Klausur Softwaretechnik

16. März 2004

Musterlösung

Aufg.1	Aufg.2	Aufg.3	Aufg.4	Aufg.5	Aufg.6	Σ	Note
10	6	6	10	15	13	60	1.0

Die Klausur besteht aus 20 Seiten und ist geheftet abzugeben. Für die Vollständigkeit nicht gehefteter Klausuren wird keine Verantwortung übernommen.

Punktegrenzen	Note
0.0	5.0
18.0	4.0
20.0	3.7
23.0	3.3
26.0	3.0
29.0	2.7
32.0	2.3
35.0	2.0
38.0	1.7
41.0	1.3
44.0	1.0

Aufgabe 1 (Wissensfragen) – Für alle (10 Punkte)

Hinweis: Falsche Kreuze geben negative Punkte, fehlende Kreuze, sowie fehlende oder falsche Freitext-Antworten bewirken nichts. Weniger als 0 Punkte können Sie mit dieser Aufgabe insgesamt nicht erreichen.

a) Nennen Sie 3 Grundprinzipien zur Beherrschung der Komplexität bei der Entwicklung von Software.

Abstraktion — 0,5

Modularisierung — 0,5

Hierarchisierung oder Verfeinerung — 0,5

Strukturierung — 0,5

1,5

b) Kreuzen Sie nachfolgend alle Tätigkeiten an, die Sie während der Definitionsphase durchführen.

☐ Wirtschaftlichkeitsrechnung — - 0,5

X Pflichtenheft erstellen — 0,5

☐ Durchführbarkeitsuntersuchung — - 0,5

X Anforderungen verabschieden — 0,5

1,0

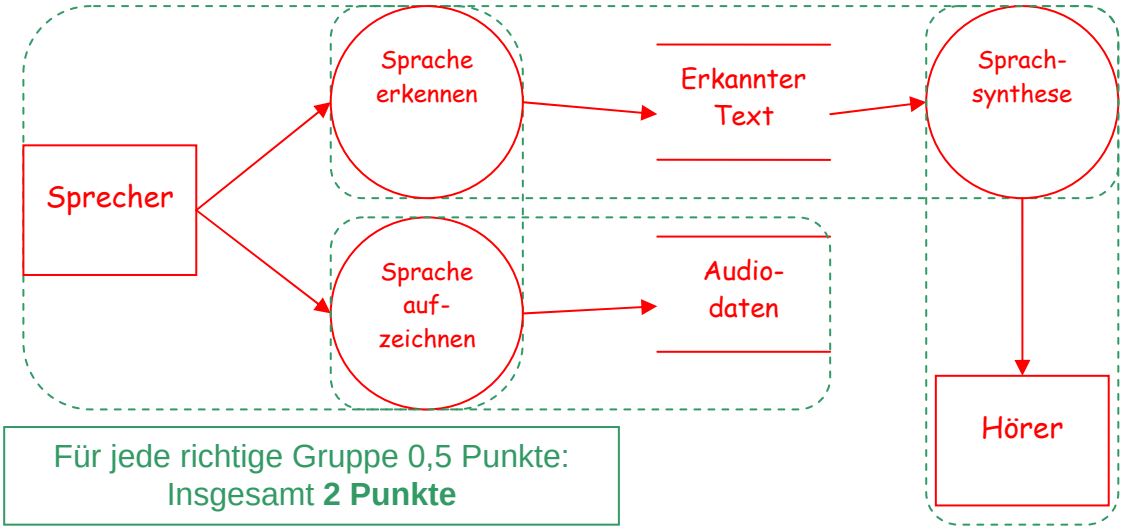
c) Nachfolgend sind einige Basiskonzepte der Software-Entwicklung und verschiedene Sichten auf zu entwickelnde Software aufgeführt. Ordnen Sie Konzepte und Sichten einander korrekt zu.

1. Funktionale Sicht	___ D ___	0,5	A. Klassendiagramm
2. Datenorientierte Sicht	__ F, D _	0,5	B. Entscheidungstabelle
3. Objektorientierte Sicht	___ A ___	0,5	C. Pseudocode
4. Algorithmische Sicht	___ C ___	0,5	D. Datenflußdiagramm
5. Regelbasierte Sicht	___ B ___	0,5	E. Petrinetze
6. Zustandsorientierte Sicht	___ E ___	0,5	F. Syntaxdiagramm

Insgesamt: 3

d) Ihre Aufgabe ist es, ein Sprachverarbeitungssystem mittels eines Datenflußdiagramms zu modellieren. Die erste Funktion ist das *Erkennen von Sprache*: *Erkannter Text* eines *Sprechers* wird in einer XML-Datenbank abgespeichert. Gleichzeitig wird die *Sprache aufgezeichnet* und in einer zweiten Datenbank für binäre *Audiodaten* abgespeichert. Neben dem Erkennen von Sprache bietet das System noch eine zweite Funktion, die *Sprachsynthese*: In der XML-Datenbank abgespeicherter Text kann synthetisiert und als Sprache einem *Hörer* ausgegeben werden.

Hinweis: Verwenden Sie für die einzelnen Teile des Datenflußdiagramms ähnliche Bezeichner wie im obigen Text kursiv hervorgehoben.



Für jede richtige Gruppe 0,5 Punkte:
Insgesamt **2 Punkte**

e) Beantworten Sie die folgenden Fragen

Für jede richtige Antwort +0,5.
Für jede falsche Antwort -0,5.
Insgesamt **2,5 Punkte**

Richtig/Falsch

X / ☐ Eine Transition in einem Petrinetz ist L3-lebendig für eine Markierung M wenn Sie für jede erreichbare Markierung L1-lebendig ist.

☐ / X Eine Unterklasse muss alle abstrakten Methoden ihrer Oberklasse implementieren.

X / ☐ Als Regressionstest bezeichnet man das wiederholte Ausführen einer Testsuite nach Programmänderungen.

☐ / X Eine inhaltlich vollständige Entscheidungstabelle ist immer auch formal vollständig.

☐ / X Ein Cache heißt direkt-abgebildet, wenn er nur aus einem einzigen Satz besteht.

Aufgabe 2 (XML) – Für Informatiker (6 Punkte)

Betrachten Sie das folgende XML-Instanzdokument. Vervollständigen Sie das gegebene XSD-Schema so, daß das XML-Instanzdokument für dieses Schema gültig ist.

```
<?xml version="1.0" encoding="utf-8" ?>
<swt:Brief xmlns:swt="http://www.ipd.uka.de/Tichy/SWT/"
           Absender="Alice" Betreff="Projektbesprechung">
  <swt:Absatz>Bob, wir wollen uns wegen des Projektes treffen.</swt:Absatz>
  <swt:Absatz>Kannst Du morgen mal vorbei kommen?</swt:Absatz>
  <swt:Absatz>Bis dann, Alice.</swt:Absatz>
</swt:Brief>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema targetNamespace="http://www.ipd.uka.de/Tichy/SWT/"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

  <xs:element name="Brief">

    <xs:complexType>
      <xs:sequence>
        <xs:element name="Absatz" type="xs:string"
                                  maxOccurs="unbounded"></xs:element>
      </xs:sequence>
        <xs:attribute name="Absender" type="xs:string"
                                   use="required"></xs:attribute>
        <xs:attribute name="Betreff" type="xs:string"
                                  use="required"></xs:attribute>
    </xs:complexType>

  </xs:element>

</xs:schema>
```

Insgesamt: 4

complexType + sequence: 1

element: 1

element: 1

element: 1

Insgesamt: 2

Betrachten Sie das obige XML-Instanzdokument. Welche Werte haben die folgenden Informationseinheiten des XML Information Set?

a) Document_Information_Item::document_encoding_scheme — utf-8 — 0,5

b) Document_Information_Item::standalone — yes — 0,5

c) Element_Information_Item::namespace_Name — http://www.ipd.uka.de/Tichy/SWT/ — 0,5

d) Attribute_Information_Item::prefix — swt — 0,5

Aufgabe 3 (CVS, VSS) – Für Informatiker (6 Punkte)

In dieser Aufgabe geht es um die gleichzeitige Verwendung des *Concurrent Versions Systems* durch mehrere Benutzer.

Ranjid ist Programmierer bei einer kleinen Outsourcing-Softwarefirma in Bangalore. Die folgende Funktion zur Fehlerbehandlung ist ein Ausschnitt aus einer größeren Quelltextdatei.

```
protected void Fehlerbehandlung(string strFehler)
{
}
```

In Ranjids Firma ist der CVS-Server `cvs.mydoomsoft.in` verfügbar. Die Dateien des aktuellen Projekts liegen in dem Repository `/projects/TBQ`. Ranjid verwendet das `pserver`-Protkoll und den Benutzernamen `ranjid`.

Wie muss Ranjid die Umgebungsvariable `cvsroot` auf seinem Rechner setzen, um sich auf dem CVS-Server einloggen zu können?

1
fehlender Teil -0,5

```
d:\Ranjid\Prj>set cvsroot= :pserver:ranjid@cvs.mydoomsoft.in:
                           /projects/TBQ
```

Ranjid erstellt nun wie folgt für die obige Quelltextdatei ein neues Modul.

```
d:\Ranjid\Prj>cvs import -m "Die 1. Version" Modul vtag rtag
```

Am nächsten Tag möchte Ranjid seine Arbeit fortsetzen. Geben Sie dazu das CVS-Kommando an, mit dem er seine Quelltextdatei auschecken kann.

```
d:\Ranjid\Prj> cvs checkout Modul  oder  cvs co Modul
```

0,5

In seinem lokalen Arbeitsbereich nimmt Ranjid nun die folgenden Ergänzungen an seinem Quelltext vor.

```
protected void Fehlerbehandlung(string strFehler)
{
   MessageBox.Show("Fehler: " + strFehler);
}
```

Kurz nach Ranjid hat auch sein Kollege **Yun** dieselbe Datei ausgecheckt und in seinem **lokalen Arbeitsbereich** die folgenden Änderungen vorgenommen:

```
protected void Fehlerbehandlung(string strFehler)
{
   Debug.WriteLine("Fehler: " + strFehler);
}
```

Mit welchem CVS-Kommando kann Ranjid Änderungen am Repository in seinem Arbeitsbereich sichtbar werden lassen?

`d:\Ranjid\Prj>` cvs update

0,5

Wie sieht die Quelltextdatei in Ranjids lokalem Arbeitsbereich nach dem Abgleich mit dem Repository aus?

```
protected void Fehlerbehandlung(string strFehler)
{
   MessageBox.Show("Fehler: " + strFehler);
}
```

1

Im Laufe der Entwicklung hat Ranjids und Yuns Quelltextdatei die Version 4.2.1.3. Was sind dabei die Werte für...

... das Variantenlevel? ___3___

... das Releaselevel? ___2___

... die Variantennummer? ___1___

... die Releasenummer? ___4___

Variantennummer/level: 0,5

Releasenummer/level: 0,5

In Zukunft möchte Ranjid *Microsoft Visual Source Safe* verwenden. Er hat dazu aber noch zwei Fragen:

1. Warum kann man bei *Visual Source Safe* binäre Dateien nicht mehrfach auschecken bzw. warum würde das auch keinen Sinn machen?

   Kein binary/semantic merge definiert.

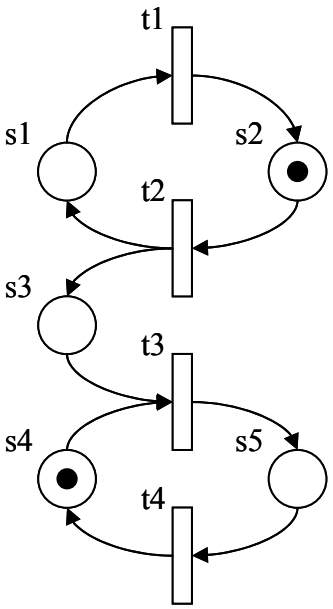
   1

2. Bei dem von *VSS* verwendeten Rückwärts-Delta Mechanismus zum Speichern von Änderungen an Dateien wird die aktuelle Version gespeichert und die Änderungen zu früheren Versionen gespeichert.

   0,5 + 0,5

Aufgabe 2 (Petrinetze) – Für Informationswirte (6 Punkte)

Betrachten sie die folgenden Petrinetze und beantworten Sie die zugehörigen Fragen.

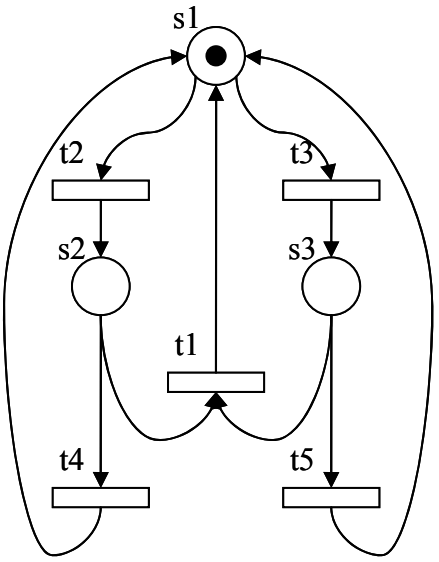


0,5

Ist das Petrinetz beschränkt (Ja/Nein)? nein

Geben Sie die Lebendigkeit (L0 – L4) des Petrinetzes an:

Das Netz ist L4-lebendig. 1

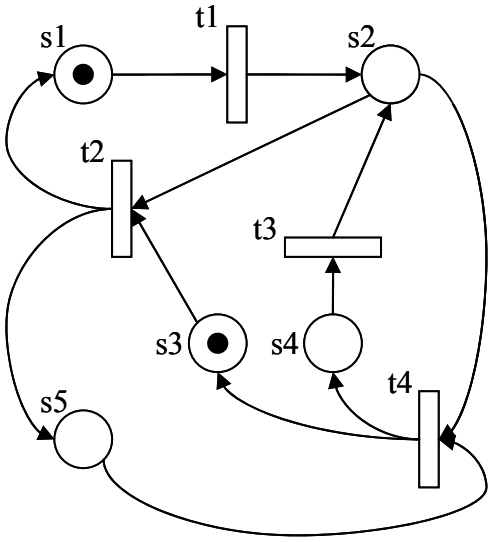


Geben Sie die Lebendigkeit (L0 – L4) der Transitionen t1, t2 und t4 an:

Die Transition t1 ist L0. 0,5

Die Transition t2 ist L4. 0,5

Die Transition t4 ist L4. 0,5



Geben Sie die Lebendigkeit (L0 – L4) der Transitionen t3 und t4 an:

Die Transition t3 ist L4. 1

Die Transition t4 ist L4. 1

Geben Sie die Lebendigkeit (L0 – L4) des Petrinetzes an:

Das Netz ist L4. 1

Aufgabe 3 (Optimierung) – Für Informationswirte (6 Punkte)

Die folgende Java-Klasse repräsentiere einen Ordner in einem Dateisystem.

```
public class FileFolder
  {
     // Name des Ordners
     private String strFolderName;

     // Wenn dieser Ordner keinen Elternordner hat,
     // ist fileFolderParent gleich null
     private FileFolder fileFolderParent;

     // Für einen Ordner "/Verwaltung/Buchhaltung/"
     // gäbe diese Methode den lokalen Ordnernamen
     // "Buchhaltung" zurück
     public String getFolderName()
     {
        return strFolderName;
     }

     // Gibt eine Referenz auf den Eltern-Ordner zurück
     public FileFolder getParentFolder()
     {
        return fileFolderParent;
     }
  }
```

Vervollständigen Sie die folgende **rechtsrekursive** Funktion `buildFilePathHelper` so, dass nach dem Ende aller rekursiven Aufrufe der komplette Pfad eines Ordners zurückgegeben wird. Für einen Ordner „Einkauf“ würde z.B. der Pfad „/Verwaltung/Buchhaltung/Einkauf/“ zurückgegeben werden, wenn die Elternordner Buchhaltung und Verwaltung sind.

```
String buildFilePath(FileFolder f)
{
    String strPath = "";

    return buildFilePathHelper(f, strPath);
}

String buildFilePathHelper(FileFolder f, String strPath)
{

    if (f.getParentFolder() != null)
    {

        strPath = f.getFolderName() + "/" + strPath;


        return buildFilePathHelper(f.getParentFolder(), strPath);
    }
    else
    {
        return "/" + strPath;
    }
}
```

Insgesamt: 3,5

1

1

1

0,5 (nur richtig mit “/”)

Wandeln Sie nun die obige rechtsrekursive Funktion `buildFilePathHelper` nach dem aus der Vorlesung bekanntenVerfahren in eine iterative Form `buildFilePathHelper2` um.

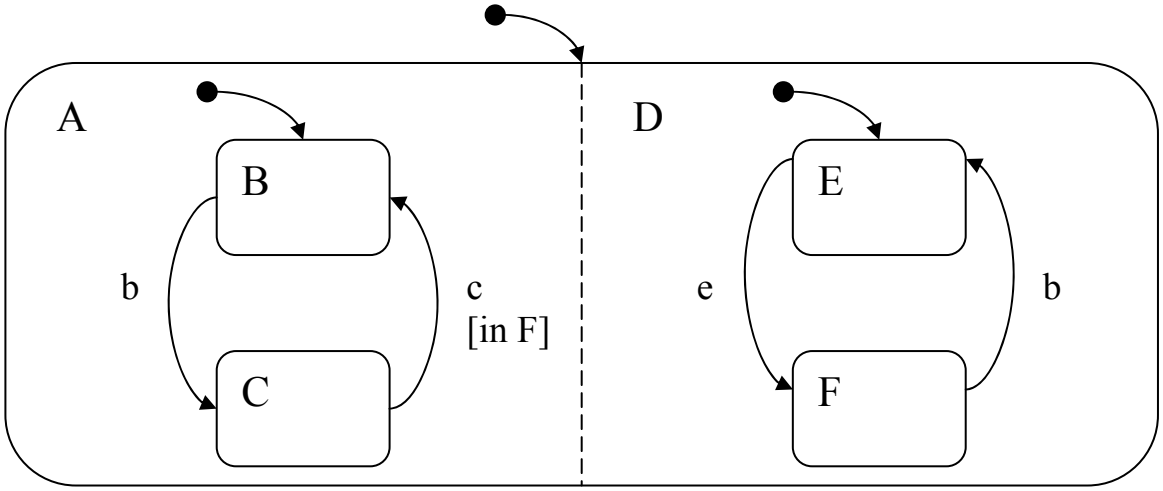
```
String buildFilePathHelper2(FileFolder f, String strPath)
{
    while(f != null)
    {
        strPath = f.getFolderName() + "/" + strPath;

        f = f.getParentFolder();
    }
    return "/" + strPath;
}
```

`while(f != null)`	0,5
`strPath = f.getFolderName() + "/" + strPath;`	1
`f = f.getParentFolder();`	0,5
`return "/" + strPath;`	0,5 (nur richtig mit "/")
	Insgesamt: 2,5

Aufgabe 4 (Petrinetze und Automaten) – Für alle (10 Punkte)

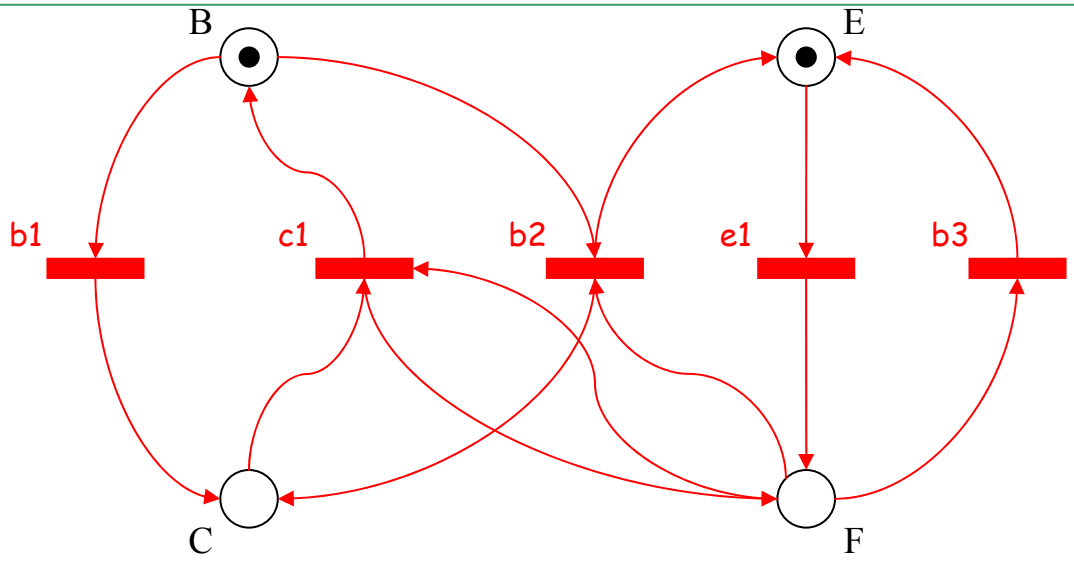
Betrachten Sie den folgenden Harelautomaten.



a) Wandeln Sie den obigen Harelautomaten in ein Petrinetz um. Modellieren Sie das Petrinetz so, daß wenn der Harelautomat sich in den Zuständen B, C, E und F befindet, sich jeweils genau eine Marke in den Stellen B, C, E und F des Petrinetzes befindet (z.B. entspräche der Zustand BE des Harelautomaten der Markierung (1, 0, 1, 0) im Petrinetz).

Hinweis: Die Eingaben b, c und e sollen durch das Schalten von entsprechenden Transitionen im Petrinetz modelliert werden. Für eine Kante k in dem Harelautomaten können Sie wenn nötig mehrere Transitionen k_1, k_2, usw. in dem Petrinetz einführen. Beachten Sie, daß in dem gegebenen Harelautomaten der Zustand *BE* nur einmal angenommen werden kann.

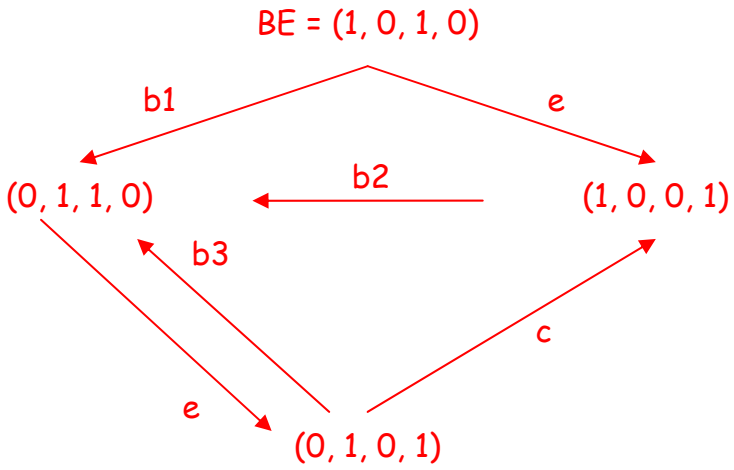
1 Punkt für jede richtige Transition mit allen Ein-/Ausgangskanten
fehlender Pfeil -0.5, Anfangsbelegung fehlt oder falsch -0.5
Benamung fehlt oder falsch -0.5



Insgesamt: 5

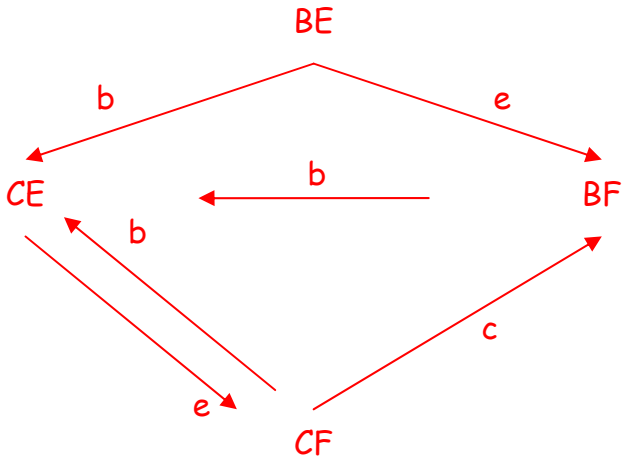
b) Zeigen Sie mittels eines Erreichbarkeitsgraphen, daß die für den Harelautomaten genannte Eigenschaft, daß der Zustand *BE* nur einmal angenommen werden kann, auch für Ihr Petrinetz gilt.

0,5 Punkte für jeden richtigen Übergang (2 Zustände und Transition) **Insgesamt: 3 Punkte**



c) Wandeln Sie den Harelautomaten in einen äquivalenten Mealy-Automaten um.

0,5 Punkte für jeden richtigen Zustand (inkl. aller Übergänge). **Insgesamt: 2 Punkte**



Aufgabe 5 (Kontrollflußorientierter Strukturtest) – Für alle (15 Punkte)

Die Java Methode `sort` sortiert ein Feld von Ganzzahlen mit Hilfe des Bubblesort Algorithmus:

```
public int[] sort(int[] array) {
  int i, j, temp;

  i = array.length - 1;
  while (i > 0) {
    j = 1;
    while (j <= i) {
      if (array[j-1] > array[j]) {
        temp = array[j-1];
        array[j-1] = array[j];
        array[j] = temp;
      }
      j++;
    }
    i--;
  }
  return array;
}
```

a) Vervollständigen Sie die Schablone des Kontrollflussgraphen der Funktion `sort` auf der folgenden Seite. Verwenden Sie dabei für die `while`- und `if`-Anweisungen leere Knoten und numerieren Sie alle Knoten in der Reihenfolge, in der sie im Quelltext auftreten.

b) Formulieren Sie für die gegebenen Testkriterien jeweils eine minimale Menge von Testdaten. Für den Boundary-Interior Test müssen Sie nur Testfälle für das Testen der inneren `while`-Schleife angeben. Geben Sie für alle Testfälle den im Kontrollflussgraphen durchlaufenen Pfad an.

Anweisungsüberdeckungstest: 1 für Testfall, 1 für Pfad. **Insgesamt 2**

[2, 1] → 1, 2, 3, 4, 5, 6, 7, 8, 5, 9, 3, 10

Zweigüberdeckungstest: 1 für Testfall, 1 für Pfad. **Insgesamt 2**

[1, 3, 2] → 1, 2, 3, 4, 5, 6, 8, 5, 6, 7, 8, 5, 9, 3, 4, 5, 6, 8, 5, 9, 3, 10

Boundary-Interior Test (nur für die innere while-Schleife!):

Insgesamt 3

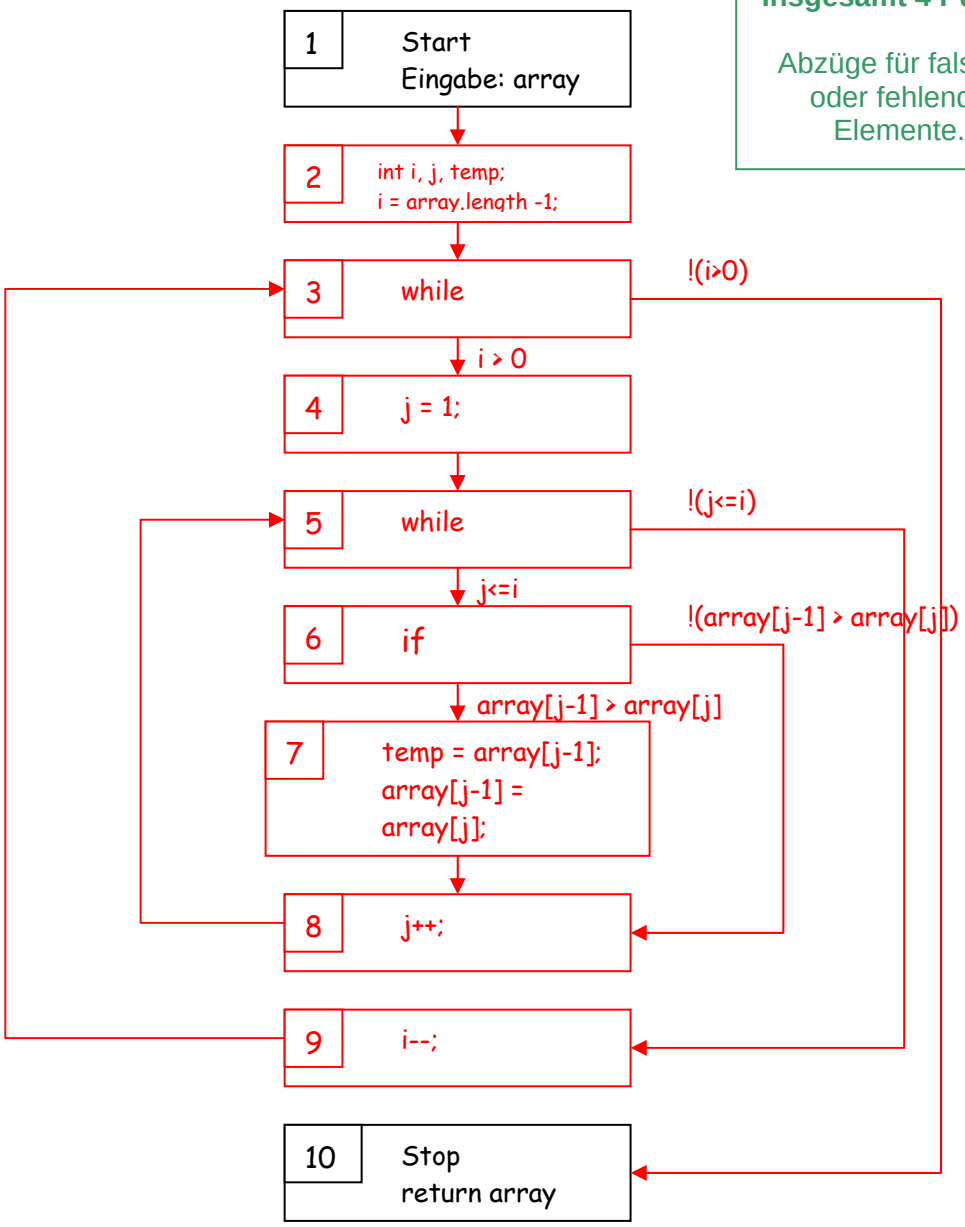
nicht betreten: nicht möglich! — 0,5

Einmal betreten (boundary): [1, 2], [2, 1] — 1

Zwei Iterationen (interior): [1, 2, 3], [2, 1, 3], [1, 3, 2], [3, 2, 1] — 1,5

Insgesamt 4 Punkte

Abzüge für falsche oder fehlende Elemente.



c) Schreiben Sie eine Methode in Java, bei der für den minimalen-mehrfachen Bedingungsüberdeckungstest mehr Testfälle erforderlich sind, als für den Zweigüberdeckungstest.

2

```
public void beispielC(boolean a, boolean b) {

        if (a ^ b)
            System.out.println("if-Teil");
        else
            System.out.println("else-Teil");
}
```

oder

```
public void beispielC(int i) {

        if ((i < 0) || (i > 10))
            System.out.println("if-Teil");
        else
            System.out.println("else-Teil");
}
```

1

Testfälle für minimalen-mehrfachen Bedingungsüberdeckungstest:

a = false, b = false
a = true, b = false
a = false, b = true

oder i = -1, i = 5, i = 15

1

Testfälle für minimalen Zweigüberdeckungstest:

a = false, b = false und a = true, b = false

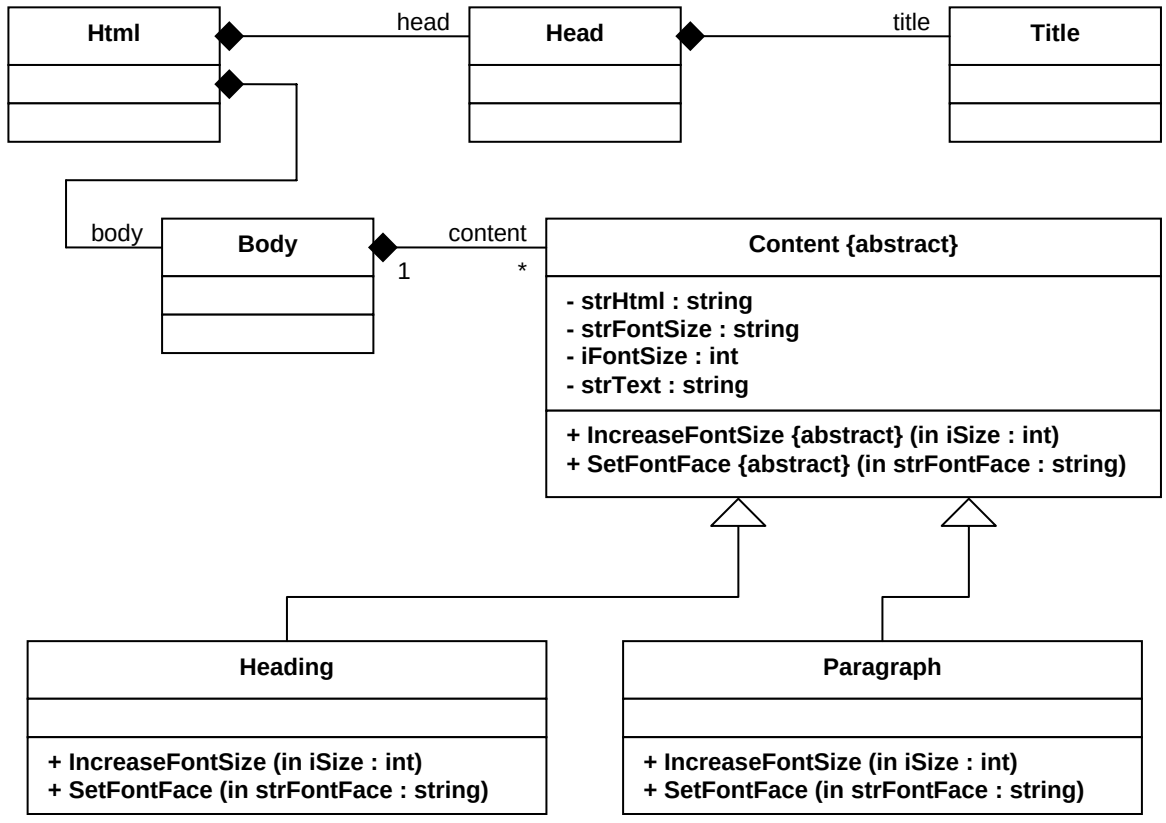
oder i = -1, i = 5

Aufgabe 6 (Entwurfsmuster) – Für alle (13 Punkte)

Das *W3C Document Object Model (DOM)* ist ein sprachunabhängiges Datenmodell und eine API für den programmatischen Zugriff auf z.B. (X)HTML. Solche Dokumente können mit dem DOM komplett als Baum von Objekten im Speicher dargestellt werden. Bestimmte im Dokument vorkommende Elemente werden dabei von einer Instanz der diesem Element entsprechenden Klasse repräsentiert.

Betrachten Sie das folgende HTML-Dokument. Für die Elemente `<html>`, `<head>`, `<title>`, `<body>`, `<p>`, `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` und `<h6>` gebe es in dem DOM jeweils die entsprechenden Klassen `Html`, `Head`, `Title`, `Body`, und `Paragraph`. Die Elemente `<h1>` bis `<h6>` seien alle durch die Klasse `Heading` repräsentiert.

```
<html>
   <head>
      <title>Softwaretechnik</title>
   </head>
   <body>
      <h1>
         <font face="Arial">Willkommen auf der SWT Seite</font>
      </h1>
      <p>
         <font face="Arial" size="4">Die Klausur ist am 16. März</font>
      </p>
   </body>
</html>
```

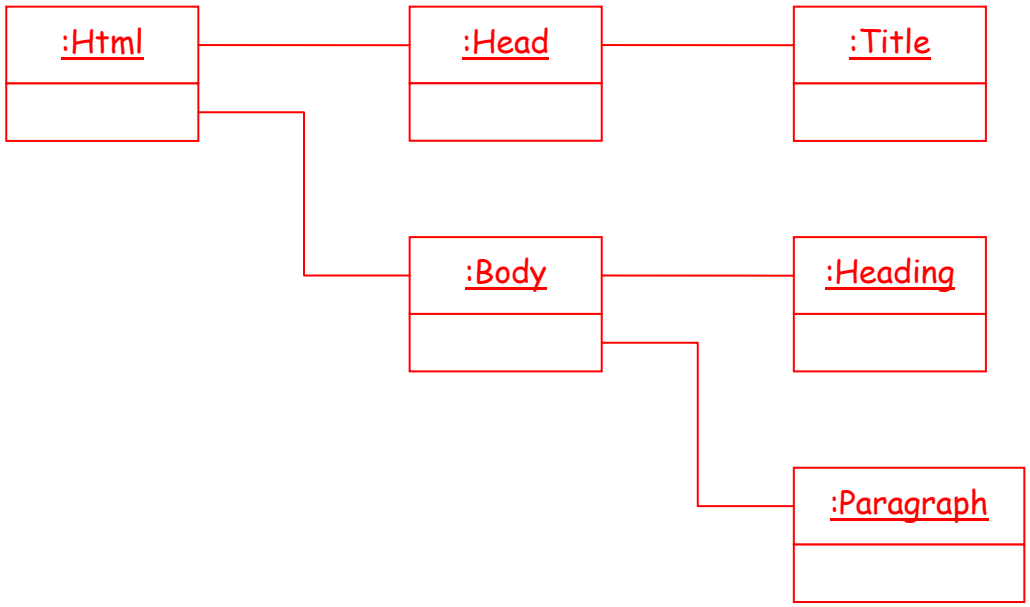


a) Erstellen Sie ein UML-Objektdiagramm, das das DOM für die oben gegebene HTML-Datei repräsentiert. Verwenden Sie dazu die in dem oben gegebenen UML-Klassendiagramm definierten Klassen.

Hinweise:

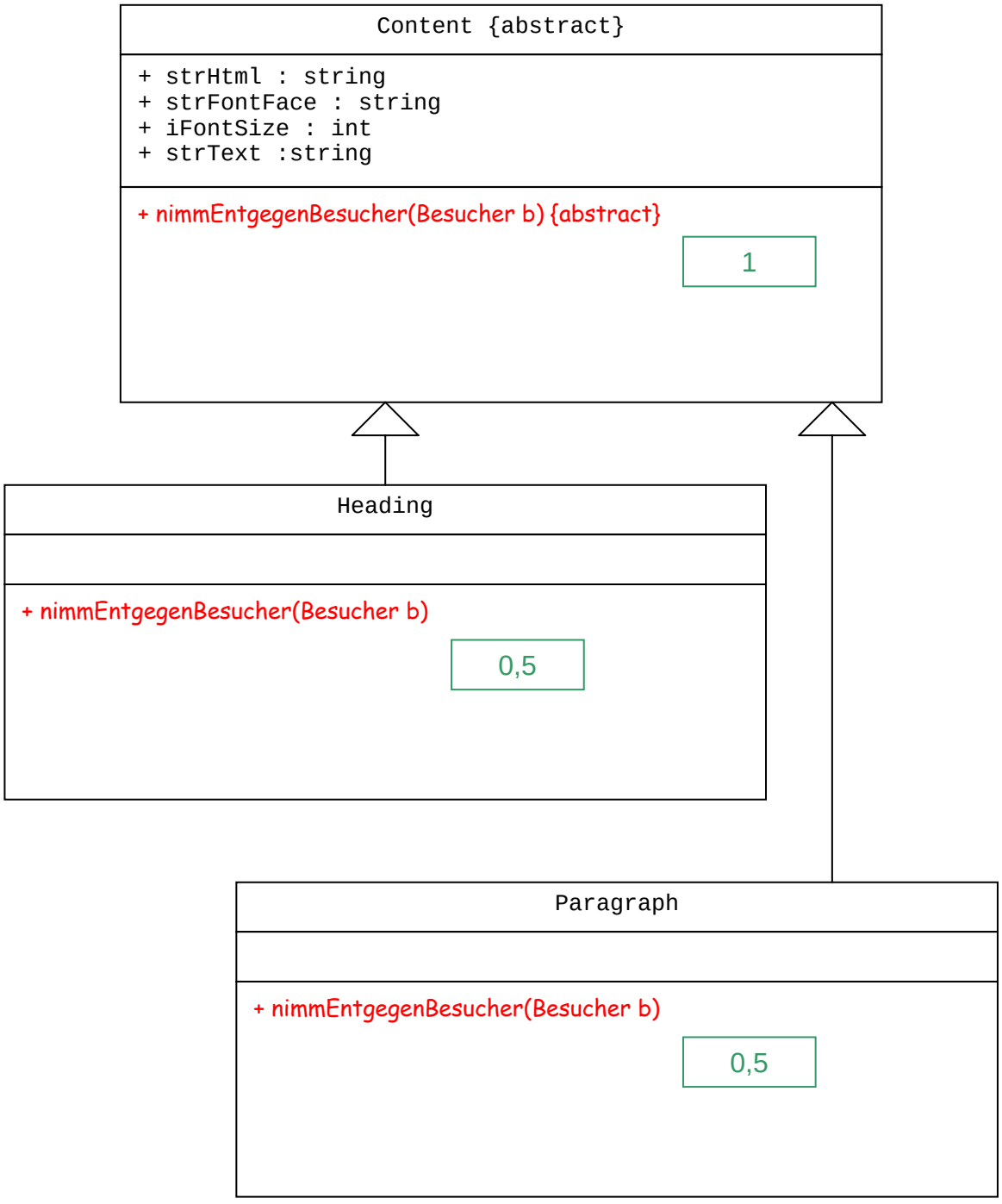
- Sie brauchen die Objekte nicht zu benennen sondern können einfach nur den entsprechenden Klassennamen als Bezeichner verwenden.
- Sie brauchen die Belegungen für die Instanzvariablen nicht anzugeben.

0,5 Punkte pro richtigem Objekt (inkl. Kanten). **Insgesamt 3 Punkte**

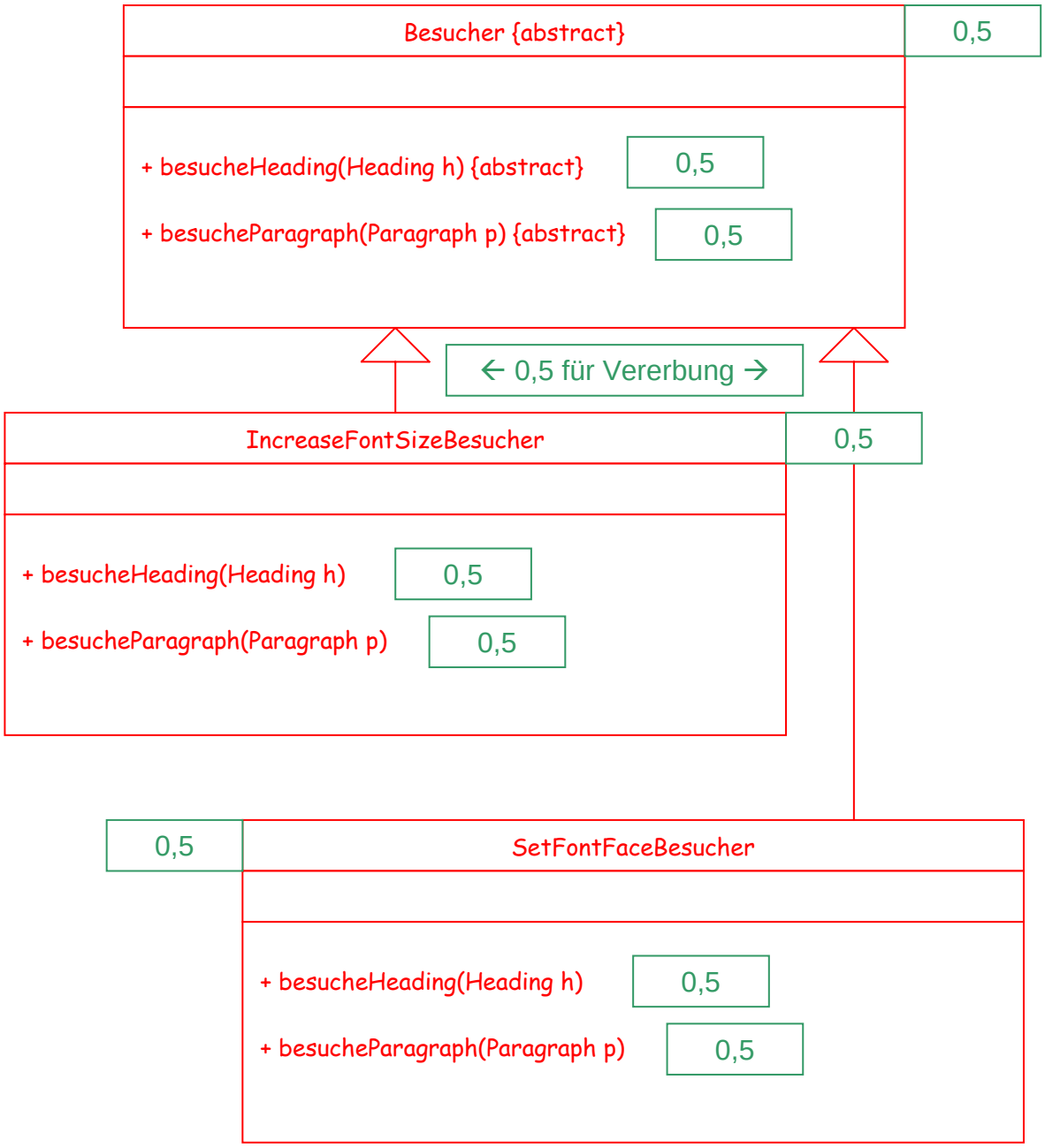


Die von der abstrakten Oberklasse `Content` abgeleiteten Klassen `Heading` und `Paragraph` weisen beide die Funktionen `IncreaseFontSize` und `SetFontFace` auf. In Zunkunft werden jedoch noch weitere `Content`-Unterklassen hinzugefügt. Daher sollen gleiche Operationen in einer gemeinsamen Klasse gekapselt werden.

b) Verwenden Sie das Entwurfsmuster *Besucher*. Kapseln Sie die Operationen `IncreaseFontSize` und `SetFontFace` in zwei Besucherklassen. Vervollständigen Sie dazu auch das folgende UML-Klassendiagramm. Hinweis: Sie brauchen in die Klassen `Content`, `Heading` und `Paragraph` nur die neuen Funktionen einzutragen, die für das Besuchermuster relevant sind. Die Klassen `Html`, `Head`, `Title` und `Body` brauchen Sie nicht anzugeben. Zeichnen Sie das Klassendiagramm für die Besucherklassen auf die nächste Seite.



Insgesamt für dieses Klassendiagramm: 2 Punkte

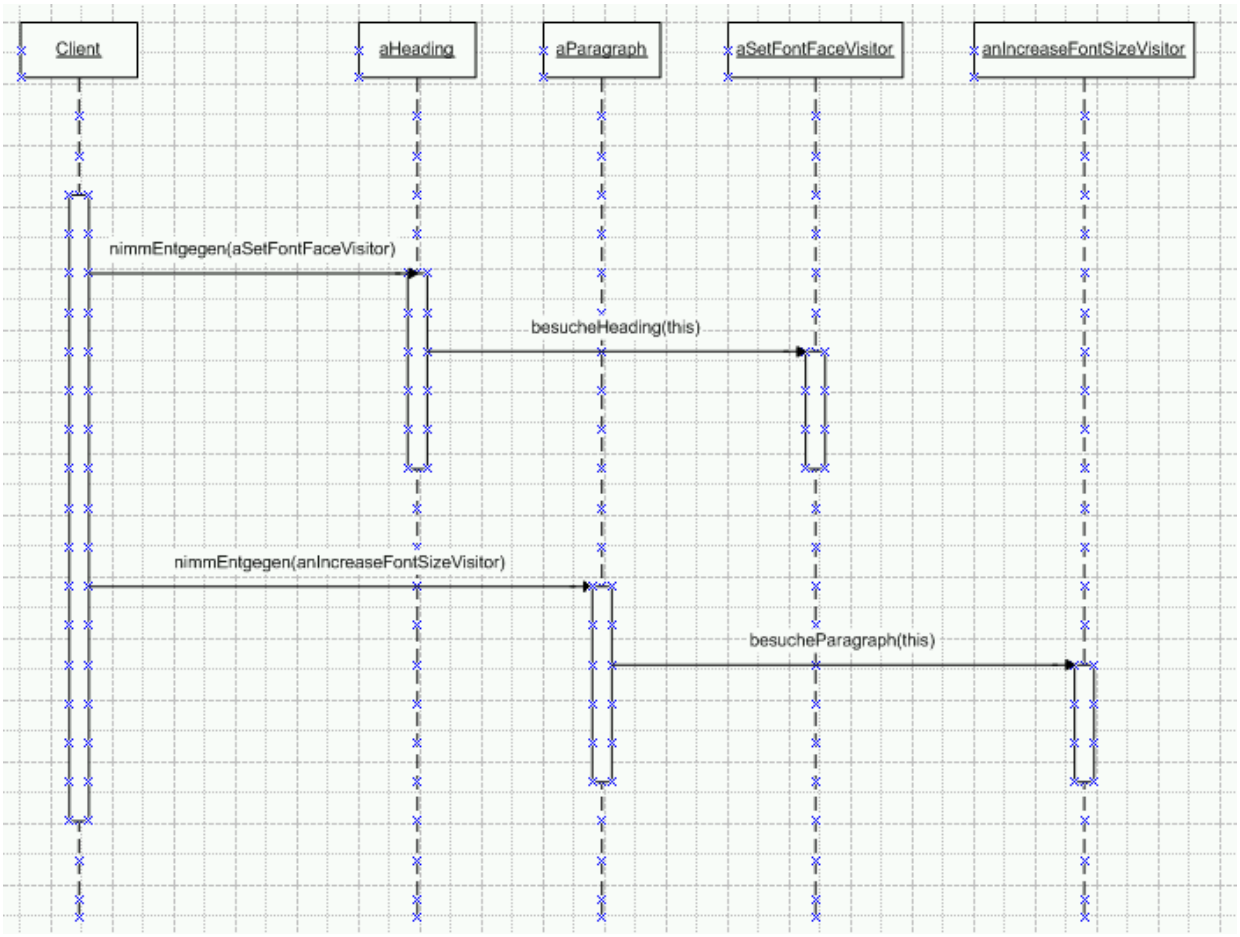


Klassen und Methoden müssen entsprechend als abstrakt gekennzeichnet sein.

Insgesamt für dieses Klassendiagramm: 5 Punkte

c) Vervollständigen Sie das UML-Sequenzdiagramm für folgende Interaktionen:

- Ein Klient `aClient` ändert die Schriftart einer Überschrift, indem er einen entsprechenden Besucher bei einer Instanz der Klasse `Head` vorbeischickt.
- Ein Klient `aClient` ändert die Schriftgröße eines Paragraphen, indem er einen entsprechenden Besucher bei einer Instanz der Klasse `Paragraph` vorbeischickt.



1,5 Punkte für obere und untere Hälfte inkl. korrektem Aktivitätsbalken.

Insgesamt 3 Punkte.