Abschlussprüfung Winter 2017

Lösungshinweise

Fachinformatiker/Fachinformatikerin Anwendungsentwicklung 1196

1 Ganzheitliche Aufgabe I Fachqualifikationen

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.). Wird eine bestimmte Anzahl verlangt (z. B. „Nennen Sie fünf Merkmale ..."), so ist bei Aufzählung von fünf richtigen Merkmalen die volle vorgesehene Punktzahl zu geben, auch wenn im Lösungshinweis mehr als fünf Merkmale genannt sind. Bei Angabe von Teilpunkten in den Lösungshinweisen sind diese auch für richtig erbrachte Teilleistungen zu geben.

In den Fällen, in denen vom Prüfungsteilnehmer

- keiner der fünf Handlungsschritte ausdrücklich als „nicht bearbeitet" gekennzeichnet wurde,
- der 5. Handlungsschritt bearbeitet wurde,
- einer der Handlungsschritte 1 bis 4 deutlich erkennbar nicht bearbeitet wurde,

ist der tatsächlich nicht bearbeitete Handlungsschritt von der Bewertung auszuschließen.

Ein weiterer Punktabzug für den bearbeiteten 5. Handlungsschritt soll in diesen Fällen allein wegen des Verstoßes gegen die Formvorschrift nicht erfolgen!

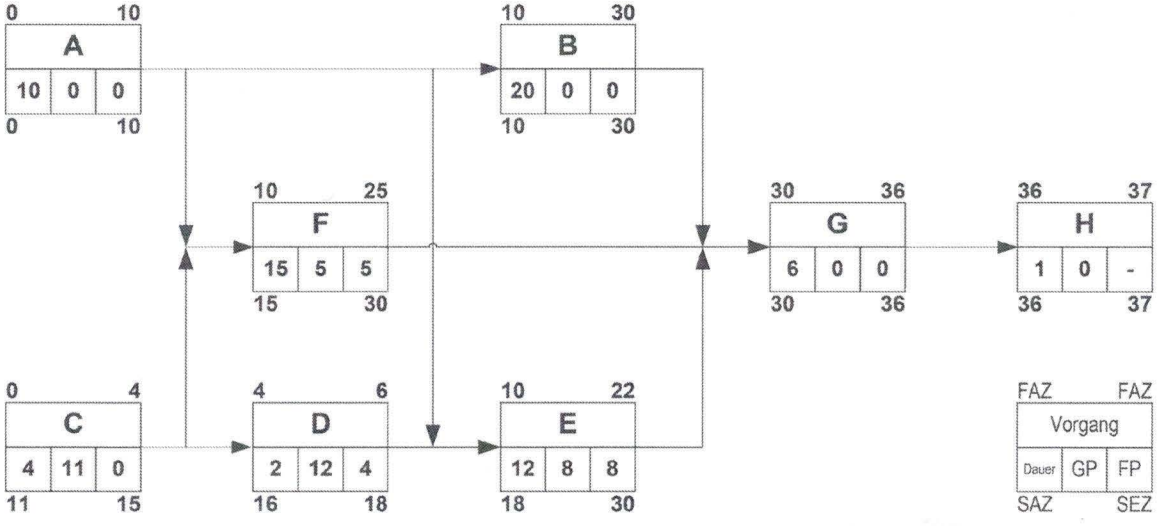
Für die Bewertung gilt folgender Punkte-Noten-Schlüssel:

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. Handlungsschritt (25 Punkte)

a) 18 Punkte
Punkteverteilung
16 Punkte, 8 x 2 Punkte je Knoten
2 Punkte, Markierung des kritischen Pfads

Vorgang	Dauer	Vorgänger
A	10	-
B	20	A
C	4	-
D	2	C
E	12	A, D
F	15	A, C
G	6	B, E, F
H	1	G



Kritischer Pfad: A – B – G – H

b) 3 Punkte
Das Projektende verzögert sich um zwei Tage (FEZ = 39), weil der Vorgang B auf dem kritischen Pfad liegt.

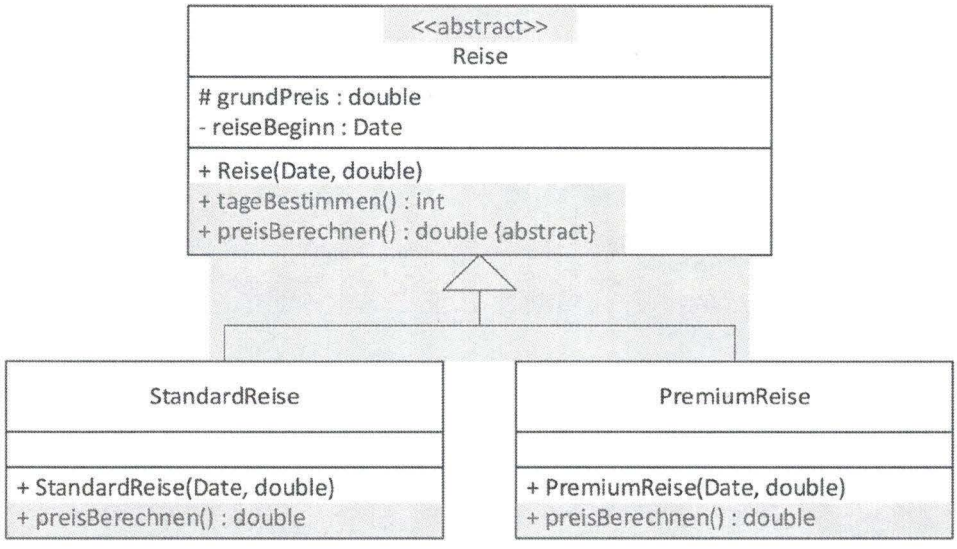
c) 4 Punkte, 2 x 2 Punkte
Der Vorgang D kann sich …
um vier Tage (Freier Puffer) verzögern, ohne dass der Beginn (FAZ) des folgenden Vorgangs E verschoben werden muss.
um zwölf Tage (Gesamtpuffer) verzögern, ohne dass das Projektende verschoben werden muss.

Hinweis für Prüfer:
Falls kein Bezug zum Vorgang D genommen wurde und eine allgemeine Beschreibung von FP und GP erfolgte, dann nur 2 Punkte (2 x 1 Punkt)

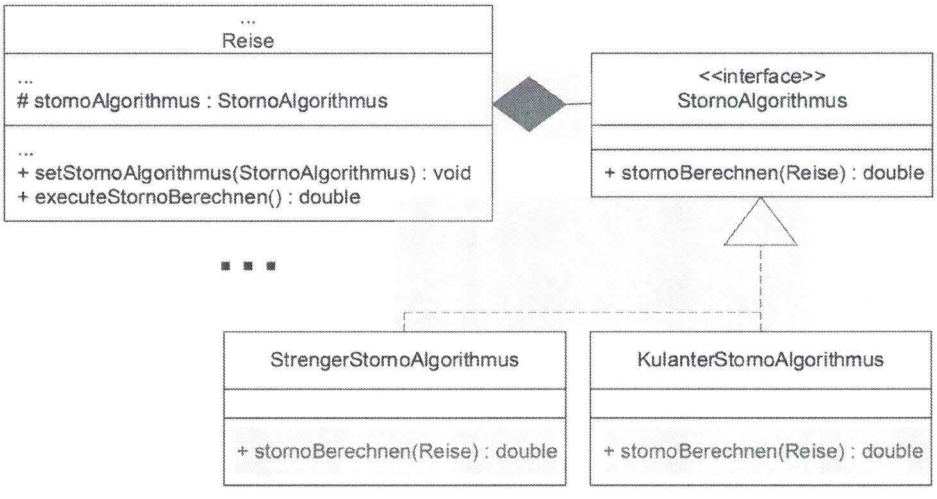
2. Handlungsschritt (25 Punkte)

aa) 3 Punkte
2 Punkte, 2 x 1 Punkt je Vererbungsbeziehung zwischen
– *Reise und StandardReise*
– *Reise und PremiumReise*
1 Punkt für Klasseneigenschaft *<<abstract>>*

ab) 4 Punkte, 4 x 1 Punkt für jede richtig eingetragene Methode



b) 6 Punkte
2 Punkte für Komposition zwischen Reise und *StornoAlgorithmus*
2 Punkte, 2 x 1 Punkt je Implementierung zwischen
– *StornoAlgorithmus* und *StrengerStornoAlgorithmus*
– *StornoAlgorithmus* und *KulanterStornoAlgorithmus*
2 Punkte, 2 x 1 Punkt je Methode



ca) 4 Punkte

```
executeStornoBerechnen(): double
  Rückgabe stornoAlgorithmus.stornoBerechnen(this)
```

cb) 5 Punkte

```
StandardReise(Date rD, double gP)
  Aufruf des Basisklassenkonstruktors Reise(rD, gP)
  stornoAlgorithmus := new KulanterStornoAlgorithmus()
```

cc) 3 Punkte

```
hurtigSR.setStornoAlgorithmus(new KulanterStornoAlgorithmus())
```

Hinweis für Prüfer:
Am Pseudocode sollte der Umgang mit dem Aufruf von objektorientierten Methoden erkennbar sein.

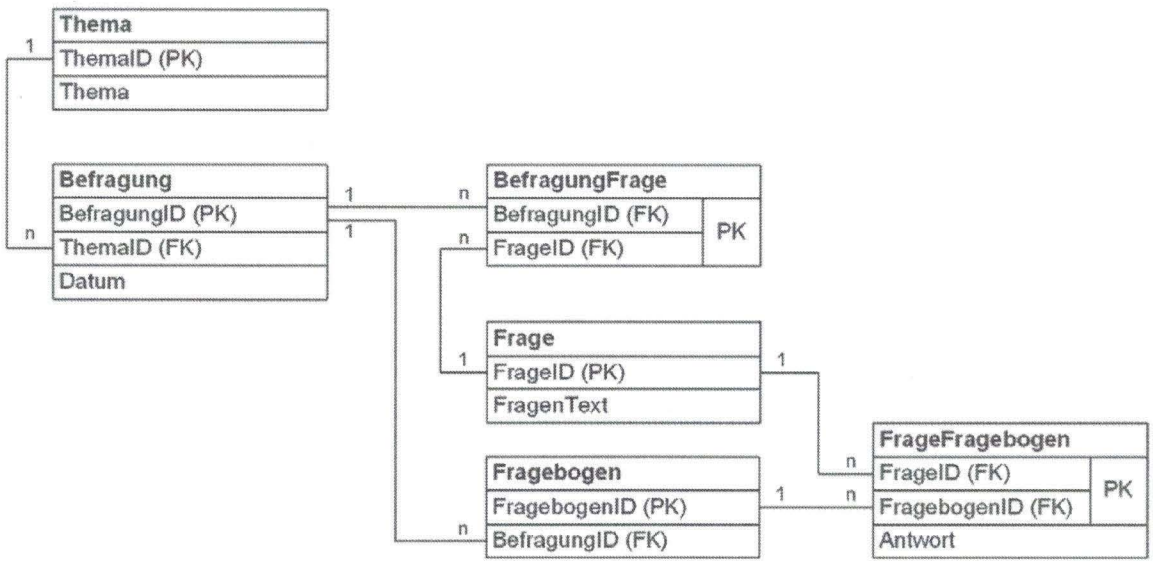
3. Handlungsschritt (25 Punkte)

```
Funktion freieSitze(anzahlSitze: int): int

int sitzNr := 0

// geht Reihe für Reihe durch
für reihe := 0 bis 6

  // geht Reihe Sitz für Sitz durch
  sitz := 0
  solange sitz < 30 - anzahlSitze + 1

    wenn array[reihe][sitz] dann
      freieSitze := 0
      sitzNr := (reihe+1) * 100 + sitz + 1
      // prüft, ob die gewünschte Anzahl
      // nebeneinanderliegender freier Sitze vorhanden ist
      solange sitz < 30 und array[reihe][sitz]
        freieSitze := freieSitze + 1
        sitz := sitz + 1
        wenn freieSitze = anzahlSitze dann
          Rückgabe sitzNr
        Ende wenn
      Ende solange

    sonst
      sitz := sitz + 1
    Ende wenn

  Ende solange sitz

Ende für reihe

Rückgabe 0

Ende Funktion
```

Hinweis für Prüfer:
Die Kommentare sind nicht Teil der erwarteten Lösung.
Andere Lösungen sind möglich.

4. Handlungsschritt (25 Punkte)

6 Punkte, 6 x 1 Punkt je Tabelle
6 Punkte, 6 x 1 Punkt je Primärschlüssel-Attribut
4 Punkte, 4 x 1 Punkt je Fremdschlüssel-Attribut
3 Punkte, 3 x 1 Punkt je einfaches Attribut
6 Punkte, 6 x 1 Punkt je Beziehung mit Kardinalitäten



5. Handlungsschritt (25 Punkte)

a) 5 Punkte

```
UPDATE Fehlzeit
SET Fehlzeit.Bis_Datum = '18.11.2017',
    Fehlzeit.Grund = 'Dienstreise',
    Fehlzeit.Fehltage = 2
WHERE Fehlzeit.FZ_ID = 4;
```

b) 10 Punkte

```
SELECT Mitarbeiter.MA_ID, Mitarbeiter.Nachname,
       Mitarbeiter.Vorname, SUM(Fehlzeit.Fehltage)
FROM Mitarbeiter
LEFT JOIN Fehlzeit ON Mitarbeiter.MA_ID = Fehlzeit.MA_ID
WHERE Fehlzeit.Grund = 'Urlaub'
       AND Fehlzeit.Von_Datum >= '01.01.2017'
       AND Fehlzeit.Bis_Datum <= '31.12.2017'
GROUP BY Mitarbeiter.MA_ID, Mitarbeiter.Nachname, Mitarbeiter.Vorname;
```

ca) 2 Punkte

```
DROP TABLE Fehlzeit;
```

cb) 3 Punkte

```
CREATE TABLE Fehlzeitgrund(
    Grund_ID integer,
    Grund string,
    PRIMARY KEY(Grund_ID)
    );
```

Formulierung mit **CONSTRAINT** auch möglich

cc) 5 Punkte

```
CREATE TABLE Fehlzeit(
    Fehlzeit.MA_ID INTEGER,
    Fehlzeit.Von_Datum DATE,
    Fehlzeit.Bis_Datum DATE,
    Fehlzeit.Grund_ID INTEGER,
    Fehlzeit.Fehltage INTEGER,
    PRIMARY KEY(Fehlzeit.MA_ID),
    FOREIGN KEY(Fehlzeit.Grund_ID) REFERENCES Fehlzeitgrund(Grund_ID)
    );
```

Formulierung mit **CONSTRAINT** ist auch möglich.