

PRÜFUNG WS 2018/19

Technologien und Methoden der Softwaresysteme I

Softwaretechnik I

Name:	Vorname:	Matrikel-Nr:
-------	----------	--------------

Prüfungstag: 26.02.2019

Prüfungsdauer: 120 Minuten

Maximalpunktzahl: 120 Punkte

Prüfungsumfang: 5 Aufgaben (27 Seiten)

Hilfsmittel: Alle, außer Kommunikationsgeräte

- Hinweise:**
- 1.) Verlangt sind alle Aufgaben.
 - 2.) Verwenden Sie **keinen Rotstift / Grünstift / Bleistift**.
 - 3.) Tragen Sie auf dem **Deckblatt** Ihren **vollen Namen** und Ihre **Matrikelnummer** ein.
 - 4.) Tragen Sie auf **allen Blättern** Ihre **Initialen** (1. Buchstabe von Vor- und Nachnamen) ein.
 - 5.) Verwenden Sie für Ihre Lösung unbedingt die **jeweiligen Lösungsfelder**.
 - 6.) Geben Sie dieses **Deckblatt zusammen** mit Ihren in der **richtigen Reihenfolge sortierten Aufgabenblättern** ab.

----- Nicht von Studierenden auszufüllen -----

Aufgabe	1	2	3	4	5			
Maximal	30	20	20	20	30			
Erreicht								

Aufgabe 1: Analyse und Entwurf (30 Punkte)

Sie arbeiten als Softwareentwickler am IAS, dieses will einen automatisierten Smoothie-Mixer „SMOOTHIAS“ entwickeln. SMOOTHIAS soll aus verschiedenen Fruchtsäften Erfrischungsgetränke zubereiten können. Als einer der Entwickler sollen Sie zunächst die Analyse der neuen Steuerungssoftware durchführen. Da am IAS innovativ entwickelt wird, soll dies objektorientiert erfolgen.

SMOOTHIAS besteht aus sechs Tanks, die gekühlt werden können (fünf der Tanks sind mit verschiedenen Fruchtsäften gefüllt). Hierfür wurden in jeden Tank ein Kühlaggregat und ein Temperaturfühler eingebaut.

Je eine Pumpe kann den Inhalt aus den Tanks in den Reaktor pumpen. Die gepumpte Menge wird jeweils mit einem Durchfluss-Sensor erfasst, der die Durchflussmenge der Flüssigkeit misst. Die Durchfluss-Sensoren werden vor dem Start des Pumpvorgangs von der Pumpe zurückgesetzt. Daraufhin startet der Pumpvorgang, bis das geforderte Volumen vom Durchfluss-Sensor detektiert wurde und der Pumpvorgang wird gestoppt.

Im Reaktor befindet sich ein Rührer, um den Inhalt durch Rühren zu vermischen. Das fertige Produkt wird mit einem Ventil, am Boden des Reaktors, in einen Becher abgelassen. Der Reaktor verfügt außerdem über eine Heizspule, um die Flüssigkeit im Reaktor erhitzen zu können, sofern das gewählte Rezept dies vorsieht. SMOOTHIAS wird über einen Touchscreen bedient. Am Touchscreen wird das gewünschte Rezept ausgewählt.

Weiterhin steht ein Reinigungsprogramm zur Verfügung, das vom Putzpersonal gestartet werden kann. Im Reinigungsprogramm wird das Reinigungsmittel aus Tank 6 in den Reaktor gepumpt, anschließend erhitzt, dann gerührt und nach der Reinigung abgelassen.

Der Servicetechniker soll über den Touchscreen einzelne Tanks auswählen und anschließend eine individuelle Kühltemperatur einstellen können.

Hinweise:

- **Vereinfachung:** Die Tanks sind genügend groß dimensioniert und beinhalten daher immer genügend Flüssigkeit, um Erfrischungsgetränke herstellen bzw. das Reinigungsprogramm durchführen zu können.
- Der Motor für den Rührvorgang besitzt ein definiertes Programm, das unter Angabe der Rührdauern (in Sekunden) aufgerufen werden kann.
- Das Ventil zum Ablassen der Flüssigkeit aus dem Reaktor in den Becher besitzt genau zwei Zustände (offen und geschlossen).
- Das Heizelement erhitzt die Flüssigkeit im Reaktor auf die Soll-Temperatur und setzt den im Reaktor angebrachten Temperaturfühler ein, um die aktuelle Temperatur der Flüssigkeit im Reaktor zu ermitteln.
- Die Kühlaggregate kühlen die Flüssigkeit im jeweiligen Tank auf die Soll-Temperatur und setzen jeweils ihren Temperaturfühler ein, um die aktuelle Temperatur der Flüssigkeit im Tank zu ermitteln.
- Beim Ablassen der Flüssigkeit aus dem Reaktor soll angenommen werden, dass nach 5 Sekunden der Ablassvorgang abgeschlossen ist. Das Ventil soll daher nach 5 Sekunden geschlossen werden.

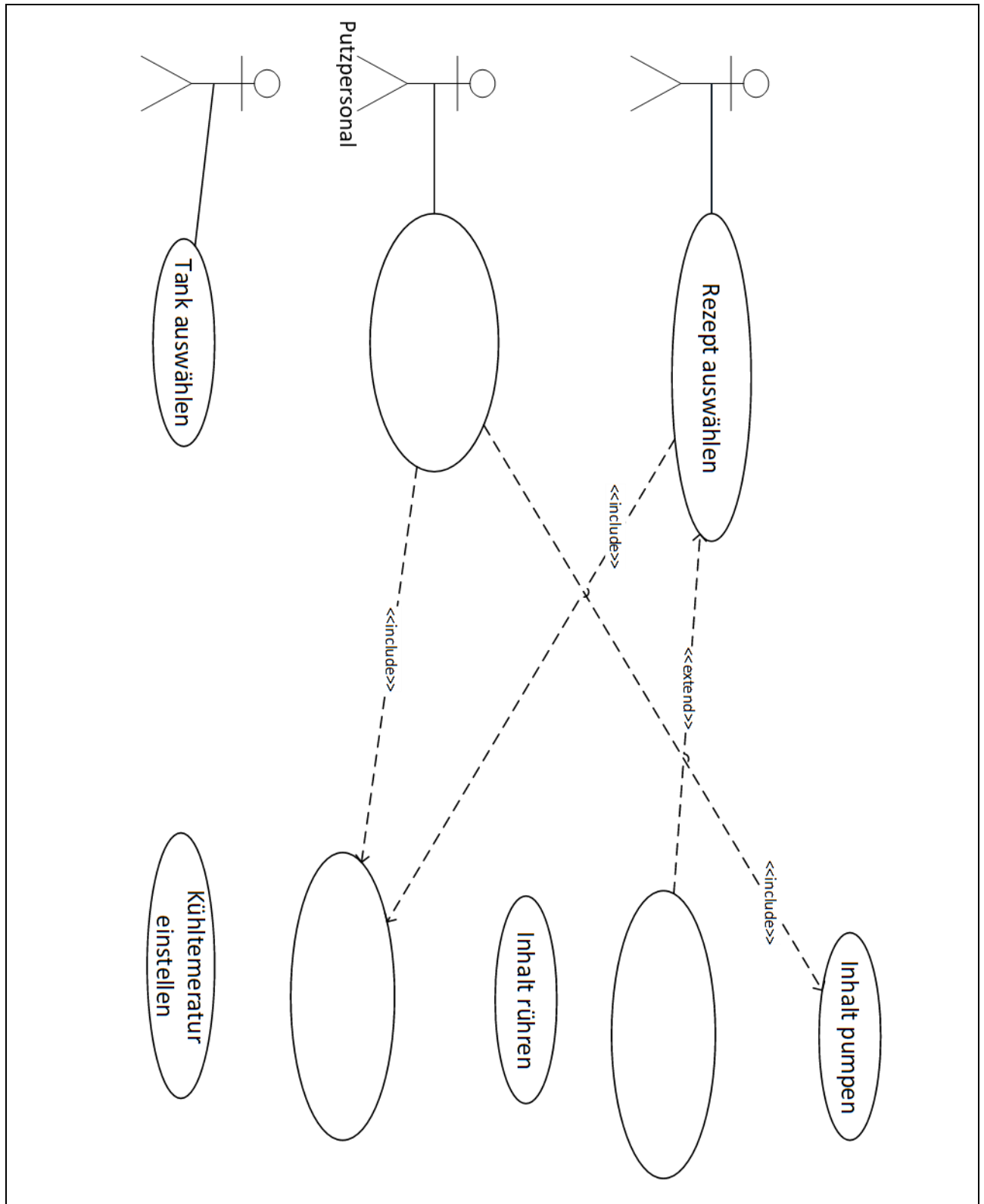
Aufgabe 1.1: Use Cases (6 Punkte)

Die Softwaresteuerung von SMOOTHIAS soll ausgehend von obiger verbaler Beschreibung mit Hilfe von Use Cases modelliert werden. Gehen Sie dabei wie folgt vor:

Vervollständigen Sie das angegebene Use Case Diagramm unter Beachtung der Hinweise um **alle fehlenden Informationen**. Verwenden Sie die, aus der Lehrveranstaltung bekannte, UML-Notation.

Hinweise:

- Ergänzen Sie **keine neuen Anwendungsfälle**.
- Ergänzen Sie **keine neuen Verbindungen zwischen Akteuren und Anwendungsfällen**.
- Ergänzen Sie **weitere Beziehungen zwischen den Anwendungsfällen**, sofern notwendig.



Aufgabe 1.2: Sequenzdiagramm (11 Punkte)

Identifizieren Sie relevante Objekte für die Steuerungssoftware und erstellen Sie ein Sequenzdiagramm nach UML-Notation für das Rezept Maracuja-Kirsch-Smoothie. Die Rezeptsteuerung sieht für dieses Produkt folgenden Ablauf vor:

- Maracuja-Kirsch wird am Touch-Screen gewählt.
- 10cl Kirschsafte werden in den Reaktor gepumpt.
- 20cl Maracujasaft werden in den Reaktor gepumpt.
- Der Rührer rührt 5 Sekunden lang.
- Das Produkt wird in den Becher abgepumpt.

Bilden Sie **alle benötigten physikalischen Entitäten** aus obigem Text (Aufgabe 1) ab. Modellieren Sie **alle Botschaften zwischen Objekten als Operationen** (Methoden).

Hinweis:

- Bedenken Sie die Angabe von Übergabeparametern für Operationen, bei denen diese benötigt werden.
- Nutzen Sie das Lösungsfeld auf der nachfolgenden Seite.

☐

Aufgabe 1.3: Klassendiagramm (13 Punkte)

Zeichnen Sie **basierend auf dem in Aufgabe 1.2 erstellten Sequenzdiagramm** ein Klassendiagramm der Softwaresteuerung von SMOOTHIAS nach UML-Notation. Identifizieren Sie alle relevanten Klassen und ihre Attribute und Methoden, sowie die Beziehungen untereinander mit den entsprechenden Multiplizitäten. Geben Sie die Sichtbarkeiten an.

Hinweise:

- Die Tanks und der Reaktor sollen nun über einen Status verfügen, der abgefragt werden kann, um zu überprüfen, ob die jeweilige Komponente in Ordnung ist. Dieser wird nach jedem Einschalten einmalig abgefragt.
- Die Datentypen müssen nicht berücksichtigt werden.



Aufgabe 2: Implementierung und Test (20 Punkte)

Aufgabe 2.1: Implementierung (12 Punkte)

Gegeben ist der folgende Pseudocode:

```
START
String currentState := "RunUp"
DO
  SWITCH currentState

    CASE „RunUp“:
      RunUpLED.on() //Entry-Actions
      WHILE NOT RunUpCompleted == true
        NOP          // NOP = no operation, do nothing
      ENDWHILE
      currentState := "StandBy"
      RunUpLED.off()

    BREAK

    CASE „StandBy“:
      Motor1.off()
      Motor2.off()
      WHILE NOT StartButton.isPressed()
        NOP
      ENDWHILE
      currentState := "NormalOperation"

    BREAK

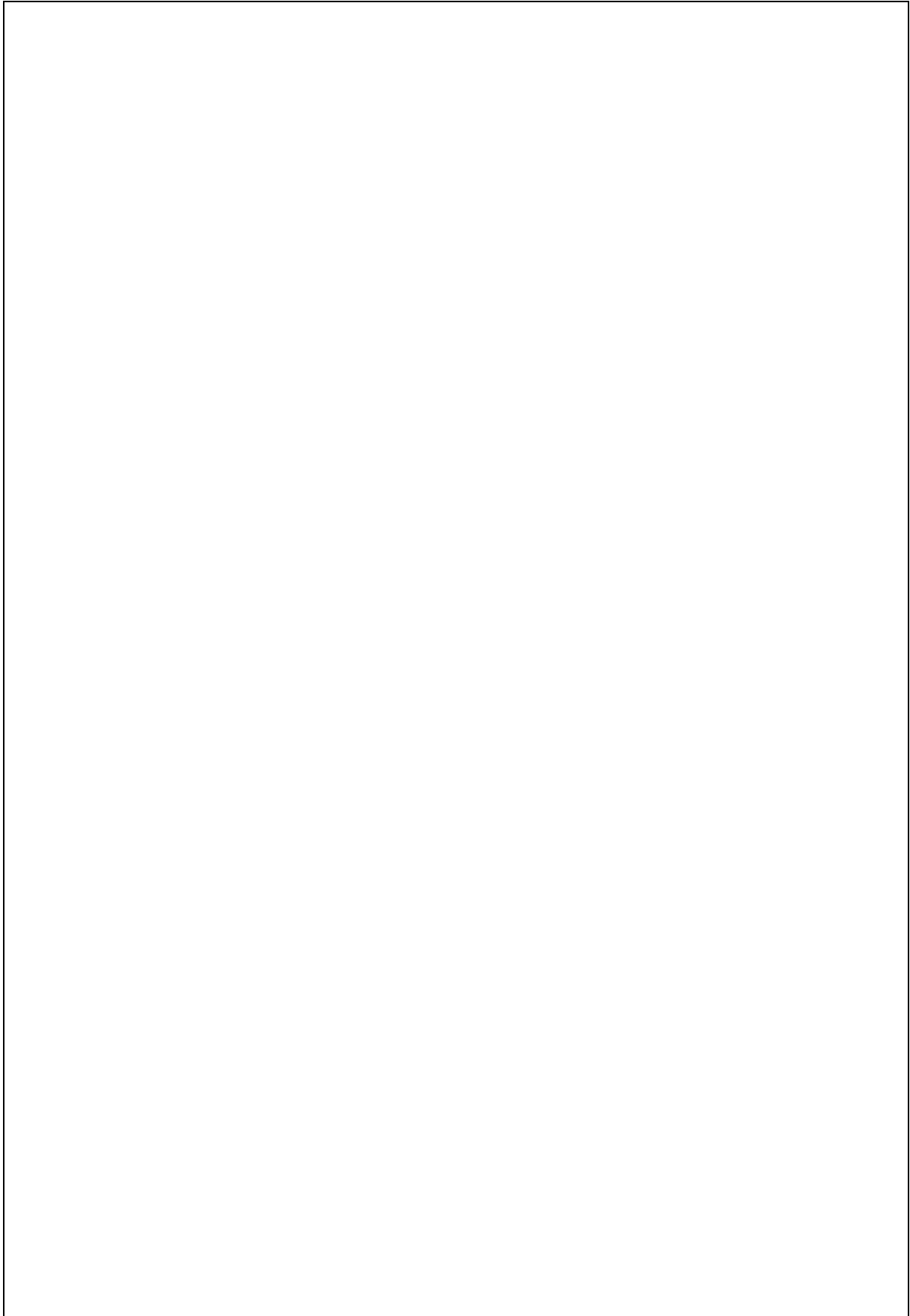
    CASE „NormalOperation“:
      OperationLED.on() //Entry-Actions
      Motor1.on()
      Motor2.on()
      WHILE NOT StopButton.isPressed() AND NOT errorID != 0
        checkStatus()
      ENDWHILE
      IF StopButton.isPressed() THEN
        currentState := "StandBy"
      ENDIF
      IF errorID != 0 THEN
        currentState := "FaultMode"
      ENDIF
      OperationLED.off()

    BREAK

    CASE „FaultMode“:
      Motor1.abruptStop()
      Motor2.abruptStop()
      FaultLED.blink()
      WHILE NOT StopButton.isPressed() AND NOT ResetButton.isPressed()
        NOP
      ENDWHILE
      IF StopButton.isPressed() AND ResetButton.isPressed() THEN
        currentState := "StandBy"
      ENDIF
      IF ResetButton.isPressed() AND StopButton.isPressed() == false THEN
        currentState := "NormalOperation"
      ENDIF
      FaultLED.off()

    BREAK
  WHILE TRUE
END
```

Zeichnen Sie ausgehend von diesem Pseudocode das zugehörige Zustandsübergangsdiagramm (Reverse Engineering), nach der aus der Lehrveranstaltung bekannten Notation, in das Lösungsfeld auf der nächsten Seite.

☐

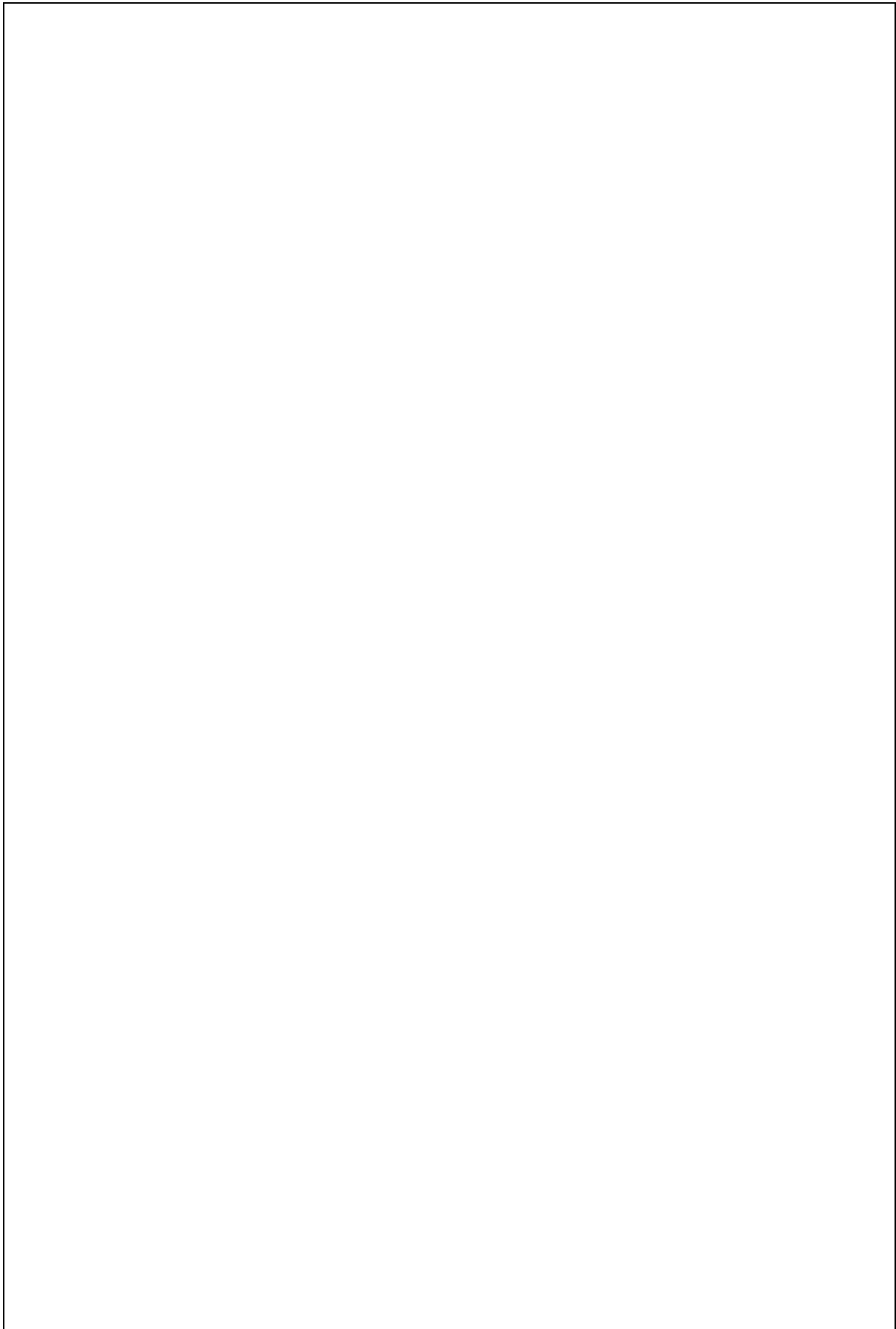
Aufgabe 2.2: Test (8 Punkte)

Sie sollen eine Funktion *calcLargestNumber()* testen, diese ist in Abbildung 1 dargestellt. Die Funktion ermittelt die höchste Zahl von 3 übergebenen Parametern i1, i2 und i3 und gibt diese aus.

```
int calcLargestNumber(int i1, int i2, int i3)      nS
{
    int largestNumber = i1;                        n1
    if (i1 >= i2)                                   n2
    {
        if(i1 >= i3)                               n3
            printf("%d is the largest number.", i1); n4
            largestNumber = i1;                   n5
        else                                       n6
            printf("%d is the largest number.", i3); n7
            largestNumber = i3;                   n8
    }
    else                                          n9
    {
        if(i2 >= i3)                               n10
            printf("%d is the largest number.", i2); n11
            largestNumber = i2;                   n12
        else                                       n13
            printf("%d is the largest number.", i3); n14
            largestNumber = i3;                   n15
    }
    return largestNumber;                        nE
}
```

Abbildung 1: Funktion *calcLargestNumber()* in C-Code

a) Zeichnen Sie den Kontrollflussgraphen für die Funktion *calcLargestNumber()* (5 Punkte)

☐

$\Sigma =$

Aufgabe 3: Softwarearchitektur (20 Punkte)

Aufgabe 3.1: Bausteine und Schnittstellen (12 Punkte)

Gegeben ist eine generische Architektur bestehend aus System, Logic und 5 weiteren Bausteinen (A, B, C, D, E). Sowohl die Visualisierung der Architektur als auch die Beschreibung der Schnittstellen ist dabei nicht vollständig. Ergänzen Sie die Visualisierung der Architektur im nachfolgenden Lösungsfeld, sowie die Beschreibung der Schnittstellen in den Tabellen 2-5. Die Architektur soll genau 7 Schnittstellen besitzen (interfaceX mit $1 \leq X \leq 7$). Übergabeparameter und Rückgabebetyp der Schnittstellen müssen nicht betrachtet werden.

Visualisierung der Architektur (muss ergänzt werden):

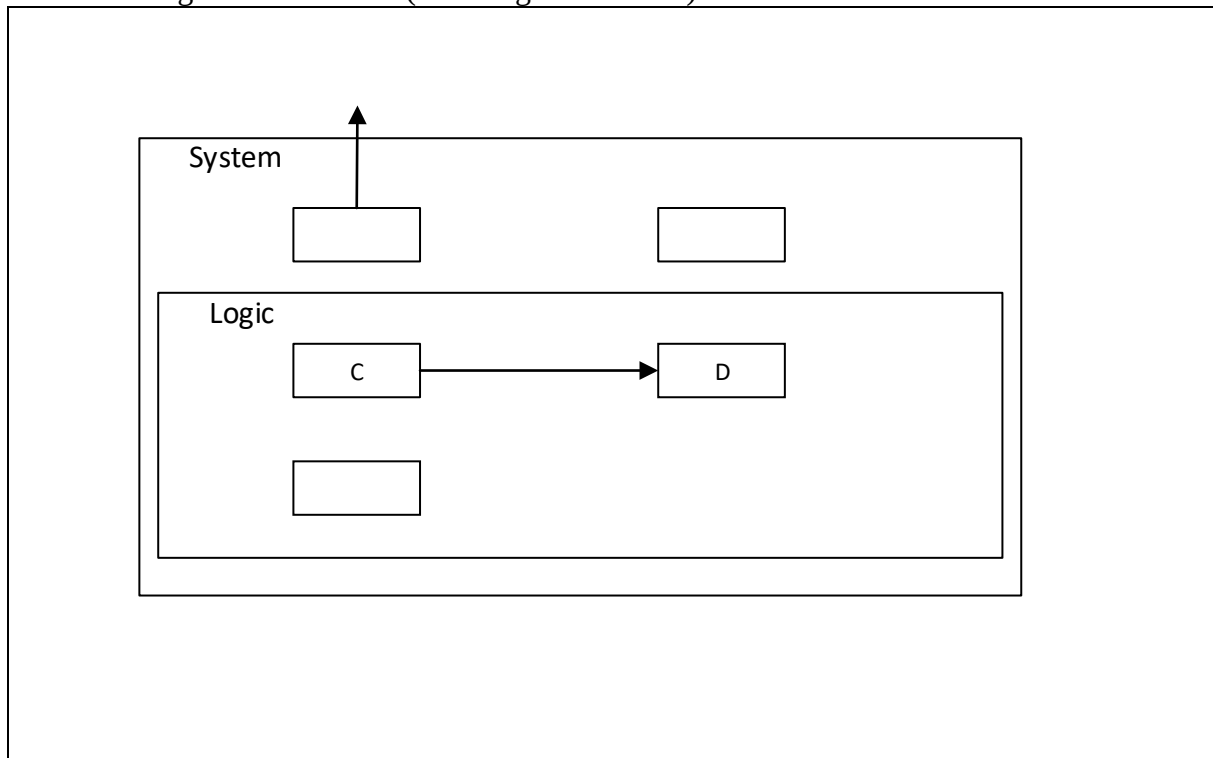


Tabelle 1: Schnittstellen von Baustein A (vollständig)

Funktion	Typ	Schnittstelle zu Baustein(en)
interface1	Imp.	extern
interface2	Imp.	C

Tabelle 2: Schnittstellen von Baustein B (muss ergänzt werden)

Funktion	Typ	Schnittstelle zu Baustein(en)
interface3	Exp.	extern
		D

Tabelle 3: Schnittstellen von Baustein C (muss ergänzt werden)

Funktion	Typ	Schnittstelle zu Baustein(en)
interface5		D
interface6		E
interface7		

Tabelle 4: Schnittstellen von Baustein D (muss ergänzt werden)

Funktion	Typ	Schnittstelle zu Baustein(en)
	Exp.	B
interface6		

Tabelle 5: Schnittstellen von Baustein E (muss ergänzt werden)

Funktion	Typ	Schnittstelle zu Baustein(en)
	Imp.	C, D
interface7	Exp.	C

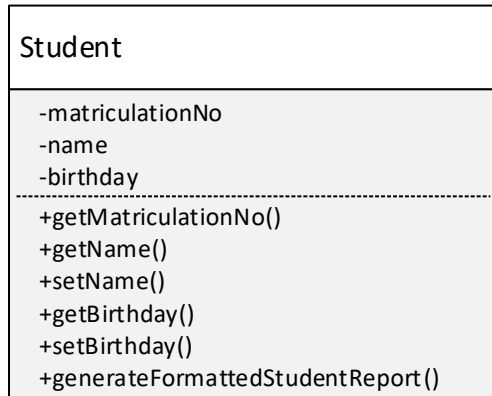
Aufgabe 3.2: Auflösung von zyklischen Abhängigkeiten (4 Punkte)

Identifizieren Sie die zyklischen Abhängigkeiten in der Architektur und lösen Sie diese auf. Es müssen nur die an der zyklischen Abhängigkeit beteiligten Bausteine gezeichnet werden.

Aufgabe 3.3: SOLID-Prinzipien (4 Punkte)

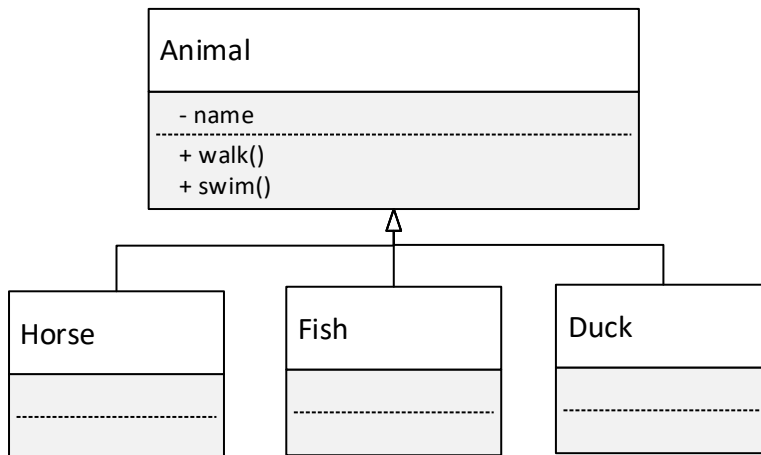
Welches der SOLID-Prinzipien wird von den folgenden Beispielen verletzt und warum?

a) Beispiel Klassendiagramm 1 (2 Punkte)



Verletztes SOLID-Prinzip und Begründung:

b) Beispiel Klassendiagramm 2 (2 Punkte)



Verletztes SOLID-Prinzip und Begründung:

Aufgabe 4: Projektmanagement (20 Punkte)

Sie arbeiten als Projektleiter in einem Softwareunternehmen. Ihr Unternehmen will ein neues IoT-Device auf den Markt bringen. Sie sind mit der Projektplanung und -durchführung betraut. Bei einer ersten Projektleitersitzung wurden die in Tabelle 6 dargestellten Arbeitspakete (AP) identifiziert.

Tabelle 6: Auflistung der Arbeitspakete

AP	Beschreibung	Dauer in Monaten	Vorgänger
A	Anforderungsanalyse	3	-
B	Software-Systemmodell	2	A
C	Software-Architektur	2	B
D	Software-Entwurf	8	C
E	Implementierung	4	D
F	Modultest	4	E
G	Hardware-Entwurf	6	A
H	Hardware-Aufbau	8	G
I	Integration	3	H, F
J	Abnahme	4	I

Aufgabe 4.1: Netzplan (6 Punkte)

Zeichnen Sie einen Netzplan nach der Metra-Potenzial-Methode (MPM) für die in Tabelle 6 angegebenen Arbeitspakete, Dauern und Vorgängerbeziehungen und geben Sie den kritischen Pfad bzw. die kritischen Pfade an.

Berechnen Sie für jedes Arbeitspaket den frühesten und spätesten Anfangs- und End-Zeitpunkt.



Aufgabe 4.2: Balkenplan (6 Punkte)

Nach der Fertigstellung des Projektplans erkennen Sie, dass der Software-Entwurf (AP D) vom Hardware-Entwurf (AP G) abhängig ist.

a) Wie wirkt sich diese neue Abhängigkeit auf den Projektendtermin aus? (1 Punkt)

b) Wie wirkt sich diese neue Abhängigkeit auf den kritischen Pfad aus? (1 Punkt)

- c) Visualisieren Sie den geplanten Projektverlauf, unter Berücksichtigung der neuen Abhängigkeit, in einem Balkenplan. Zeichnen Sie alle Arbeitspakete und deren Abhängigkeiten ein. (4 Punkte)

AP	5	10	15	20	25	30	35	40
A								
B								
C								
D								
E								
F								
G								
H								
I								
J								

Aufgabe 4.3: Berechnung von Puffer (5 Punkte)

Der freie Puffer eines Arbeitspaketes gibt an, wie lange dieses verzögert werden kann, bevor sich ein anderes Arbeitspaket verzögert. Der Gesamtpuffer eines Arbeitspaketes gibt an, wie lange dieses verzögert werden kann, bevor sich der Projektendtermin verzögert.

Berechnen Sie für die folgenden Arbeitspakete, entsprechend dem in Aufgabe 4.2 erstellten Balkenplan, jeweils den freien Puffer und den Gesamtpuffer:

Arbeitspaket	Freier Puffer	Gesamtpuffer
A		
B		
C		
H		
J		

Aufgabe 4.4: Aufwandsabschätzung (3 Punkte)

Eine erste Schätzung hat ergeben, dass das Projekt insgesamt 30000 Quellcodezeilen umfassen wird. Davon müssen 10000 Quellcodezeilen neu erstellt werden. Eine automatisierte Code-Transformation kann nicht eingesetzt werden. Aus Erfahrungen mit früheren Projekten schätzt man eine Größenänderung durch geänderte Anforderungen von 10% ab. Ihr Team ist sehr erfahren bezüglich der Thematik und arbeitet seit mehreren Jahren erfolgreich zusammen und wird daher mit einem Faktor von 0,8 berücksichtigt. Das IoT-Device hat wenige Schnittstellen zum bestehenden IoT-Netzwerk.

Berechnen Sie den Aufwand für das Projekt in Personenmonaten nach dem COCOMO II-Verfahren.

Aufgabe 5: Kurzfragen (30 Punkte)

Wird nicht veröffentlicht!