war als die planmäßige Fahrzeit.

Es soll ein Algorithmus für eine Funktion

`ermittleFahrzeiten(Abfahrtszeit[] zeiten) : Integer[]`

entwickelt werden, der für jede der 15 Strecken die Anzahl der Verspätungen ermittelt.

Die ermittelten Häufigkeiten sollen in einem eindimensionalen Array vom Typ **Integer** gespeichert und von der Funktion zurückgegeben werden.

Ergebnis-Array (Beispiel):

Korrekturrand

Index	Anzahl Fahrzeiten > 2	Kommentar
[0]	0	Anzahl Verspätungen (im Beispiel 0) auf der Strecke 0
[1]	1	... auf der Strecke 1
[2]	4	... auf der Strecke 2
...	...	...
[14]	3	... auf der Strecke 14

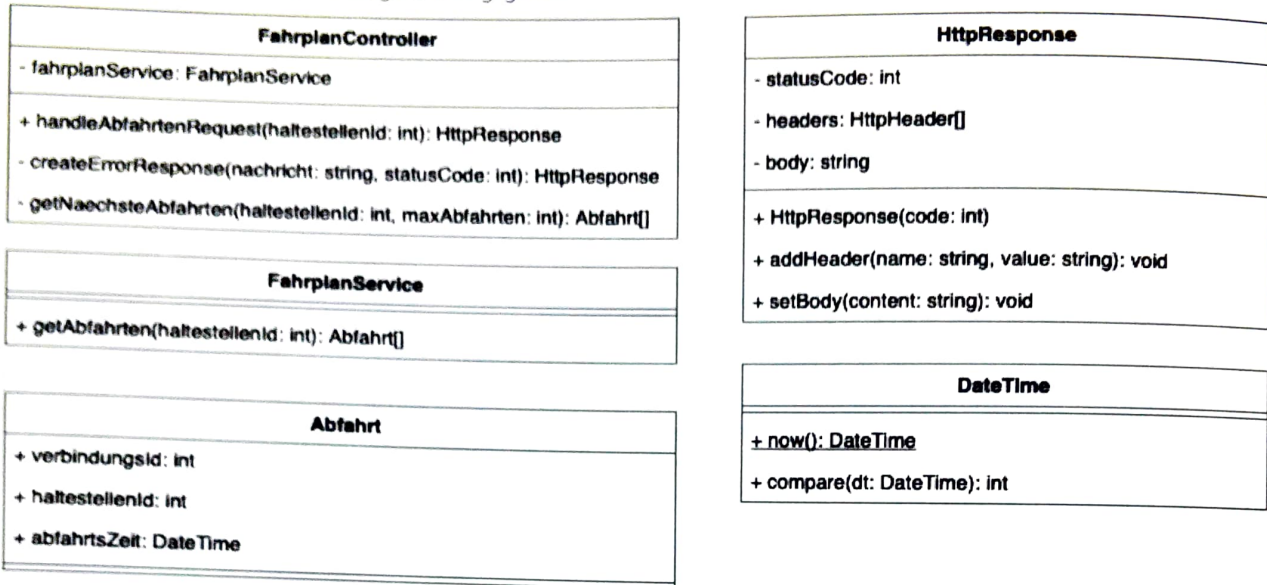
Stellen Sie den Algorithmus in Pseudocode dar.

`ermittle_fahrzeiten(Abfahrtszeit[] zeiten) : Integer[]`

2. Aufgabe (25 Punkte)

Die Fahrgäste sollen die Möglichkeit haben, in der App die aktuellen Abfahrtszeiten und Verspätungen einsehen zu können. Diese Daten sollen über eine API abgerufen werden. Ihre Aufgabe ist es, den API-Endpunkt zu implementieren.

Folgender Auszug aus dem Klassendiagramm ist gegeben:



Die Methode `compare` aus der Klasse `DateTime` gibt folgende Rückgabewerte zurück:

- Wenn der Datumswert des aktuellen Objekts vor dem des übergebenen Objekts ist: -1
- Wenn beide Datumswerte gleich sind: 0
- Wenn der Datumswert des aktuellen Objekts nach dem des übergebenen Objekts ist: 1

- a) In der Klasse `FahrplanController` soll zunächst die Methode `createErrorResponse(nachricht: string, statusCode: int): HttpResponse` implementiert werden.

Die Nachricht und der Status-Code werden übergeben.

Die erzeugte HTTP-Antwort soll wie folgt aussehen.

HTTP < der übergebene Status-Code >

Content-Type: text/plain

Content-Length: < Anzahl der Bytes des Response Bodys >

< Response Body: die übergebene Nachricht >

Implementieren Sie die Methode `createErrorResponse`.

10 Punkte

```
private createErrorResponse(nachricht: string, statusCode: int): HttpResponse
```

b) Die App soll die nächsten Abfahrten anzeigen.

Dazu soll in der Klasse **FahrplanController** die private Methode **getNaechsteAbfahrten(haltestellenId: int, maxAbfahrten: int): Abfahrt[]** erstellt werden.

Diese Methode soll die nächsten Abfahrten ab dem aktuellen Zeitpunkt zurückgeben. **maxAbfahrten** gibt an, wie viele Abfahrten maximal zurückgegeben werden sollen.

Die Methode soll das **Abfahrt** Array mit den ermittelten Abfahrten zurückgeben.

Mithilfe der Methode **FahrplanService.getAbfahrten** können die Abfahrten von einer Haltestelle abgerufen werden. **getAbfahrten** liefert ein aufsteigend nach Abfahrtszeit sortiertes Array von Abfahrten zurück.

Implementieren Sie die Methode **getNaechsteAbfahrten**.

15 Punkte

```
private getNaechsteAbfahrten (haltestellenId: int, maxAbfahrten: int):  
    Abfahrt[]
```

3. Aufgabe (29 Punkte)

a) Es sollen stichprobenartig Kontrollen von Mitarbeitern der Stadtbahn auf den Fahrten durchgeführt werden. Diese Daten sollen in einer relationalen Datenbank verwaltet werden. Für einen ersten Entwurf eines relationalen Datenmodells liegen Ihnen folgende Informationen vor:

- Jede Kontrolle findet immer auf einer bestimmten Fahrt statt.
- Auf einer Fahrt können auch mehrere Kontrollen durchgeführt werden.
- Zu jeder Kontrolle gehören mindestens zwei Mitarbeiter.
- Bei jeder Kontrolle kann es zu verschiedenen Arten von Vorkommnissen kommen.
- Jedem Vorkommnis wird die Art der Maßnahme zugordnet, welche ergriffen wurde.

Erstellen Sie ein relationales Datenmodell, welches der 3. Normalform genügt. Tragen Sie in die Tabellen alle notwendigen Primär- und Fremdschlüssel ein und kennzeichnen Sie diese mit PK bzw. FK. Weitere Attribute können vernachlässigt werden. Tragen Sie alle Beziehungen zwischen den Tabellen mit den entsprechenden Kardinalitäten ein.

16 Punkte

b) Im Vorfeld der Einführung des neuen Systems wurden bereits Daten manuell erfasst. Jetzt wird in Betracht gezogen, diese Daten ins System zu importieren. Folgender Auszug aus einer Tabelle mit den Daten wurde erstellt. Der Datenumfang beträgt über 1.000 Zeilen:

Datum	Mitarbeiter	Fahrt	Vorkommnis	Maßnahme
1.7.	Müller, Meier	Linie 13	Kein Ticket	erhöhtes Beförderungsentgelt
3.8.23	Schmitt, Müller	1	Ohne Fahrkarte	60 EUR Strafe
31.4.	Mueller, Schmitt	L 4	Randalierer	Anzeige
4.7.	Müller; Meier	Linie 6	Kein Ticket	60 EUR
2023-4-13	Schmitt,	Linie 17	Ticket nicht lesbar	Verwarnung
23.6.23	Schmitt; Hans	Linie 3	Kein Fahrschein	60 EUR erhöhtes Entgelt

ba) Die vorliegenden Daten sollen beurteilt werden.

Beschreiben Sie drei Probleme bezüglich der Qualität der Daten und nennen Sie jeweils eine Folge, die sich durch den Import der mangelhaften Daten ergeben würde. 9 Punkte

bb) Zeigen Sie einen möglichen Weg auf, wie die Daten dennoch importiert werden können. 2 Punkte

bc) Beurteilen Sie abschließend die Importidee und argumentieren Sie für oder gegen den Import der Daten. 2 Punkte

4. Aufgabe (21 Punkte)

Die Tabellenauszüge (siehe perforierte Anlage) stehen für die im Folgenden beschriebenen SQL-Anweisungen zur Verfügung:

- a) Erstellen Sie eine SQL-Anweisung, mit der Sie alle aktiven Haltestellen der Linie 250 erhalten.
- 3 Punkte

Ergebnistabelle:

Name der Haltestelle
Am Faulbach
Von-Lohe-Str.
Windmühlenstr.
Ebertplatz

- b) Erstellen Sie eine SQL-Anweisung, mit der Sie alle Abfahrtszeiten der Haltestelle ‚Am Faulbach‘ absteigend nach Datum und Uhrzeit sortiert erhalten.
- 6 Punkte

Ergebnistabelle:

HSt_Name	HSt_Linie	HStP_Abfahrt_Plan
Am Faulbach	250	2024-12-01 12:12:00
Am Faulbach	250	2024-12-01 11:12:00
Am Faulbach	250	2024-12-01 10:03:00

Dieses Blatt kann an der Perforation aus dem Aufgabensatz herausgetrennt werden!

Tabelle: Haltestelle

HSt_IdKey	HSt_Name	HSt_Aktiv	HSt_Linie
1	Halfengasse	1	124
2	Nesselrodestr.	1	124
3	Amsterdamer Gürtel	1	124
4	Xantener Str.	1	124
5	Worringer Str.	1	124
6	Am Faulbach	1	250
7	Von-Lohe-Str.	1	250
8	Schützenhofstr.	0	250
9	Windmühlenstr.	1	250
10	Reichensberger Platz	1	124
11	Ebertplatz	1	250
12	Ebertplatz	1	140
... weitere			

Tabelle: Haltestelle_Plan (Planzeiten der täglich geplanten Ankünfte und Abfahrten)

HStP_IdKey	HStP_HStIdKey	HStP_Ankunft_Plan	HStP_Abfahrt_Plan
1	1	2024-12-01 10:01:00	2024-12-01 10:01:00
2	2	2024-12-01 10:04:00	2024-12-01 10:04:00
3	3	2024-12-01 10:08:00	2024-12-01 10:08:00
4	4	2024-12-01 10:09:00	2024-12-01 10:09:00
5	5	2024-12-01 10:13:00	2024-12-01 10:18:00
6	10	2024-12-01 10:22:00	2024-12-01 10:22:00
7	6	2024-12-01 10:03:00	2024-12-01 10:03:00
8	7	2024-12-01 10:06:00	2024-12-01 10:06:00
9	9	2024-12-01 10:11:00	2024-12-01 10:11:00
10	11	2024-12-01 10:12:00	2024-12-01 10:12:00
11	6	2024-12-01 11:12:00	2024-12-01 11:12:00
12	6	2024-12-01 12:12:00	2024-12-01 12:12:00
...weitere			

Tabelle: Haltestelle_Zeiten (Istzeiten der täglichen Ankünfte und Abfahrten)

HStZ_IdKey	HStZ_HStIdKey	HStZ_HStPidKey	HStZ_Ankunft	HSt_Abfahrt
1	1	1	2024-12-01 10:01:00	2024-12-01 10:01:00
2	2	2	2024-12-01 10:05:00	2024-12-01 10:05:00
3	3	3	2024-12-01 10:09:00	2024-12-01 10:09:00
4	4	4	2024-12-01 10:10:00	2024-12-01 10:10:00
5	5	5	2024-12-01 10:15:00	2024-12-01 10:18:00
6	10	6	2024-12-01 10:22:00	2024-12-01 10:22:00
7	6	7	2024-12-01 10:03:00	2024-12-01 10:03:00
8	7	8	2024-12-01 10:06:00	2024-12-01 10:06:00
9	9	9	2024-12-01 10:11:00	2024-12-01 10:11:00
10	11	10	2024-12-01 10:12:00	2024-12-01 10:12:00
...weitere				

c) Erstellen Sie eine SQL-Anweisung, mit der Sie alle Verspätungen aller Haltestellen erhalten.
Ergebnistabelle:

12 Punkte Korrekturrand

HSt_Name	HSt_Linie	HStP_Ankunft_Plan	HStZ_Ankunft	Verspätung in Minuten
Nesselrodestr.	124	2024-12-01 10:04:00	2024-12-01 10:05:00	1
Amsterdamer Gürtel	124	2024-12-01 10:08:00	2024-12-01 10:09:00	1
Xantener Str.	124	2024-12-01 10:09:00	2024-12-01 10:10:00	1
Worringer Str.	124	2024-12-01 10:13:00	2024-12-01 10:15:00	2

PRÜFUNGSZEIT – NICHT BESTANDTEIL DER PRÜFUNG!

Wie beurteilen Sie nach der Bearbeitung der Aufgaben die zur Verfügung stehende Prüfungszeit?

- ☐ 1 Sie hätte kürzer sein können.
- ☐ 2 Sie war angemessen.
- ☐ 3 Sie hätte länger sein müssen.

