Klausur Softwaretechnik

28. Juli 2004

Hier das Namensschild aufkleben.

Aufg.1	Aufg.2	Aufg.3	Aufg.4	Aufg.5	Σ	Note
/15	/12	/11	/13	/9	/60	

Die Klausur besteht aus 3 Seiten und ist geheftet abzugeben. Für die Vollständigkeit nicht gehefteter Klausuren wird keine Verantwortung übernommen.

Bitte beachten Sie, dass in Aufgabe 1 der Aufgabenteil i) nur von Informatikern und der Aufgabenteil j) nur von Informationswirten zu bearbeiten ist!

Aufgabe 1 (Wissensfragen) **(15 Punkte)**

Beantworten Sie folgende Fragen.
Falsche Kreuze geben negative Punkte. Fehlende Kreuze, sowie fehlende oder falsche Freitext-Antworten bewirken nichts. Weniger als 0 Punkte können Sie in keinem Aufgabenteil erreichen.

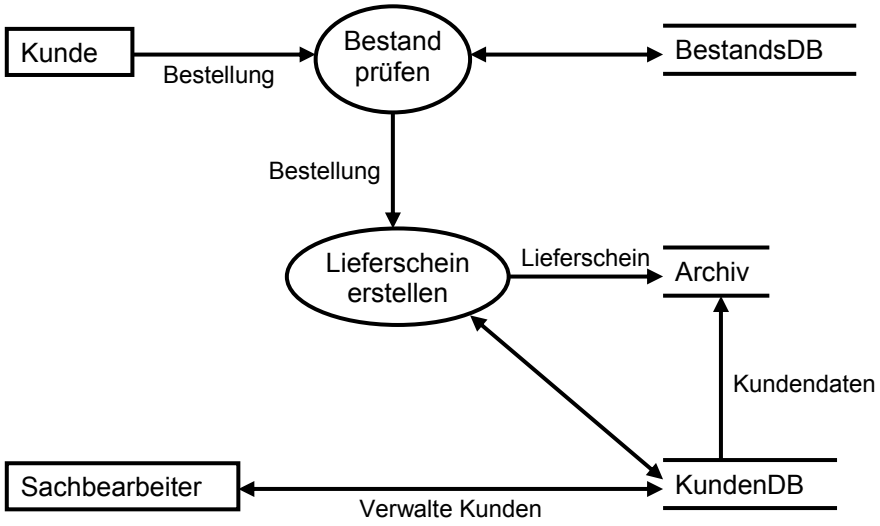
a) Ordnen Sie jedes der folgenden Dokumente der Entwicklungsphase zu, deren Ergebnis es im Idealfall ist. (2 Punkte)

Dokument	Phase		
	Planungs-	Definitions-	Entwurfs-
Benutzerhandbuch	☐	☐	☐
Lastenheft	☐	☐	☐
Projektplan	☐	☐	☐
Spezifikation der Systemkomponenten	☐	☐	☐

b) „Platzhalter“, „Schutzwand“ (firewall) und „Dekorierer“ sind spezialisierte Varianten welches Entwurfsmusters? (0,5 Punkte)

Antwort: _____

c) Das folgende Datenflussdiagramm nach DeMarco enthält 2 syntaktische Fehler. Markieren Sie die Fehler im Diagramm und erklären Sie diese kurz. (1 Punkt)



Fehler 1: _____

_____

Fehler 2: _____

_____

d) „Jedes Modul verbirgt eine wichtige Entwurfsentscheidung hinter einer wohldefinierten Schnittstelle, die sich bei einer Änderung der Entscheidung nicht mitändert.“ Nennen Sie drei Beispiele für Entwurfsentscheidungen, welche hinter einer möglichst änderungsneutralen (Modul-) Schnittstelle verborgen werden sollten: (1,5 Punkte)

1. _____

2. _____

3. _____

e) Wie wurde die „Benutzt-Relation“ in der Vorlesung definiert? (1 Punkt)

Eine Programmkomponente A benutzt eine Programmkomponente B genau dann, wenn _____

_____

_____ .

f) Erstellen Sie einen Funktionsbaum für folgende Aufgabenstellung: (2,5 Punkte)

Die Auftragsabwicklung besteht aus zwei Modulen: In einem werden die Kundendaten und im anderen die damit assoziierten Bestellungen verwaltet. Ein Kundendatensatz besteht aus Namen, Adresse und Kontoverbindung eines Kunden; eine Bestellung ist fest an einen Kunden gekoppelt und enthält Stückzahlen und Produkte. Im Rahmen der Auftragsabwicklung können Kunden aufgenommen, bearbeitet und gelöscht werden. Wird ein Kunde gelöscht, so werden auch die zugehörigen Bestellungen gelöscht. Darüber hinaus können Bestellungen aufgenommen, geändert und gedruckt werden.

g) Entscheiden Sie, ob die folgenden Aussagen richtig oder falsch sind. (3 Punkte)

Richtig/Falsch

□ / □ Mache beim Erstellen von Vererbungsstrukturen eine Klasse B erst dann zur Unterklasse einer Klasse A, wenn gezeigt werden kann, dass jede Instanz von B auch als eine Instanz von A gesehen werden kann.

□ / □ In Petrinetzen gilt: Eine Transition t kann schalten oder „feuern“, wenn mindestens eine Eingabestelle von t mindestens eine Marke enthält.

□ / □ Die Frage nach der Lebendigkeit eines Petrinetzes ist nur im Falle von B/E-Petrinetzen entscheidbar.

□ / □ Im objektorientierten Entwurf behalten die Prinzipien des modularen Entwurfs (Flexibilisierung der Software durch das Geheimnisprinzip) ihre Gültigkeit.

□ / □ Bei der Entwicklung von Software nach dem „Test-First“-Prinzip kann auf Regressionstests verzichtet werden.

□ / □ Wenn ein Datenflussdiagramm sehr groß wird, darf man eine Schnittstelle mehrfach einzeichnen, um die Übersichtlichkeit zu erhöhen.

h) Beschreiben Sie das Merkmal des Entwurfsmusters „Erbauer“, das es von den anderen vorgestellten Erzeugungsmustern unterscheidet. (Punkte gibt es nur für vollständige Sätze.) (1,5 Punkte)

_____

_____

_____

_____

► NUR FÜR INFORMATIKER: ◄

i) Nennen Sie Vor- und Nachteile von mehrfachem (multiplem) Check-Out gegenüber dem einfachen Check-Out beim Konfigurationsmanagement? (2 Punkte)

Vorteile: _____

_____

Nachteile: _____

_____

► NUR FÜR INFORMATIONSWIRTE: ◄

j) Geben Sie zwei Beispiele für „nichtfunktionale Produkt- und Qualitätsanforderungen“, die in der Entwurfsphase zu berücksichtigen sind. (2 Punkte)

1. _____

2. _____

Aufgabe 2 (Entscheidungstabellen) **(12 Punkte)**

Das Preissystem der Deutschen Bahn stand bei seiner Einführung Ende 2002 aufgrund seiner Komplexität in der Kritik. Hier ein Auszug aus den „Plan&Spar"-Bedingungen für Frühbucher:

> *Kunden, die sich zwischen einem und sieben Tagen im Voraus entscheiden, können mit dem „Plan&Spar"-Preis zwischen 10 und 40 Prozent vom Normalpreis sparen. Bei einem Tag Vorverkaufsfrist werden zehn Prozent Rabatt gewährt. Bei drei Tagen im Voraus gibt es 25 Prozent Rabatt, aber nur bei Hin- und Rückfahrt. Ab sieben Tage vor Reiseantritt werden 40 Prozent Nachlass gewährt, allerdings muss zwischen Hin- und Rückfahrt ein Wochenende (eine Nacht von Samstag auf Sonntag) liegen. Die Rabatte gelten immer nur für einen bestimmten Zug, auf den sich der Kunde beim Kauf der Fahrkarte festlegen muss.*

a) Vervollständigen Sie die folgende *begrenzte* **Eintreffer**-Entscheidungstabelle anhand der oben stehenden Beschreibung. Tragen Sie dazu in die Bedingungszeilen „J", „N" oder „–" und in die Aktionszeilen „X" oder „–" ein. Hinweis: Punkte gibt es nur für vollständige Regeln, also ohne „leere" Felder. Eine richtige Lösung kann nur bei Ausnutzung aller Spalten erreicht werden. (4,5 Punkte)

ET1: *Plan&Spar*		R1	R2	R3	R4	R5	R6	R7	R8	R9	Else
B1	1-2 Tage Vorverkauf										
B2	3-6 Tage Vorverkauf										
B3	7-∞ Tage Vorverkauf										
B4	Hin- und Rückfahrt										
B5	Wochenende zwischen Hin- und Rückfahrt										
A1	Normalpreis										-
A2	10% Rabatt										-
A3	25% Rabatt										-
A4	40% Rabatt										-
	Unlogisch	-	-	-	-	-	-	-	-	-	X

b) Vervollständigen Sie die folgende *erweiterte* **Mehrtreffer**-Entscheidungstabelle anhand der oben stehenden Beschreibung. Bei dieser Tabelle dürfen sich die Bedingungen der Regeln überschneiden, es soll jedoch nur eine der Regeln zur Ausführung kommen und zwar diejenige mit der kleinsten Nummer. (Es gelten die gleichen Hinweise wie bei Aufgabenteil a) (4 Punkte)

ET2: *Plan&Spar*		R1	R2	R3	R4	R5
B1		-				
B2	Hin- und Rückfahrt	N				
B3	Wochenende zw. Hin- und Rückfahrt	J				
A1		-				
	Unlogisch	X				

Betrachten Sie die folgende erweiterte Entscheidungstabelle zur Steuerung einer einfachen Klimaautomatik:

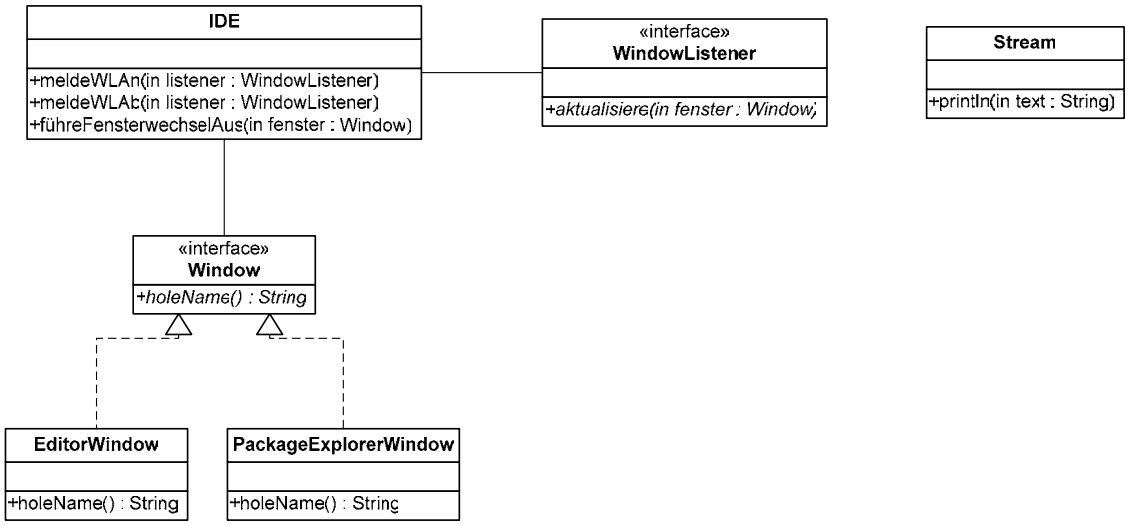
ET3: *KlimaSteuerung*	R1	R2	R3	Else
Uhrzeit	*Tag*	*Tag*	*Nacht*	
Temperatur	< 18	> 22	–	
Heizung einschalten	X	–	–	–
Heizung ausschalten	–	X	X	X
Kühlung einschalten	–	X	–	–
Kühlung ausschalten	X	–	X	X

c) Setzen Sie die Entscheidungstabelle ET3 in ein **Java-Programm** um, welches die gleiche Funktionalität besitzt. Verwenden Sie dazu das folgende Programmskelett und implementieren Sie darin die Methode *steuere*, die einer „Ausführung" der Entscheidungstabelle entsprechen soll. (3,5 Punkte)
Hinweis: Bei der Abfrage der Uhrzeit soll davon ausgegangen werden, dass es entweder Nacht oder Tag ist.

```
public class KlimaSteuerung {
  // Eingabewerte
  private boolean istTag() { … }
  private int temperatur() { … }
  // Aktionen
  private void heizungEin() { … }
  private void heizungAus() { … }
  private void kuehlungEin() { … }
  private void kuehlungAus() { … }
  // Steuerung
  public void steuere() {

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

  }
}
```

Aufgabe 3 (UML, Entwurfsmuster) **(11 Punkte)**

Sie sollen in Java ein Protokollierungswerkzeug für eine Entwicklungsumgebung erstellen, das mitschreibt, welche Fenster innerhalb der Entwicklungsumgebung nacheinander aktiv sind. Nach Studie der API der Entwicklungsumgebung werden Sie auf den *WindowListener* aufmerksam, der immer dann benachrichtigt wird, wenn ein Fensterwechsel stattfindet. Das zugehörige UML-Klassendiagramm sieht wie folgt aus: Die Hauptklasse *IDE* besitzt die Methoden *meldeWLAn*, *meldeWLAb* und *führeFensterwechselAus*. Mit *meldeWLAn* können neue *WindowListener* hinzugefügt und mit *meldeWLAb* wieder gelöscht werden. Wird ein Fenster aktiviert, so wird von der Methode *führeFensterwechselAus* in allen angemeldeten *WindowListener* Objekten die Methode *aktualisiere* aufgerufen.



a) Welches Entwurfsmuster wird hier benutzt?

(1 Punkt)

_____

b) Erweitern Sie obiges Klassendiagramm um die oben beschriebene Protokollfunktionalität, die Sie mit nur einer neuen Klasse *ProtokollListener* implementieren sollen. Diese Klasse soll mit dem in a) angegebenen Entwurfsmuster implementiert werden. Der *ProtokollListener* soll eine Referenz auf ein Objekt vom Typ *Stream* besitzen, welche durch den Konstruktor der Klasse gesetzt wird. Vergessen Sie nicht, alle Methodensignaturen (inkl. Sichtbarkeit, Parameter) des *ProtokollListener* anzugeben. (3,5 Punkte)

c) Ihr *ProtokollListener* soll bei seiner Aktualisierung den Namen des Fensters mit der Methode *println* in den assoziierten *Stream* schreiben. Geben Sie die Signatur und die Implementierung (in Java!) der entsprechenden Methode des *ProtokollListener*s an. (2 Punkte)

_____ {

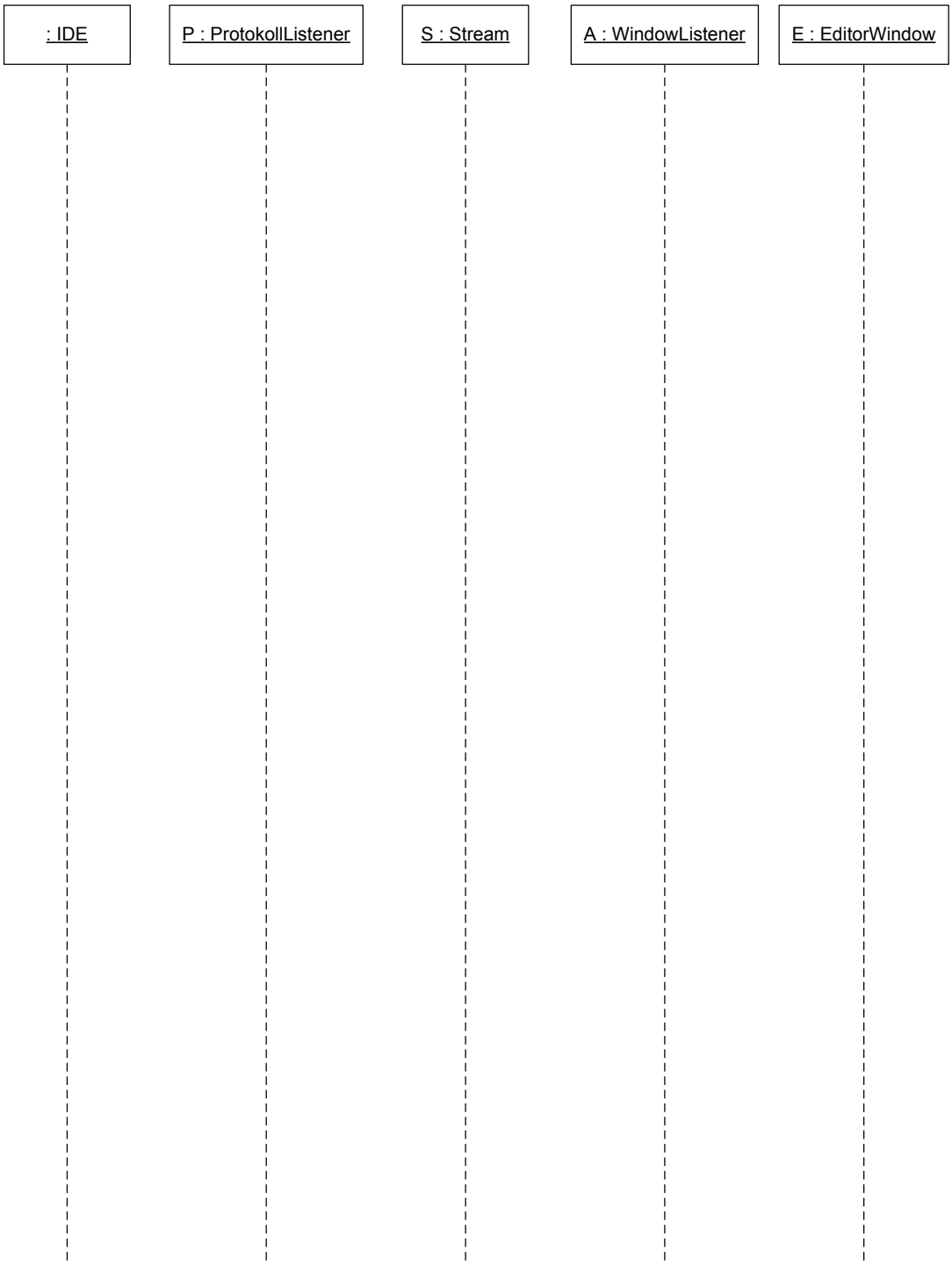
_____

_____

_____

}

d) Erweitern Sie das unten stehende Sequenzdiagramm um folgendes Szenario. (4,5 Punkte) Modellieren Sie die Aufrufsequenzen bis der Kontrollfluss wieder zu *IDE* zurückkehrt.
   - Eine Instanz Ihres *ProtokollListener*s **P** wird bei der *IDE* angemeldet. (**P** ist mit **S**, der Instanz eines *Streams* assoziiert.)
   - Ein anderer *Listener* **A** wird bei *IDE* angemeldet.
   - Die *IDE* erfährt von einem Fensterwechsel durch Aufruf von *führeFensterwechselAus*. Das neue aktive Fenster ist das *EditorWindow* **E**.



Aufgabe 4 (Kontrollflussorientierter Strukturtest) **(13 Punkte)**

Achtung: Falsche Kreuze geben negative Punkte. Fehlende Kreuze, sowie fehlende oder falsche Freitext-Antworten bewirken nichts. Weniger als 0 Punkte können Sie in keinem Aufgabenteil erreichen.

Betrachten Sie folgende Funktion (in Pseudonotation), die den Rabatt für Flüge einer Fluglinie berechnet:

```
1:  double getRabatt(int alter, String ziel, int personen) {
2:    double rabatt = 0.0;

3:    if ((alter>=18) AND (ziel=="New York") AND (personen>=3))
4:      rabatt = rabatt + 0.3;

5:    if ((alter>=0) AND (alter<=3))
6:      rabatt = rabatt + 0.7;

7:    return rabatt;
8:  }
```

Den Ausdruck in der If-Abfrage von Zeile 3 nennen wir P1 und den Ausdruck in der If-Abfrage von Zeile 5 nennen wir P2. P1 und P2 werden komplett ausgewertet, d.h. es gilt keine Kurzauswertung von Ausdrücken.

a) Vervollständigen Sie die untenstehende Schablone des Kontrollflussgraphen der Funktion *getRabatt*. Verwenden Sie dabei für die If-Anweisungen leere Knoten und nummerieren Sie Ihre Knoten in der Reihenfolge in der sie im Quelltext auftreten. (4 Punkte)

1	**Start**
Eingabe: *alter, ziel, personen*	

7	**return** *rabatt*

In den folgenden Aufgabenteilen hat eine Testeingabe die Form eines Tripels (Alter, Ziel, Personen).

b) Welche Bedingungen erfüllt die folgende Testeingabe **`(2, "Adelaide",1)`**?

(2 Punkte)

Richtig/Falsch	Bedingung
□ / □	Anweisungsüberdeckung in Methode *getRabatt*
□ / □	Zweigüberdeckung in Methode *getRabatt*
□ / □	einfache Bedingungsüberdeckung P1
□ / □	einfache Bedingungsüberdeckung P2

c) Welche Bedingungen erfüllen die folgenden Testeingaben **`(3, "New York", 5)`** und **`(30, "New York", 5)`**.

(2 Punkte)

Richtig/Falsch	Bedingung
□ / □	Anweisungsüberdeckung in Methode *getRabatt*
□ / □	Zweigüberdeckung in Methode *getRabatt*
□ / □	einfache Bedingungsüberdeckung P1
□ / □	einfache Bedingungsüberdeckung P2

d) Geben Sie Eingaben in obiger Tripel-Notation für die Methode *getRabatt* an, so dass für P1 das Kriterium „einfache Bedingungsüberdeckung“ erfüllt wird: (1,5 Punkte)

e) Erweitern Sie die Eingabemenge aus a), so dass Ihre Eingaben bei P1 das Kriterium „minimal-mehrfache Bedingungsüberdeckung“ erfüllen. (1,5 Punkte)

f) Kann die einfache Bedingungsüberdeckung für P2 erfüllt werden, wenn für das Alter immer ein positiver ganzzahliger Wert angegeben werden muss? Begründen Sie Ihre Antwort. (Punkte gibt es nur für vollständige Sätze.) (2 Punkte)

Aufgabe 5 (Endliche Automaten) **(9 Punkte)**

In dieser Aufgabe ist ein Programm zu entwickeln, das die Anzahl der Zeichen zählt, die sich innerhalb von Kommentaren in einem Java-Quelltext befinden. (Die Kommentarzeichen sowie die Zeilenumbrüche werden nicht mitgezählt!)
Sie sollen einen **Mealy-Automaten** entwickeln, der mit seinen Ausgaben einen Zähler steuert. Der Automat soll als Eingabe **einzelne Zeichen** verwenden und entweder „−1“, „+1“.oder „0“ ausgeben, um den Zähler zu erniedrigen, zu erhöhen oder unverändert zu lassen. Modellieren Sie neben Ihrem **Startzustand** auch einen **Endzustand**, der beim Erreichen des Dateiendes angenommen wird. Gehen Sie bei dem Entwurf ihres Automaten nicht von syntaktisch korrekten Java-Programmen aus!
Halten Sie sich an den folgenden Ausschnitt aus der „Java Language Specification“:

> *3.7 Kommentare*
>
> *Es gibt zwei Arten von Kommentaren:*
>
> `/*` Text `*/` *Ein herkömmlicher Kommentar: Der gesamte Text wird beginnend bei den ASCII-Zeichen „/*“ bis zu den ASCII-Zeichen „*/“ ignoriert (wie in C und C++).*
>
> `//` Text *Ein einzeiliger Kommentar: Der gesamte Text wird beginnend bei den ASCII-Zeichen „//“ bis zum Ende der Zeile ignoriert (wie in C++).*

Verwenden Sie folgende Mengen von Eingabesymbolen für die Beschreibung des Automaten:

- *ASCII*: alle erlaubten Zeichen. (Kann verkleinert werden. Beispiel: *ASCII-„x“* → alle Zeichen außer dem „x“)
- *newLine*: alle Zeichen, die ein Zeilenende bedeuten.
- *EOF*: Spezielles Symbol, um das Ende der Datei anzuzeigen ($EOF \notin ASCII$).