# Musterlösung
# Klausur Softwaretechnik

**08.09.2006**

Prof. Dr. Walter F. Tichy
M. Sc. A. Jannesari
Dipl.-Inform. G. Malpohl

| Note | Punkte | |
|---|---|---|
| 5,0 | 0,0 | 18,5 |
| 4,0 | 19 | 21,5 |
| 3,7 | 22 | 24,5 |
| 3,3 | 25 | 27,5 |
| 3,0 | 28 | 30,5 |
| 2,7 | 31 | 33,5 |
| 2,3 | 34 | 37,5 |
| 2,0 | 38 | 41,5 |
| 1,7 | 42 | 45,5 |
| 1,3 | 46 | 49,5 |
| 1,0 | 50 | 60 |

**Aufgabe 1:** Aufwärmen (4+2+2,5+2+3+1,5= 15P)

a.) Kreuzen Sie an, ob die Aussage wahr oder falsch ist. (4 P)
*Hinweis:* Jedes korrekte Kreuz zählt 0,5 Punkte, jedes falsche Kreuz bewirkt 0,5 Punkte Abzug! Die Teilaufgabe wird mindestens mit 0 Punkten bewertet.

| wahr | falsch | Aussage |
|---|---|---|
| | X | Die Durchführbarkeitsuntersuchung und Erzeugung des Pflichtenhefts werden beide in der Definitionsphase durchgeführt. |
| X | | Wenn eine Klasse eine abstrakte Methode besitzt, dann ist sie auch selbst abstrakt. |
| | X | Jedes Modul verbirgt eine wichtige Entwurfsentscheidung hinter einer Schnittstelle, die sich bei einer Änderung der Entscheidung mitändert. |
| | X | Die Zuordnung von Modulen zu Schichten in der Schichtenarchitektur ist immer eindeutig. |
| X | | Bei Rekursionseliminierung ist eine Transformation in die iterative Form nicht mehr schwierig, sobald eine rechtsrekursive Form vorliegt. |
| | X | Kontrollflussorientierte Tests und datenflussorientierte Tests gehören zu der statischen Analyse von Programmen. |
| | X | Die einfache Bedingungsüberdeckung fordert, dass die atomaren Bedingungen mit allen möglichen Kombination der Wahrheitswerte W und F belegt werden. |
| | X | Im "Synchronisiere und Stabilisiere-Model" ist die Priorisierung nach Funktionen nicht möglich. |

b.) Welche vier Arten von Anforderungen gibt es bei der Anforderungsermittlung? (2 P)

- Funktionale Anforderungen
- Nicht funktionale Anforderungen
- Sicherheitsanforderungen (Safety/security requirements )
- Einschränkungen (Constraints or Pseudo requirements )

c.) Sie entwerfen ein Modul nach dem Geheimnisprinzip. Nennen Sie fünf Aspekte, die Sie dabei möglicherweise verbergen wollen. (2,5 P)

- Implementierung von Datenstrukturen und Operationen an diesen Datenstrukturen (abstract data types)
- Maschinennahe Details (z. B. Gerätetreiber, Steuerung von Ein-/Ausgabe, Zeichencodes und deren Ordnung, Speicherwortgröße etc.)
- Betriebssystemnahe Details (Ein-/Ausgabeschnittstellen, Dateiformate, Netzwerkprotokolle, Kommandosprache, etc.)
- Grundsoftware wie Datenbanken, Oberflächen-Bibliotheken, o.ä.
- Ein-/Ausgabeformate

- Benutzungsschnittstellen
- Text von Dialogen und Fehlermeldungen (Sprache!)
- Größe von Datenstrukturen
- Reihenfolge der Verarbeitung

d.) Nennen Sie vier Techniken, die bei der Optimierung auf Ebene der „Algorithmen und Datenstrukturen“ zum Einsatz kommen. (2 P)

- Speicherung von Zwischenergebnissen statt Neuberechnung
- Vorverarbeitung von Daten
- Teile und Herrsche
- Dynamisches Programmieren

e.) Definieren Sie die Begriffe Testobjekt, Testfall und Testtreiber. (3 P)

- Die zu überprüfende SW-Komponente oder Konfiguration wird Testling, Prüfling oder **Testobjekt** genannt.
- Ein **Testfall** besteht aus einem Satz von Daten für die Ausführung eines Teils oder des ganzen Testlings.
- Ein **Testtreiber** oder Testrahmen versorgen Testlinge mit Testfällen und stoßen die Ausführung der Testlinge an (interaktiv oder selbsttätig).

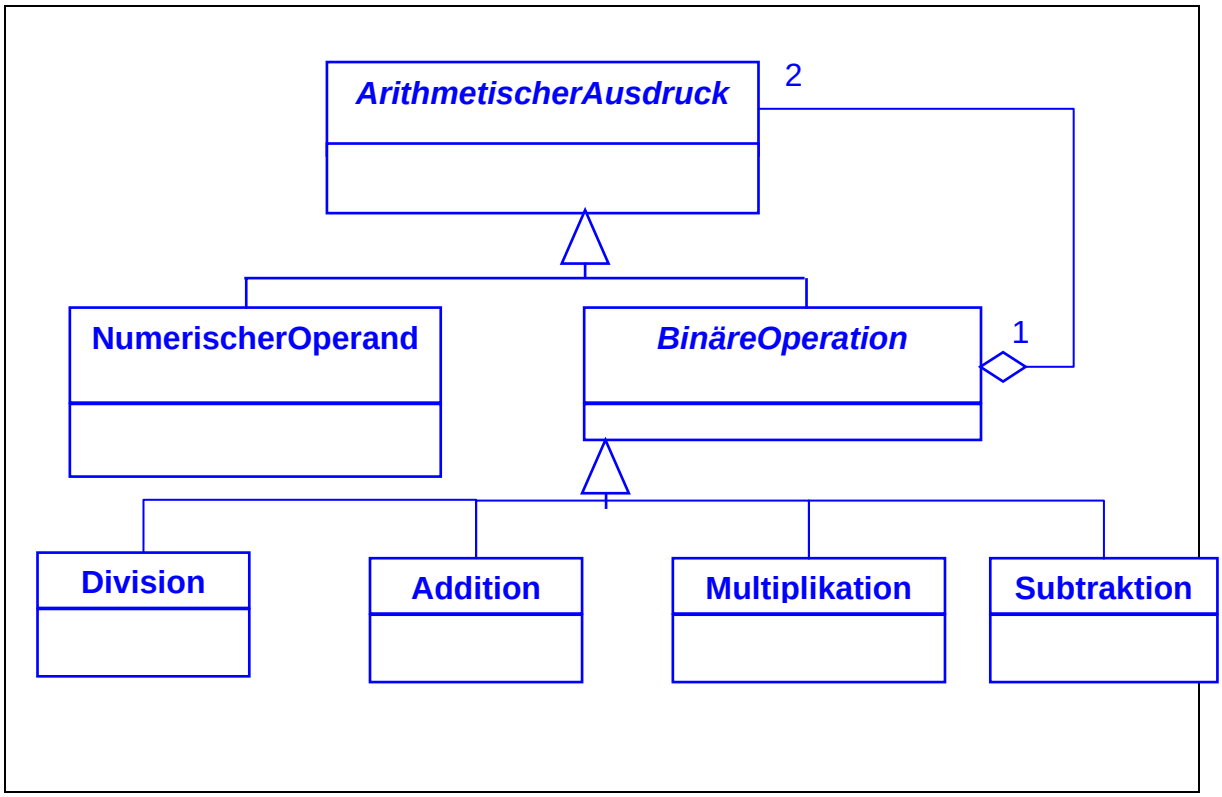
f.) Nennen Sie drei Kategorien von Tätigkeiten in der Wartungs- und Pflegephase. (1,5 P)

4 Kategorien der Wartungs- & Pflege
korrektive Tätigkeiten
- **Stabilisierung** / Korrektur
- **Optimierung** / Leistungsverbesserung

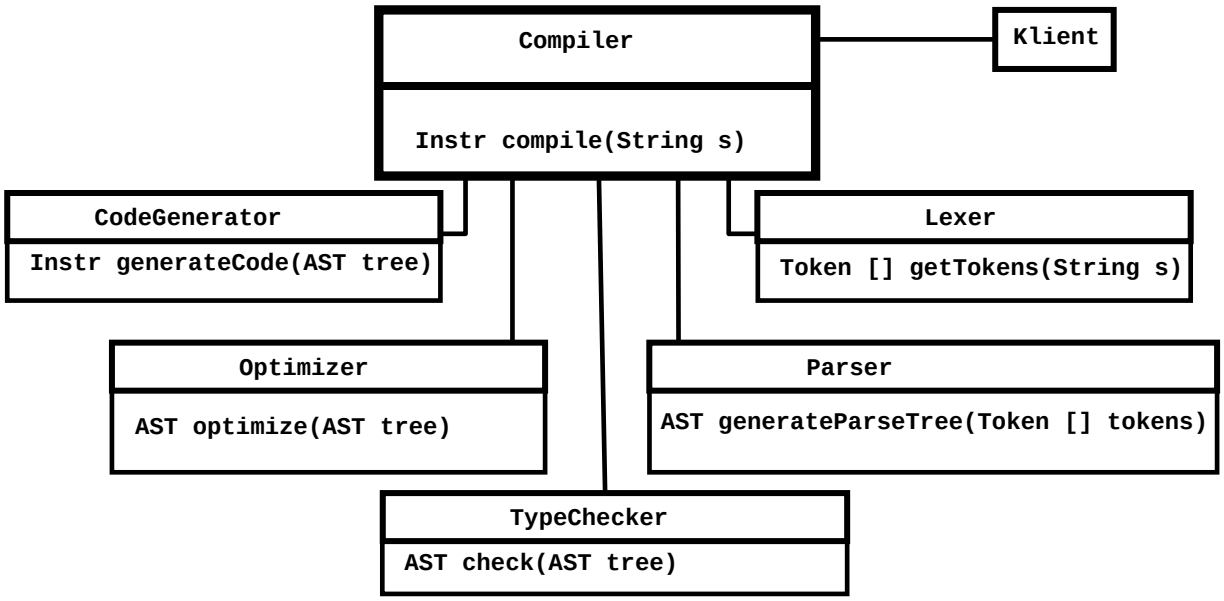
progressive Tätigkeiten
- **Anpassung** / Änderung
- **Erweiterung**.

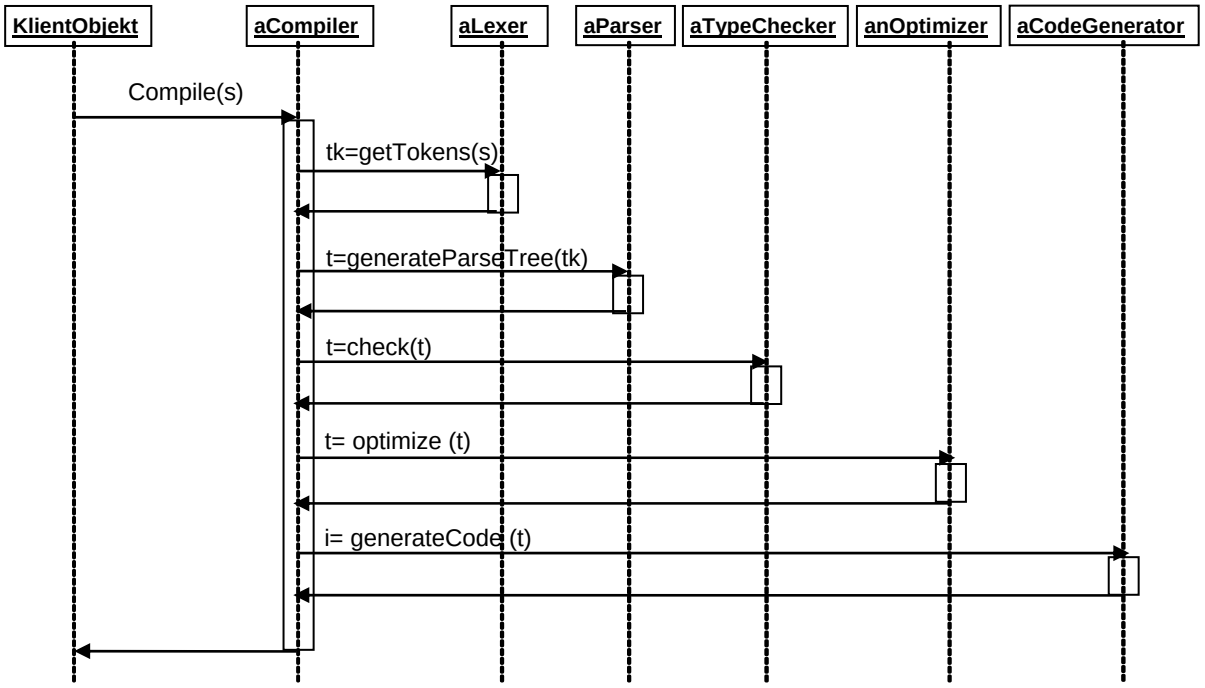
**Aufgabe 2:** Entwurfsmuster (5+3+3= 11P)

a.) Ein binärer arithmetischer Ausdruck enthält einen Operanden, einen Operator (+ - * /) und einen anderen Operanden. Die Operanden können entweder eine Zahl sein oder selbst wieder ein anderer binärer arithmetischer Ausdruck. Entwerfen Sie mit Hilfe eines Entwurfsmusters ein Klassendiagramm, um die Bestandteile eines binären arithmetischen Ausdrucks zu einer Baumstruktur zusammenzufügen und seine Bestands-Hierarchien zu repräsentieren. Nennen Sie das verwendete Entwurfsmuster. Hinweis: z.B. 2 + 3 und (2 + 3) + (4 * 6) sind beide gültige arithmetische Ausdrücke. (5 P)
Verwendetes Entwurfsmuster: **Kompositum**




b.) Gegeben sind das folgende Klassendiagramm und Sequenzdiagramm:






Welches Entwurfsmuster wäre für den oben stehenden Entwurf geeignet? Zu welcher Kategorie gehört dieses Muster und welchen Zweck erfüllt es im Allgemeinen? (3 P)

Die Fassade. (1P) Bequemlichkeits-Muster (1P) Die Compiler-Klasse bietet eine einheitliche Schnittsstelle, welche die Benutzung des Subsystems vereinfacht. (1P) (oder Fließband,Kategorie: Entkopplungs-Muster)

c.) Implementieren Sie nun die Methode c*ompile()* der Compiler-Klasse in Java. Achten Sie auf die Rolle der Compiler-Klasse in dem Klassendiagramm und dem Sequenzdiagramm. (3 P)

```
public class Compiler  {
        private Lexer aLexer = new Lexer();
        private Parser aParser = new Parser();
        private TypeChecker aTypeChecker = new TypeChecker ();
        private CodeGenerator aCodeGenerator = new CodeGenerator();
        private Optimizer anOptimizer = new Optimizer();
        public Instr compile(String s)  {
```

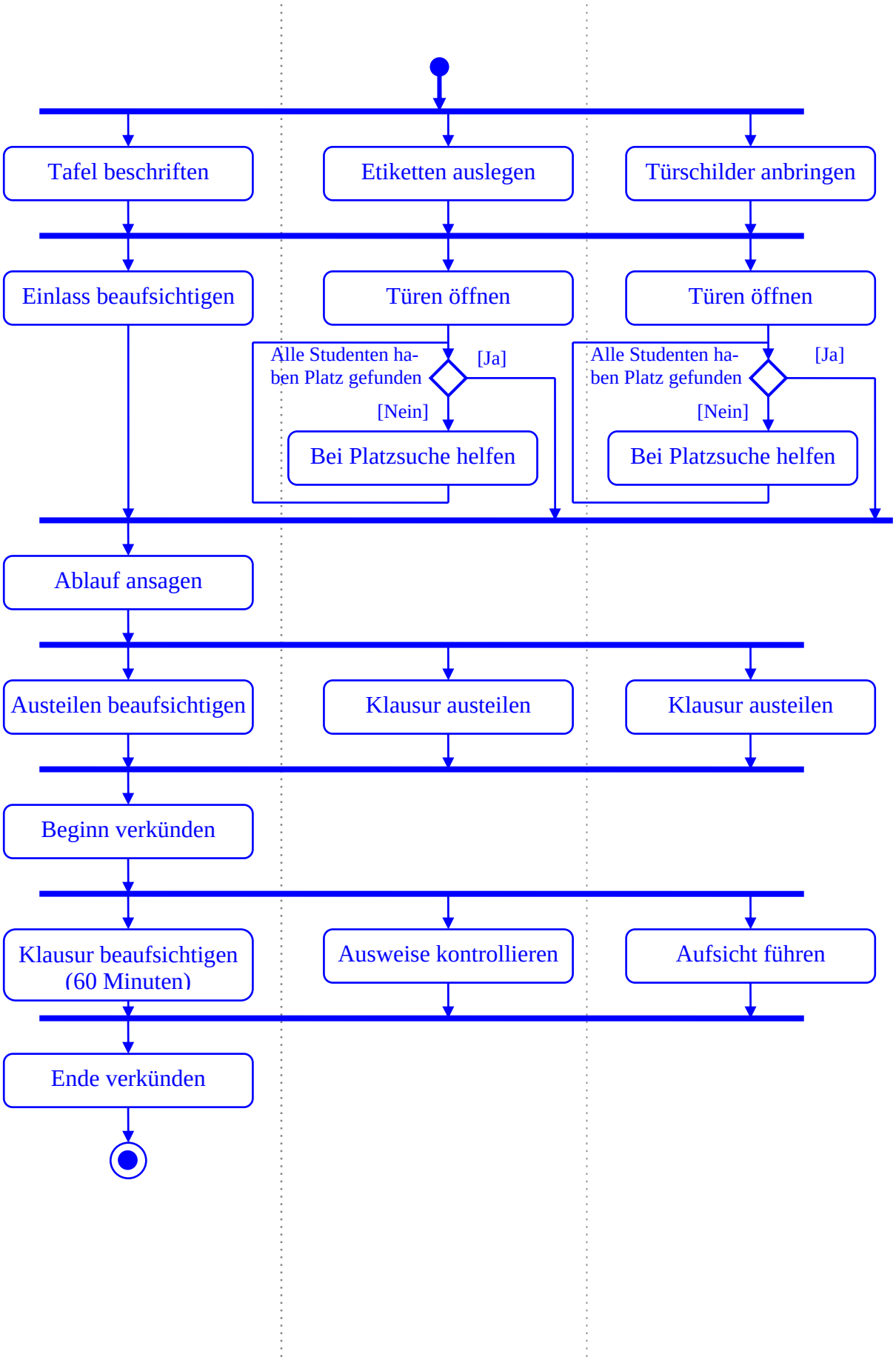
```
Token [] tokens = aLexer.getToken(s);
AST tree = aParser.generateParseTree(tokens);
tree = aTypeChecker.check(tree);
tree = anOptimizer.optimize(tree);
Instr code = aCodeGenerator.generateCode(tree);
return code;

(-0,5 für jeden Fehler)
```

```
        }
}
```

## Aufgabe 3: Aktivitätsdiagramm (11P)

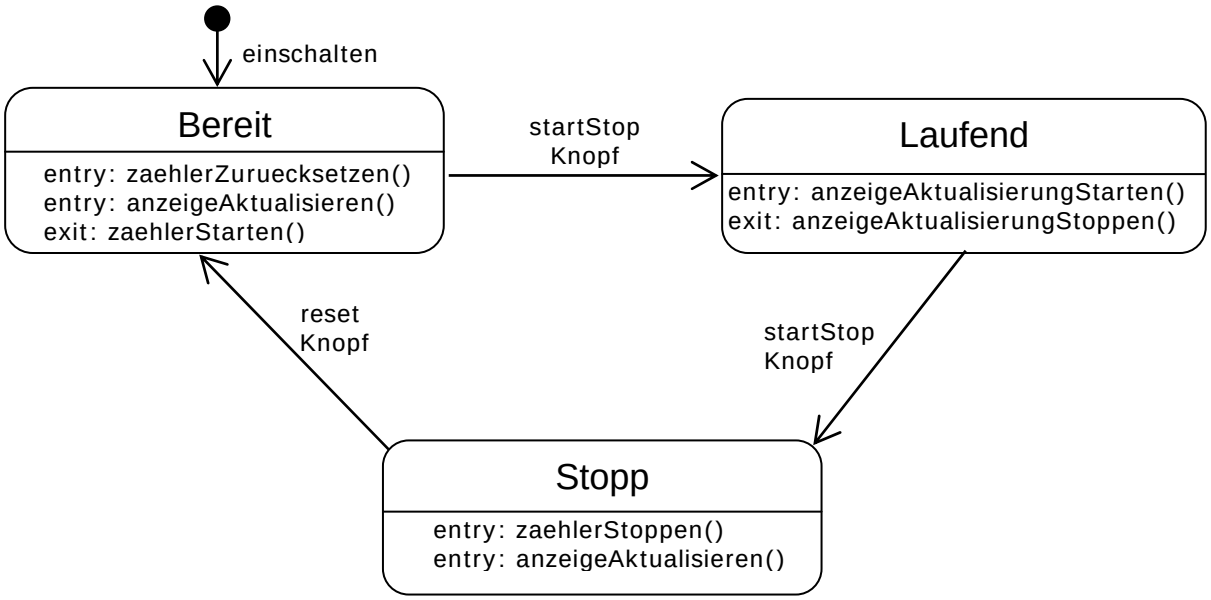
Entwerfen Sie ein Aktivitätsdiagramm, das die Durchführung einer Klausur beschreibt. Halten Sie sich dabei so eng wie möglich an die nachfolgende Beschreibung und achten Sie darauf, parallele Aktivitäten korrekt zu synchronisieren. Beginnen Sie mit der Modellierung der Aktivitäten nach *Betreten des Hörsaals*.
Hinweis: Aktivitäten der Studenten sind nicht zu modellieren.

> Die Klausuraufsicht wird von drei Personen durchgeführt, die alle Mitarbeiter der Universität sein müssen: Der Hauptverantwortliche (A1) wird von zwei Helfern (A2 und A3) unterstützt, alle drei tragen Uhren.
> *Nach Betreten des Hörsaals beginnt A1 die Tafel zu beschriften, während einer seiner Helfer die Etiketten auslegt und der andere die mitgebrachten Türschilder („Bitte nicht stören! Klausur!“) an den Türen anbringt. Auf den Etiketten stehen der Vor- und Nachname jedes Studenten, seine Matrikelnummer und eine fortlaufende Nummer, die später die Verwaltung der Klausur vereinfacht.*
> *Sind diese Aufgaben erledigt, kann der Einlass beginnen. Dazu öffnen die Helfer die Türen des Hörsaals und A1 beaufsichtigt den Einlass. Solange noch nicht alle Studenten ihren Platz gefunden haben, helfen A2 und A3 bei der Platzsuche.*
> *Der Hauptverantwortliche A1 wartet ab, bis alle ihre Plätze gefunden haben. Dann erklärt er den Ablauf, beaufsichtigt das Austeilen der Klausurblätter und verkündet den Beginn der Klausur. Anschließend beaufsichtigt er 60 Minuten lang die Klausur und sagt das Ende der Klausur an.*
> *Nachdem A1 den Ablauf der Klausur angesagt hat, sind A2 und A3 für das Austeilen der Klausur, die anschließende Aufsicht zuständig. Während der Bearbeitungszeit geht zusätzlich einer von beiden herum und kontrolliert die Studentenausweise, während der andere Aufsicht führt.*
> *Während der Klausur verlässt keine Aufsicht den Hörsaal.*


Tafel beschriften
Etiketten auslegen
Türschilder anbringen
Einlass beaufsichtigen
Türen öffnen
Türen öffnen
Alle Studenten haben Platz gefunden
[Ja]
[Nein]
Bei Platzsuche helfen
Alle Studenten haben Platz gefunden
[Ja]
[Nein]
Bei Platzsuche helfen
Ablauf ansagen
Austeilen beaufsichtigen
Klausur austeilen
Klausur austeilen
Beginn verkünden
Klausur beaufsichtigen (60 Minuten)
Ausweise kontrollieren
Aufsicht führen
Ende verkünden

## **Aufgabe 4:** Abbildung von UML auf Code (12 P)

Das folgende Zustandsdiagramm veranschaulicht die Zustände einer Stoppuhr. Sie kann sich in einem der folgenden Zustände befinden: *Bereit*, *Laufend* oder *Stopp*. Vervollständigen Sie den Java-Code für das gegebene UML-Diagramm. Implementieren Sie den Code so, dass sie möglichst genau das durch das Zustandsdiagramm spezifizierte Verhalten an den Tag legen. Verwenden Sie hierfür die Lücken der Vorlage auf dieser und der folgenden Seite. Achten Sie auf korrekte Java-Syntax!
Hinweis: Sobald ein Knopf gedrückt wird, wird die zuständige Methode (z.B. startStop-Knopf()) durch den Event-Handler ausgeführt.




```
public class StoppUhr {

      private enum Zustand {
            BEREIT, LAUFEND, STOPP
      }
      private Zustand aktuellerZustand;

      public StoppUhr() {
            aktuellerZustand = Zustand.BEREIT;
      }
      private void zaehlerZuruecksetzen() {…}
      private void anzeigeAktualisieren() {…}
      private void zaehlerStarten() {…}
      private void anzeigeAktualisierungStarten () {…}
      private void anzeigeAktualisierungStoppen () {…}
      private void zaehlerStoppen() {…}
      public void einschalten() {…}

      public void startStopKnopf() {

            switch (aktuellerZustand) {
```

```
case BEREIT: // Zustand Bereit
     zaehlerStarten(); // Exit-Aktion
     anzeigeAktualisierungStarten (); // Entry-Aktion
     aktuellerZustand = Zustand.LAUFEND; // Übergang
```

```
            break;

        case LAUFEND: // Zustand Laufend
            anzeigeAktualisierungStoppen (); // Exit-Aktion
            zaehlerStoppen(); // Entry-Aktion
            anzeigeAktualisieren(); // Entry-Aktion
            aktuellerZustand = Zustand.STOPP; // Übergang
            break;

        case STOPP: // Zustand Stopp
            break;
        // oder default: break; //anstelle des letzten Falls

        //8P
```

```
    }
}

public void resetKnopf() {
```

```
        if (aktuellerZustand == Zustand.STOPP) {
            zaehlerZuruecksetzen(); // Entry-Aktion
            anzeigeAktualisieren(); // Entry-Aktion
            aktuellerZustand = Zustand.BEREIT; // Übergang
        }



        //4P
```

```
    }
} (-1P für Semantikfehler und -0,5 für Syntax Fehler)
```

**Aufgabe 5:** Testen (6+3+1= 10 P)

Gegeben ist die folgende Java-Funktion. Es wandelt eine Binärzahl in eine Dezimalzahl um.

```
public static long wandleDezimalZahl(String binaer) {
    long zahl = 0;
    if (binaer != null) {
      for (int i = 0; i < binaer.length(); i++) {
        char zeichen = binaer.charAt(i);
        zahl *= 2;
        if (zeichen == '1') {
          zahl++;
        } else if (zeichen != '0') {
          zahl = -1;
          break;
        }
      }
    }
    return zahl;
}
```

a.) Erstellen Sie auf der folgenden Seite den Kontrollflussgraphen der Funktion *wandleDezimalZahl()*. Wenden Sie dabei das aus der Vorlesung bekannte Verfahren an. (6 P)

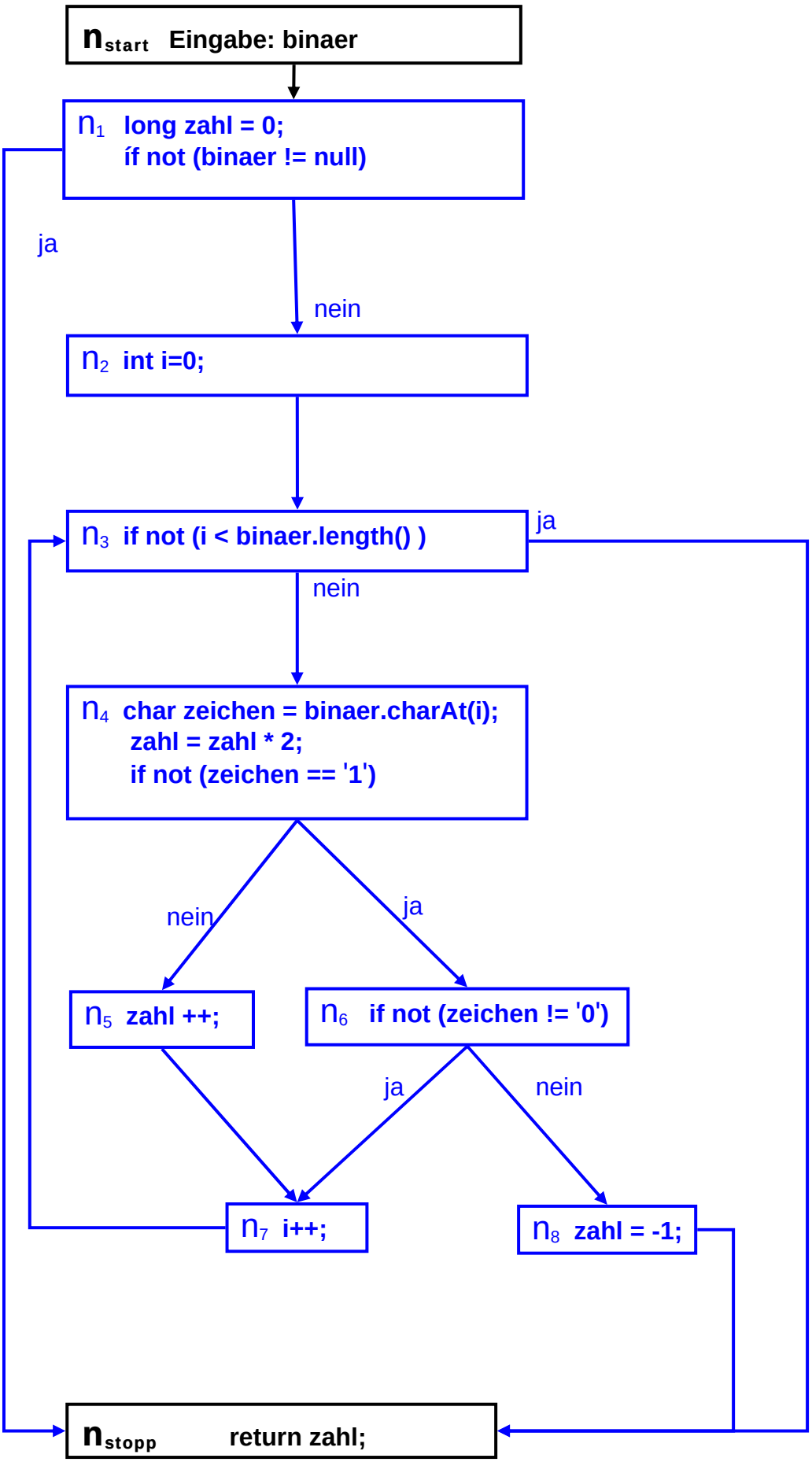
Grob: 2P für „for", 3P für „if-else" und 1P für den Rest

b.) Erstellen Sie zwei minimale Testfall-Mengen: Eine, für die Anweisungsüberdeckung und eine andere für die minimale Zweigüberdeckung des obigen Programms. Begründen Sie Ihre Antworten. (3 P)

Anweisungsüberdeckung (1P) : {"1x"}, x!= { 0,1} alle Anweisungen sind durchlaufen: $n_{start}$, $n_1$, $n_2$, $n_3$, $n_4$, …, $n_8$, $n_{stopp}$
Zweigüberdeckung (2P): {null, "10", "x"} oder {null, "10x", ""} (0,5 Punkt für jeden Testfall) alle Zweige sind durchlaufen: ($n_{start}$, $n_1$), ($n_1$, $n_{stopp}$) und ($n_{start}$, $n_1$), ($n_1$, $n_2$), ($n_2$, $n_3$), ($n_3$, $n_4$), ($n_4$, $n_5$), …, ($n_{10}$, $n_{Stopp}$).
(Punkte **nur** mit Begründung)
(0,5 für Begründung)

c.) Gibt es für dieses Programm eine Testfallmenge, die das Kriterium der Pfadüberdeckung erfüllt? Begründen Sie Ihre Antwort. (1 P)

Nein, Pfadüberdeckung ist nicht möglich. Man kann eine beliebig große Anzahl an Pfadausführungen in der for-Schleife haben.
oder: Ja, wegen String-Beschränkung in Java


n start Eingabe: binaer
n1 long zahl = 0;
íf not (binaer != null)
ja
nein
n2 int i=0;
n3 if not (i < binaer.length() )
ja
nein
n4 char zeichen = binaer.charAt(i);
zahl = zahl * 2;
if not (zeichen == '1')
nein
ja
n5 zahl ++;
n6 if not (zeichen != '0')
ja
nein
n7 i++;
n8 zahl = -1;
n stopp return zahl;