Abschlussprüfung Sommer 2024
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.

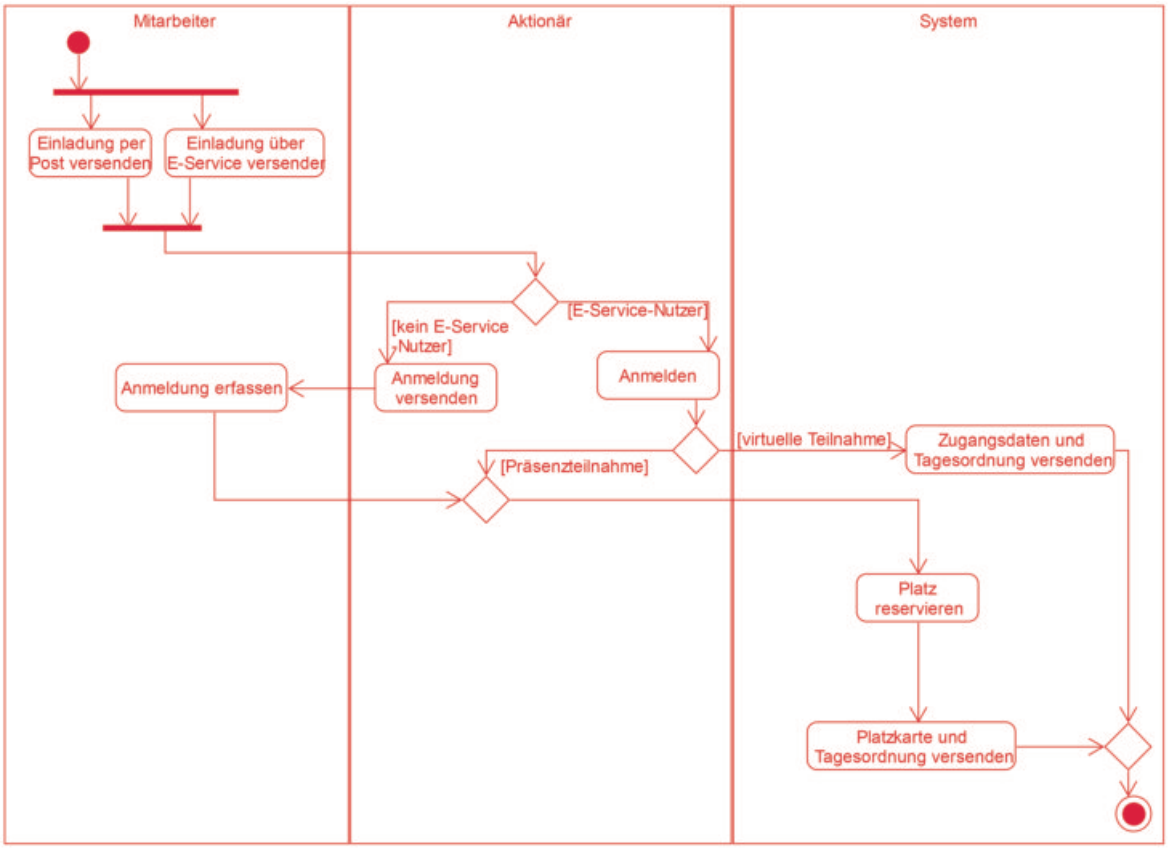
Zu beachten ist die unterschiedliche Dimension der Aufgabenstellung (nennen – erklären – beschreiben – erläutern usw.).

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)

a) 14 Punkte
Je Aktivität an der korrekten Stelle 1 Punkt
Je Verzweigung 2 Punkte
Parallelisierung 2 Punkte



ba) 2 Punkte
Z. B. Word-Dokumente aus externen Quellen zu öffnen, birgt ein hohes Sicherheitsrisiko.

bb) 3 Punkte
Z. B. Einladung direkt im Web-Browser auf der Plattform öffnen, da so keine Manipulation des Dokumentes möglich ist.

ca) 3 Punkte
Erwartet wird eine Erläuterung zu Schwachstellen,
z. B. E-Mail-Link könnte in falsche Hände gelangen.

cb) 3 Punkte
Erwartet wird eine Erläuterung zu einem Vorgehen,
z. B. eine SMS mit einem Transaktionscode parallel zum Rücksetz-Link versenden, sodass ein 2. Weg benutzt wird, um das Rücksetzen abzusichern.

2. Aufgabe (20 Punkte)

a) 12 Punkte
Beispielhaft mit bubbleSort:

```
// BubbleSort
sort (Tageskurs[] kurse, Function vergleiche) : void
   for (i = 0; i < kurse.length - 1; i++)
    for (j = 0; j < kurse.length - 1 - i; j++)
      if (vergleiche(kurse[j], kurse[j+1]) > 0)
        Tageskurs k = kurse[j]
        kurse[j] = kurse[j+1]
        kurse[j+1] = k
      End if
    End for
   End for
End sort
```

b) 8 Punkte

```
notierungPlus(Tageskurs[] kurse): Integer

   count: Integer = 0
   for (i = 0; i < kurse.length; i++)
      if (kurse[i].getProzVAktie() > kurse[i].getProzVDAX() ) count++

   End For

   Return count

End notierungPlus
```

3. Aufgabe (25 Punkte)

aa) 4 Punkte
- Es können Fehler genauer lokalisiert werden.
- Es werden alle Programmteile getestet, auch solche, welche durch die Anforderungsspezifikation nicht erfasst wurden.
- Es wird eine hohe Testabdeckung des Quellcodes erreicht.
- Unterstützt das Optimieren des Quellcodes
- u. a.

ab) 4 Punkte
Regressionstests sind Tests, mit deren Hilfe sichergestellt werden soll, dass das Hinzufügen von neuem Code, Optimierungen innerhalb des Codes oder das Beheben von Fehlern die vorhandene Funktionalität nicht beeinträchtigt und noch den Anforderungen entspricht.

b) 12 Punkte
Anweisungsüberdeckung: (4 Punkte)
Die Anweisungsüberdeckung ist ein Maß für den prozentualen Anteil der Anweisungen in einem Programm, die von einer Reihe von Tests ausgeführt werden. Bei der Anweisungsüberdeckung sollten die Testdaten so gewählt werden, dass alle Anweisungen im Code mindestens einmal ausgeführt werden.

Zweigüberdeckung: (4 Punkte)
Die Zweigüberdeckung ist ein Maß für den prozentualen Anteil der Verzweigungen in einem Programm, die von einer Reihe von Tests ausgeführt werden. Durch die Auswahl der Testdaten soll sichergestellt werden, dass alle Verzweigungen jeder Bedingung mindestens einmal ausgeführt werden. Eine vollständige Zweigüberdeckung beinhaltet automatisch eine vollständige Anweisungsüberdeckung.

Pfadüberdeckung: (4 Punkte)
Die Pfadüberdeckung ist ein Maß für den prozentualen Anteil der Pfade in einem Programm, die von einer Reihe von Tests ausgeführt werden. Ein Pfad ist eine Folge von Anweisungen, die am Einstiegspunkt des Programms beginnt und an einem Ausstiegspunkt endet. Eine vollständige Pfadüberdeckung beinhaltet automatisch eine vollständige Zweigüberdeckung.

c) 5 Punkte

Nummer	Pfad
1	B1, A2
2	B1, A1, B2, A3, B3, A4
3	B1, A1, B2, B3, A4
4	B1, A1, B2, A3, B3
5	B1, A1, B2, B3

Die Reihenfolge der Pfade ist beliebig.

4. Aufgabe (30 Punkte)

a) 2 Punkte

```
DELETE FROM AktienKurs WHERE AK_AktieID = 4
```

b) 8 Punkte

```
SELECT B.B_ID
       ,B.B_BoersenName
       ,MIN(AK.AK_Kurs) AS KursMin
       ,MAX(AK.AK_Kurs) AS KursMax
       ,AVG(AK.AK_Kurs) AS KursDurchschnitt
       ,COUNT(AK.AK_ID) AS AnzahlTransaktionen
  FROM Boerse AS B
    INNER JOIN AktienKurs AS AK ON AK.AK_BoerseID = B.B_ID
  WHERE AK.AK_AktieID = 6
  GROUP BY B.B_ID, B.B_BoersenName;
```

c) 8 Punkte

```
INSERT INTO AktienKursArchiv
     (AK_ID, AK_DatumZeit, AK_Kurs, AK_BoerseID, AK_AktieID)
SELECT
     AK_ID, AK_DatumZeit, AK_Kurs, AK_BoerseID, AK_AktieID
  FROM AktienKurs
  WHERE YEAR(AK_DatumZeit) < YEAR(TODAY);

DELETE FROM AktienKurs WHERE YEAR(AK_DatumZeit) < YEAR(TODAY);
```

Andere korrekte Lösungen sind auch möglich.

d) 12 Punkte

```
SELECT YEAR(AK.AK_DatumZeit) AS Boersenjahr
     ,MIN(AK.AK_Kurs) AS KursMin
     ,MAX(AK.AK_Kurs) AS KursMax
     ,COUNT(AK.AK_ID) AS AnzahlTransaktionen
  FROM AktienKurs AS AK
  WHERE AK.AK_AktieID = 6
   GROUP BY YEAR(AK.AK_DatumZeit)
UNION ALL
SELECT YEAR(AKA.AK_DatumZeit)
     ,MIN(AKA.AK_Kurs)
     ,MAX(AKA.AK_Kurs)
     ,COUNT(AKA.AK_ID)
  FROM AktienKursArchiv AS AKA
  WHERE AKA.AK_AktieID = 6
   GROUP BY YEAR(AKA.AK_DatumZeit)
ORDER BY Boersenjahr DESC;
```