Abschlussprüfung Winter 2024/25
Lösungshinweise

Fachinformatiker/Fachinformatikerin Anwendungsentwicklung 1201

2 Entwicklung und Umsetzung von Algorithmen

Teil 2 der Abschlussprüfung

Allgemeine Korrekturhinweise

Die Lösungs- und Bewertungshinweise zu den einzelnen Handlungsschritten sind als Korrekturhilfen zu verstehen und erheben nicht in jedem Fall Anspruch auf Vollständigkeit und Ausschließlichkeit. Neben hier beispielhaft angeführten Lösungsmöglichkeiten sind auch andere sach- und fachgerechte Lösungsalternativen bzw. Darstellungsformen mit der vorgesehenen Punktzahl zu bewerten. Der Bewertungsspielraum des Korrektors (z. B. hinsichtlich der Berücksichtigung regionaler oder branchenspezifischer Gegebenheiten) bleibt unberührt.

Zu beachten ist die unterschiedliche Dimension der Aufgabenstellung (nennen – erklären – beschreiben – erläutern usw.).

Für die Bewertung gilt folgender Punkte-Noten-Schlüssel:

Note 1	=		100 – 92 Punkte	Note 2	=	unter	92 – 81 Punkte
Note 3	=	unter	81 – 67 Punkte	Note 4	=	unter	67 – 50 Punkte
Note 5	=	unter	50 – 30 Punkte	Note 6	=	unter	30 – 0 Punkte

1. Aufgabe (25 Punkte)

```
ermittle_fahrzeiten(Abfahrtszeit[] zeiten) : Integer[]
     verspaetungen = new Integer[15]
     für (i = 1; i < zeiten.length; i++)
          wenn zeiten[i].getDatum() == zeiten[i-1].getDatum()
               z1 = zeiten[i].getPlanAbfahrt() - zeiten[i-1].getPlanAbfahrt()
               z2 = zeiten[i].getIstAbfahrt() - zeiten[i-1].getIstAbfahrt()
               wenn z2-z1 > 2
                    verspaetungen[zeiten[i].getHaltestelleNr()-1]++
               ende wenn
          ende wenn
     ende für
     Rückgabe verspaetungen
ende ermittle_fahrzeiten
```

2. Aufgabe (25 Punkte)

a) 10 Punkte

```
response = new HttpResponse(statusCode)
response.addHeader(„Content-Type“, „text/plain“)
response.addHeader(„Content-Length“, String(Length(nachricht)))
response.setBody(nachricht)

return response
```

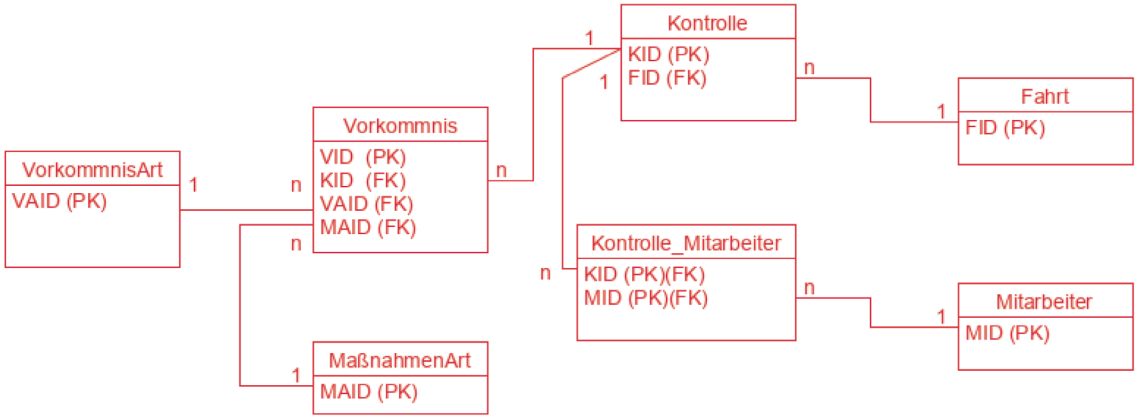
b) 15 Punkte

```
naechsteAbfahrten = new Abfahrt[maxAbfahrten]
naechsteAbfahrtenZaehler = 0

jetzt = DateTime.now()

abfahrten = this.fahrplanService.getAbfahrten(haltestellenId)

for (i=0; i < abfahrten.length && naechsteAbfahrtenZaehler < maxAbfahrten; i++) {

    abfahrt = abfahrten[i]

    if (jetzt.compare(abfahrt.abfahrtsZeit) >= 0) {
        naechsteAbfahrten[naechsteAbfahrtenZaehler] = abfahrt
        naechsteAbfahrtenZaehler++
    }
}

return naechsteAbfahrten
```

Der Datumsvergleich ist mit der Methode `DateTime.compare` durchzuführen.

3. Aufgabe (29 Punkte)

a) 16 Punkte
Pro Tabelle mit Primärschlüssel 1 Punkt
Pro Beziehung mit Kardinalität 1 Punkt
Pro Fremdschlüssel 0,5 Punkte



ba) 9 Punkte
Vorgeschlagene Lösungsmöglichkeiten:
- Uneinheitliche Formatierung (Datum 2023-4-13, 31.7.23)
  - Beim Importieren eines ungültigen Datums bzw. verschiedener Formate wird das Importprogramm abbrechen, oder wesentlich aufwendiger zu erstellen.
- Daten nicht eindeutig und Mängel in den Daten (Müller <> Mueller, 31.4.), generell Problem bei Mitarbeiternamen statt IDs.
  - Mitarbeiter werden u. U. doppelt geführt (Müller – Mueller) oder falsch zugewiesen, da Namen nicht eindeutig sind.
- Verschiedene Texte für gleiche Vorkommnisse oder Maßnahmen
  - Auswertungen werden erschwert (z. B. „wie oft kein Ticket" – aufgrund verschiedener Texte)
- Fehlende Daten: 2023-4-13 fehlt der Name eines Mitarbeiters
  - Mitarbeiter können nicht zugeordnet werden und die Nachvollziehbarkeit der Maßnahme ist nicht mehr gegeben.

bb) 2 Punkte
- Die Daten manuell verbessern.
- Das Importprogramm mit allen nötigen Fehlerbehandlungen erstellen.

bc) 2 Punkte
Möglich entweder:
Beides ist sehr aufwendig, sodass der Import aufgrund der Datenqualität sich nicht lohnt.

Oder:
Wenn die erfassten Daten weiterhin wichtig sind, so muss der größere Aufwand zum Import in Kauf genommen werden.

Andere Antworten mit passenden Begründungen können auch als richtig gewertet werden.

4. Aufgabe (21 Punkte)

a) 3 Punkte
```
SELECT HSt_Name AS [Name der Haltestelle]
FROM Haltestelle
WHERE HSt_Aktiv = 1 AND HSt_Linie = 250;
```

b) 6 Punkte
```
SELECT H.HSt_Name, H.HSt_Linie, HP.HStP_Abfahrt_Plan
FROM Haltestelle AS H
INNER JOIN Haltestelle_Plan AS HP ON H.HSt_IdKey = HP.HStP_HStIdKey
WHERE HSt_Name = ‚Am Faulbach'
ORDER BY HStP_Abfahrt_Plan DESC;
```

Lösungen mit LEFT JOIN sind ebenfalls als richtig zu werten.

c) 12 Punkte
```
SELECT H.HSt_Name, H.HSt_Linie, HP.HStP_Ankunft_Plan, HZ.HStZ_Ankunft,
    DATEDIFF(minute, HP.HStP_Ankunft_Plan, HZ.HStZ_Ankunft) AS [Verspätung in Minuten]
FROM Haltestelle AS H
INNER JOIN Haltestelle_Plan AS HP ON H.HSt_IdKey = HP.HStP_HStIdKey
INNER JOIN Haltestelle_Zeiten AS HZ ON HP.HStP_IdKey = HZ.HStZ_HStPIdKey
WHERE HP.HStP_Ankunft_Plan < HZ.HStZ_Ankunft;
```