UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

isp

Software Engineering Wintersemester 2017/18
Prof. Dr. Martin Leucker, Lukas Convent, Torben Scheffel 26.03.2018

Klausur Software Engineering

Aufgaben und Punkte

Die Bearbeitungszeit der Klausur umfasst 90 Minuten. Es gibt 10 Aufgaben mit insgesamt 100 erreichbaren Punkten. Die erreichbaren Punkte pro Aufgabe sind für jede Aufgabe oben rechts angegeben, ebenso die erreichbaren Punkte pro Teilaufgabe (in Klammern dahinter).

Notieren Sie Ihre Lösungen direkt auf dem Aufgabenblatt. Sollte der Platz nicht ausreichen, verwenden Sie zusätzliche Blätter, die Ihnen gestellt werden. Benutzen Sie jede Seite der zusätzlichen Blätter nur für genau eine Aufgabe und notieren Sie Ihren Namen und Ihre Matrikelnummer am oberen Rand dieser zusätzlichen Blätter.

Hilfsmittel

Es sind keine Hilfsmittel zugelassen mit Ausnahme eines eigenhändig (beidseitig) beschriebenen DIN A4-Zettels.

Persönliche Daten

Notieren Sie im Folgenden Ihre persönlichen Daten. Notieren Sie Ihren Namen und Ihre Matrikelnummer außerdem auf jedem zusätzlich gestellten Blatt.

Vorname: _____

Nachname: _____

Matrikelnummer: _____

Viel Erfolg!

Aufgabe 1: Softwarequalität **9 Punkte (3/3/3)**

a) Ein wichtiges Qualitätskriterium für Software ist *Zuverlässigkeit*. Nennen Sie ein anderes Qualitätskriterium, welches dem Kriterium der *Zuverlässigkeit* widersprechen kann und beschreiben Sie kurz, wie dies möglich ist.

b) Nennen Sie drei **nicht**-formale Maßnahmen/Methoden zur Qualitätssicherung von Software.

c) Inwieweit können formale Methoden verwendet werden, um Softwarequalität zu sichern?

Aufgabe 2: Vorgehensmodelle **8 Punkte (4/2/2)**

a) Beschreiben Sie, was inkrementelle Vorgehensmodelle ausmacht.

b) Beschreiben Sie kurz, was ein Sprint im Scrum-Vorgehensmodell ist.

c) Welche Aussagen sind wahr? Es gibt $+0,5$ Punkte für korrekte Kreuze und $-0,5$ Punkte für falsche (jedoch insgesamt mind. 0 Punkte).

	Wahr	Falsch
Im V-Modell steht jeder spezifizierenden Phase eine testende gegenüber.	☐	☐
Extreme Programming ist ein inkrementelles Vorgehensmodell.	☐	☐
Im Spiralmodell finden mehrfach Risikoanalysen statt.	☐	☐
Das Spiralmodell ist ein prototyporientiertes Vorgehensmodell.	☐	☐

Aufgabe 3: Management **10 Punkte (3/4/3)**

a) Ein Mitarbeiter/eine Mitarbeiterin bringt verschiedene Qualifikationen mit. Worin besteht der Unterschied zwischen horizontaler und vertikaler Ausrichtung seiner/ihrer Qualifikationen?

b) Das Vier-Ohren-Modell ist ein Kommunikationsmodell, welches vier Ebenen eines Kommunikationsakts identifiziert. Nennen (oder beschreiben) Sie die vier Ebenen.

c) Aufgrund mehrerer Unzulänglichkeiten bietet sich für ein typisches Legacy-System ein Re-Engineering an. Benennen Sie zwei Risiken, die das Re-Engineering eines Legacy-Systems sowie die Migration auf das neue System mit sich bringen.

Aufgabe 4: Planungsphase **9 Punkte (5/2/2)**

a) Nennen Sie zwei Methoden zur Aufwandsschätzung und beschreiben sie **für eine**, was diese als Grundlage nimmt und wie sie funktioniert.

b) Beschreiben Sie kurz, wovon im Allgemeinen abhängt, wie gut eine Methode zur Aufwandsschätzung in einem Unternehmen funktioniert.

c) Welche Aussagen sind wahr? Es gibt $+0,5$ Punkte für korrekte Kreuze und $-0,5$ Punkte für falsche (jedoch insgesamt mind. 0 Punkte).

	Wahr	Falsch
Ein Projektstrukturplan wird durch Termin- und Kostenplanung beeinflusst.	☐	☐
Ein Projektstrukturplan ist immer ein Baum.	☐	☐
Ein sinnvoller Meilenstein ist zum Beispiel: Die Tests sind zu 75% implementiert.	☐	☐
Ein Gantt-Diagramm dient der Planung des Projektverlaufs.	☐	☐

Aufgabe 5: Netzpläne **12 Punkte (6/2/2/2)**

Im Folgenden soll die Entwicklung eines Systems zur sicheren Kommunikation mit Kunden geplant werden, unter Verwendung eines Netzplans. Für die Planung wird angenommen, dass Team und Ressourcen ausreichend groß bemessen sind.

Die Vorgangsdauer wird in Tagen angegeben und die Modellierung soll mit Tag 1 beginnen. In einer ersten Analyse wurde folgendes Vorgehen als Basis für die Entwicklung gewählt:

- Zum Projektstart (an Tag 1) werden die Datenmodellierung für Kunden definiert (2 Tage) sowie verschiedene Verschlüsselungsalgorithmen evaluiert (3 Tage).
- Aufbauend auf der Datenmodellierung wird das Datenmodell implementiert (1 Tag). Außerdem kann nach Abschluss der Datenmodellierung mit der Implementierung der Eingabemaske (6 Tage) begonnen werden.
- Ist die Evaluation der Verschlüsselungsalgorithmen abgeschlossen, kann mit deren Implementierung (3 Tage) begonnen werden. Ist auch die Datenmodellierung bereits abgeschlossen, kann parallel mit den ersten Datenverschlüsselungstests (2 Tage) begonnen werden.
- Sind die ersten Datenverschlüsselungstests und die Implementierung der Verschlüsselungsalgorithmen abgeschlossen, werden ausführliche Tests der Verschlüsselung durchgeführt (7 Tage).
- Nachdem alle vorangehenden Vorgänge abgeschlossen wurden, gibt es eine Abnahme durch den Kunden (1 Tag), nach welcher das Projekt beendet ist.

Erstellen Sie einen Netzplan wie folgt.

a) Stellen Sie die gegebenen Aufgaben in einem Netzplan (mit gerichteten Kanten) dar. Verwenden Sie einen zusätzlichen Projektstart-Knoten mit einer Dauer von 0 Tagen.

b) Rechnen Sie vorwärts: Berechnen Sie, wann die Vorgänge frühestens begonnen werden können und notieren Sie diese Werte im Netzplan.

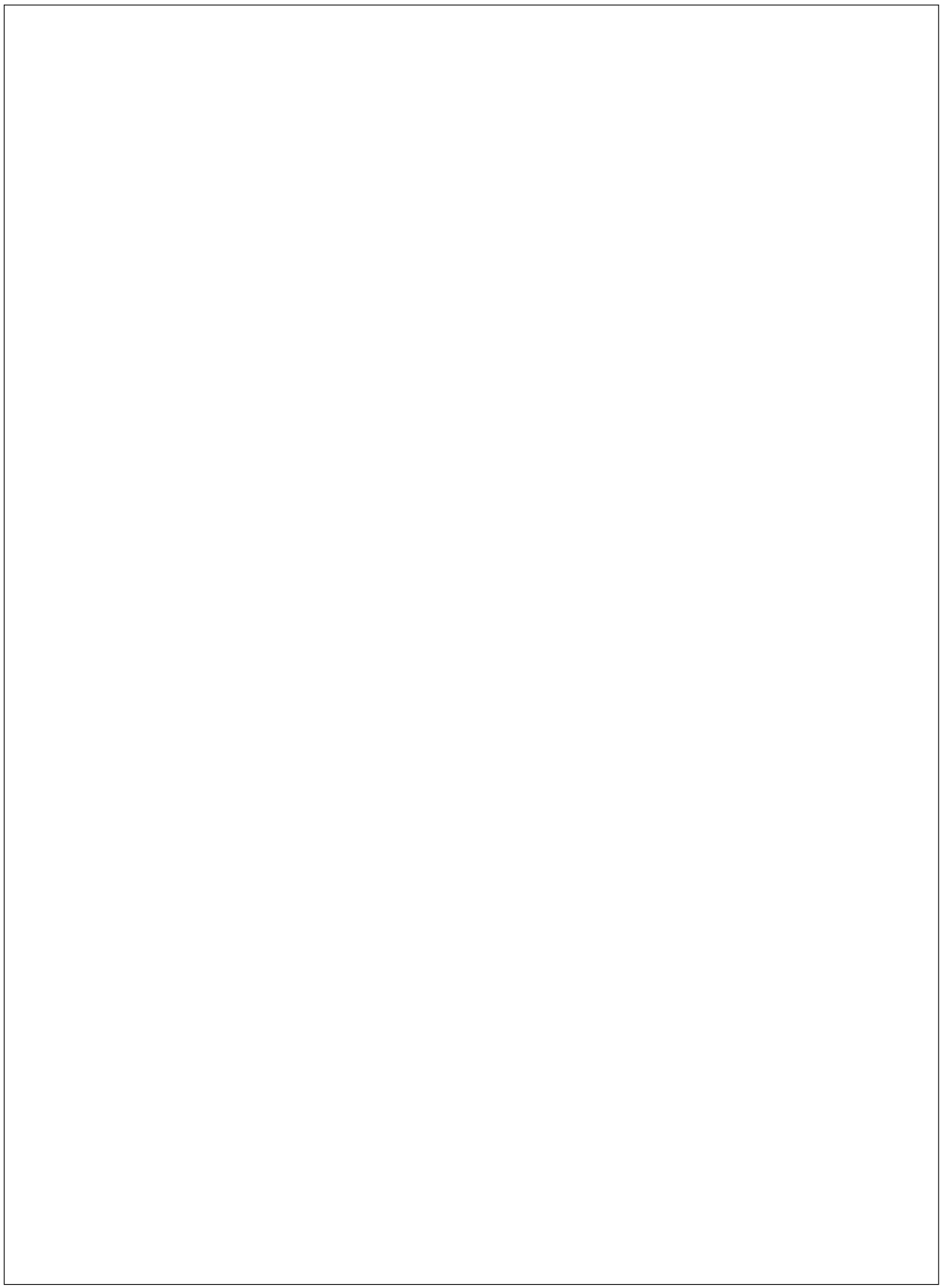
c) Rechnen Sie rückwärts: Berechnen Sie ausgehend vom frühesten Projektende, wann die einzelnen Vorgänge spätestens begonnen werden können (ohne dass sich das früheste Projektende verändert) und notieren Sie diese Werte im Netzplan.

d) Bestimmen Sie sämtliche Pufferzeiten und geben Sie einen kritischen Pfad an.

Verwenden Sie folgende Knotendarstellung:

Vorgang	
frühester Start	spätester Start
Dauer	Puffer

Benutzen Sie aus Platzgründen die nächste Seite im Querformat für die Bearbeitung.



Aufgabe 6: Sequenzdiagramme **9 Punkte**

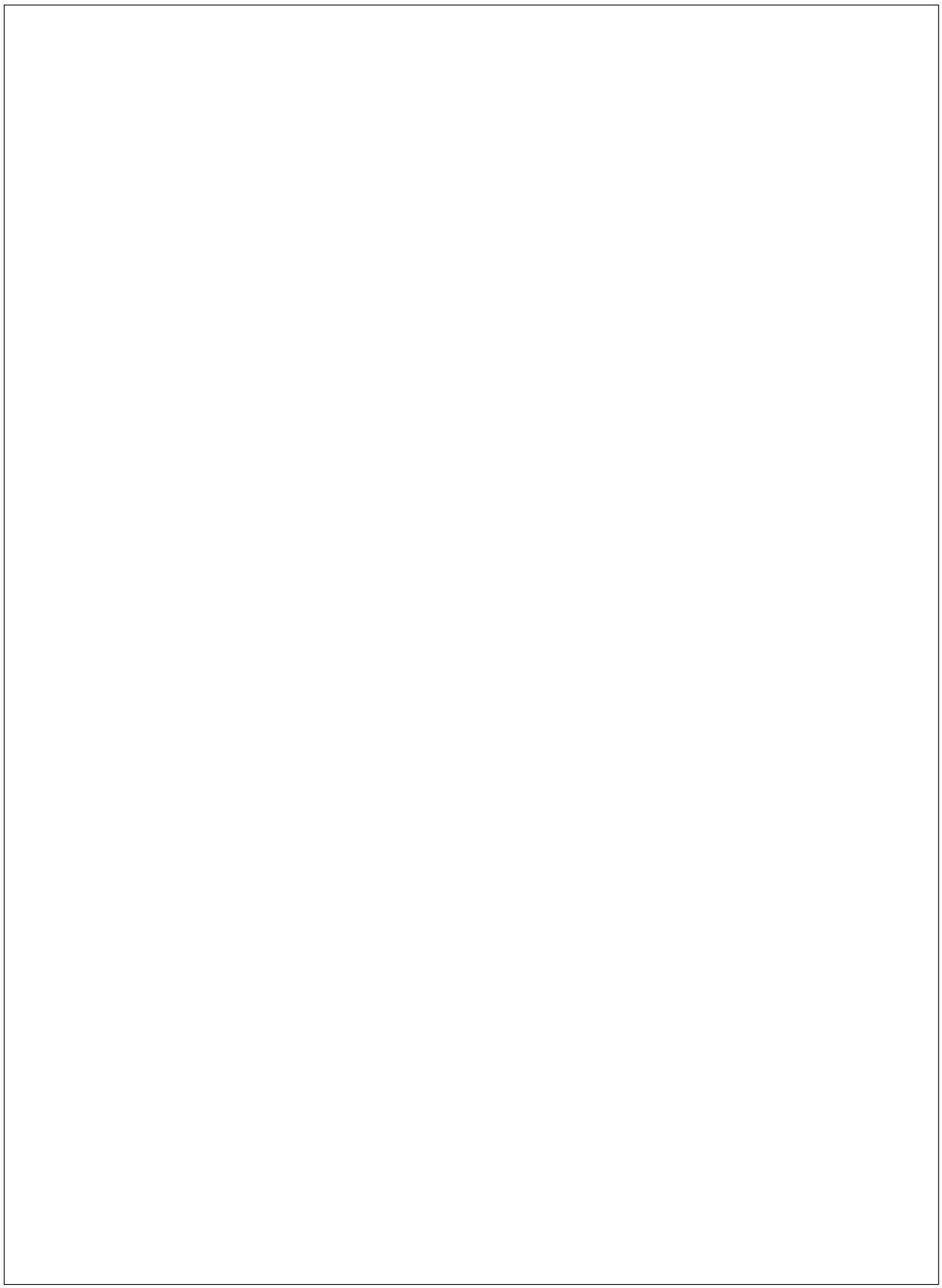
Eine Firma verwaltet ihre Kundenkartei elektronisch. Eine Kundenkartei und ein Kunde wird jeweils als Objekt modelliert. Ebenso gibt es ein Control-Panel als Benutzerschnittstelle, welches ebenfalls als Objekt modelliert wird.

Durch einen Aufruf von außen (MitarbeiterIn drückt auf *Suchen*) wird vom `ControlPanel` ein Suchvorgang ausgelöst. Dieser besteht aus einer Anfrage an die `Kundenkartei`, bei welcher der Suchausdruck übergeben wird. Nachdem die `Kundenkartei` eine Abfrage an die interne Datenbank abgeschlossen hat, wird als Ergebnis die Menge aller Kunden an das `ControlPanel` zurückgegeben, welche zum Suchausdruck gepasst haben.

Nachdem ein `Kunde` ausgewählt wurde, kann diesem durch einen Aufruf von außen (MitarbeiterIn drückt auf *Senden*) eine Nachricht geschickt werden, deren Versand jedoch nicht bestätigt wird.

Zuletzt gibt es noch die Möglichkeit, einen neuen Kunden in die Kartei einzufügen, ausgelöst durch einen Aufruf von außen (MitarbeiterIn drückt auf *Hinzufügen*). Dabei wird zunächst ein neues Kunden-Objekt erzeugt (parametrisiert mit dem Kundennamen), anschließend wird das Objekt über eine Anfrage an die `Kundenkartei` dieser hinzugefügt.

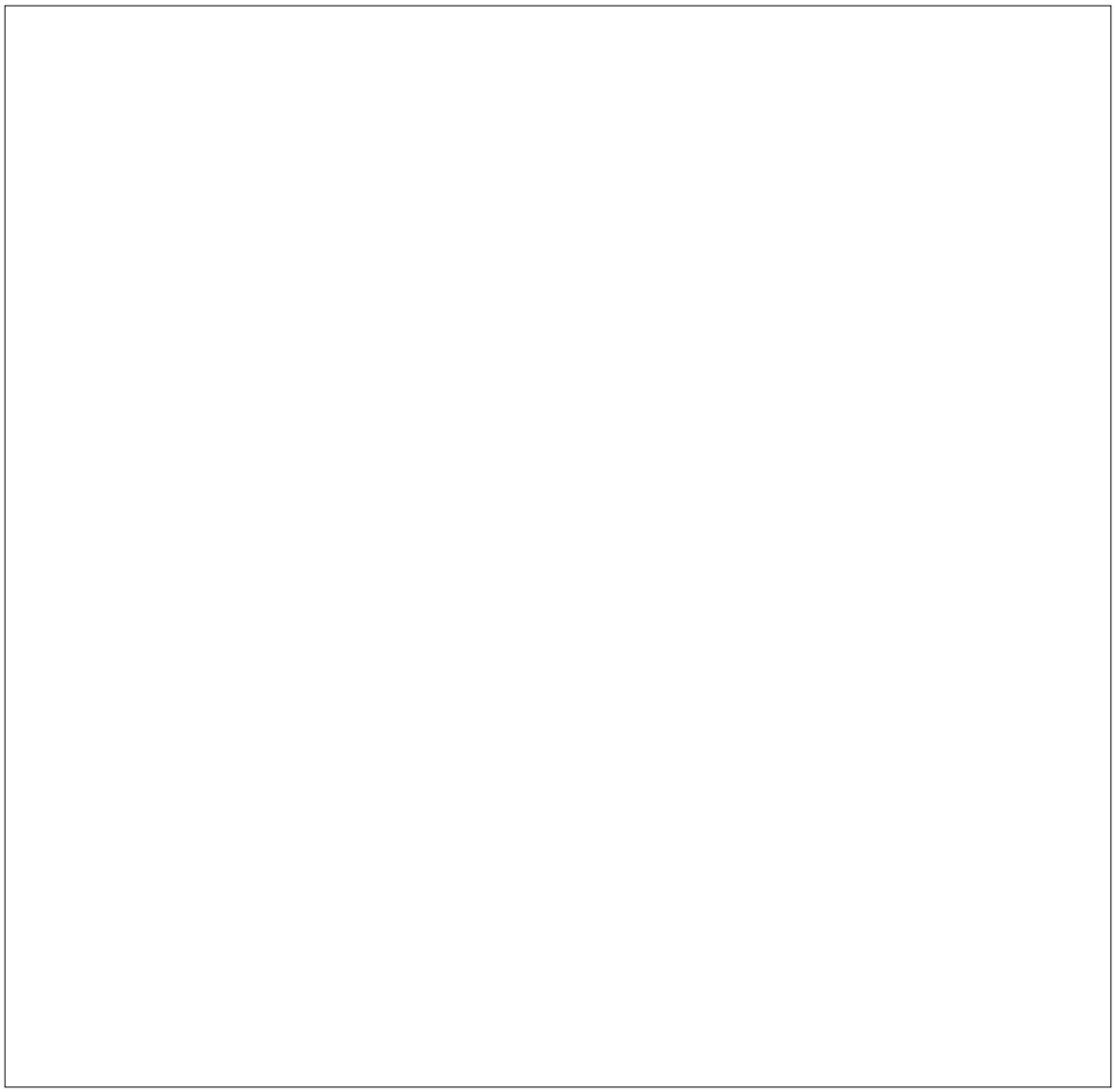
Geben Sie ein Sequenzdiagramm für ein `ControlPanel` und eine `Kundenkartei` an, bei der folgendes Szenario durchlaufen wird: Nachdem der Mitarbeiter nach Eingabe des Suchausdrucks `"Peter"` auf *Suchen* gedrückt hat, wird genau ein Kunde gefunden. An diesen Kunden wird die Nachricht `"Rechnung"` geschickt, nachdem der Mitarbeiter auf *Senden* geklickt hat. Anschließend legt der Mitarbeiter eine neue Kundin mit Namen `"Maria"` an und fügt sie der Kundenkartei hinzu.



Aufgabe 7: Objektdiagramme **7 Punkte**

Zeichnen Sie ein UML-Objektdiagramm für die Objekte, die unmittelbar nach Abarbeitung aller Befehle der `main`-Methode der Klasse `Main` im Anhang existieren.

Hinweis: Der Code ist so intuitiv wie möglich geschrieben und enthält keine absichtlichen Fallstricke.



Aufgabe 8: Lineare Temporallogik **13 Punkte (1/1/6/2/3)**

Es seien $\text{AP} = \{a, b, c\}$ und $\Sigma = 2^{\text{AP}}$.

a) Geben Sie eine zu $\mathcal{F}\, a$ semantisch äquivalente LTL-Formel an, die keinen $\mathcal{F}$-Operator enthält.

b) Geben Sie eine zu $\neg\, \mathcal{X}\, \neg a$ semantisch äquivalente LTL-Formel an, die keine $\neg$ enthält.

c) Entscheiden Sie, ob folgende Aussagen wahr oder falsch sind. Es gibt $+1$ Punkt für korrekte Kreuze und -1 Punkt für falsche (jedoch insgesamt mind. 0 Punkte).

	Wahr	Falsch
$(\mathcal{G}\, a) \vee \neg(\mathcal{G}\, a)$ ist semantisch äquivalent zu *false*.	☐	☐
$(\mathcal{G}\, b)\, \mathcal{R}\, \mathcal{F}(a\, \mathcal{U}\, c)$ ist eine syntaktisch korrekte LTL-Formel.	☐	☐
Es gibt mehr als einen unendlichen Lauf über Σ, der $\mathcal{F}(a \wedge b \wedge c)$ erfüllt.	☐	☐
Der Lauf $\{b\}\{\}\{a\}^\omega$ erfüllt $\mathcal{F}(b \rightarrow \mathcal{X}(a\, \mathcal{U}\, b))$.	☐	☐
Jeder Lauf, der $a\, \mathcal{U}\, b$ nicht erfüllt, erfüllt auch $(\mathcal{G}\, a) \vee (\mathcal{G}\, b)$ nicht.	☐	☐
Jeder Lauf, der $a\, \mathcal{U}(b \rightarrow \mathcal{X}\, c)$ erfüllt, erfüllt auch $\mathcal{F}\, b$.	☐	☐

d) Zeichnen Sie für die Formel $(\neg a) \wedge \mathcal{F}\, \mathcal{G}\, a$ ein Transitionssystem mit unendlichen Läufen, von denen **alle** die Formel erfüllen.

e) Zeichnen Sie für die Formel $(\mathcal{G}\, \mathcal{F}\, c) \wedge (\mathcal{G}\, \mathcal{F}\, \neg c) \wedge a\, \mathcal{U}\, \mathcal{X}\, \mathcal{X}\, c$ ein Transitionssystem mit unendlichen Läufen, von denen **alle** die Formel erfüllen.

Aufgabe 9: Algebraische Spezifikation **13 Punkte (2/3/5/3)**

Für die sichere Kommunikation mit ihren Kunden entwirft eine Firma einen abstrakten Datentyp, welcher auf Zeichen und Wörtern basiert und der jeweils eine Ver- und Entschlüsselungs-Operation auf ihnen definiert.
Es gibt Zeichen (der Einfachheit halber nur `A`, `B`, `C`) und Wörter. Letztere können entweder das leere Wort sein (`nil`) oder aber die Voranstellung eines Zeichens an ein Wort (`cons`). Die Operation `encZeichen` verschlüsselt, die Operation `decZeichen` entschlüsselt ein Zeichen. Die Operation `encWort` verschlüsselt, die Operation `decWort` entschlüsselt ein Wort.

```
1  spec Woerter =
2    sorts
3      zeichen = A | B | C
4      wort    = nil | cons (zeichen, wort)
5    ops
6      encZeichen: (zeichen) zeichen
7      decZeichen: (zeichen) zeichen
8      encWort:    (wort) wort
9      decWort:    (wort) wort
10   vars
11     z: zeichen
12     w: wort
13 end
```

a) Geben Sie ein Axiom an, welches fordert, dass die Entschlüsselung eines verschlüsselten Zeichens das ursprüngliche Zeichen ergibt.

b) Das Ver-/Entschlüsseln eines Wortes soll über das Ver-/Entschlüsseln eines Zeichens definiert sein, indem das ganze Wort *Zeichen für Zeichen* ver-/entschlüsselt wird. Geben Sie Axiome an, welche die zeichenweise **Ver**schlüsselung eines **Wortes** spezifizieren.

c) Im Folgenden ist für die Spezifikation `Woerter` ein Ausschnitt eines Modells $\mathcal{A}$ angegeben, welches griechische Buchstaben bzw. Listen von griechischen Buchstaben als Trägermengen verwendet.

$$\begin{aligned} \texttt{zeichen}^{\mathcal{A}} &:= \{\alpha, \beta, \gamma\} \\ \texttt{wort}^{\mathcal{A}} &:= \{\alpha, \beta, \gamma\}^* \\ \texttt{A}^{\mathcal{A}} &:= \alpha \\ \texttt{B}^{\mathcal{A}} &:= \beta \\ \texttt{C}^{\mathcal{A}} &:= \gamma \end{aligned}$$

Geben Sie für das Modell $\mathcal{A}$ Interpretationen für `nil`, `cons`, `encZeichen` und `decZeichen` an und stellen Sie dabei sicher, dass die Eigenschaft aus a) erfüllt ist.

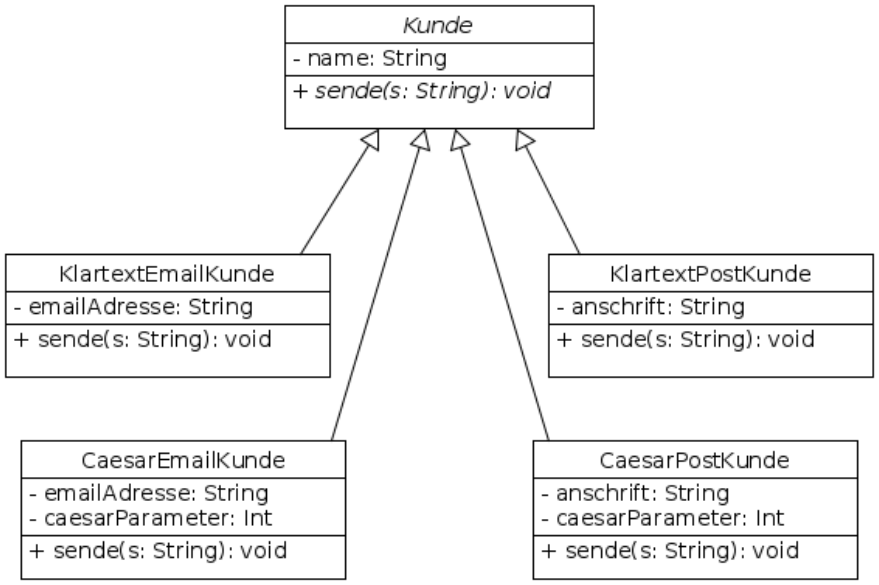
Hinweis: Listen können als Vektoren notiert werden, z.B. $\langle\rangle \in \{\alpha, \beta, \gamma\}^*$ *und* $\langle\alpha, \alpha\rangle \in \{\alpha, \beta, \gamma\}^*$.

d) Es soll eine neue Operation `ohneA: (wort) wort` geben, welche aus einem Wort alle Vorkommen des Zeichens `A` entfernt. Geben Sie Axiome an, um diese zusätzliche Operation korrekt zu spezifizieren.

Aufgabe 10: Design Patterns **10 Punkte (4/6)**

Eine Firma verwaltet ihre Kundenkartei elektronisch, wobei jeder Kunde als Objekt repräsentiert wird. Neben den Kundendaten (`name`) enthält ein Kundenobjekt eine Methode `sende`, um dem Kunden eine Textnachricht (z.B. Rechnung) zukommen zu lassen. Was genau bei einem Aufruf von `sende` geschieht, hängt vom Kunden ab: Manche Kunde bevorzugen die Zustellung per E-Mail, manche per Post. Manche Kunden wollen, dass die Nachrichten verschlüsselt zugestellt werden (z.B. mit der *Caesar-Verschlüsselung*, die mit einer Zahl parametrisiert ist), andere nicht. Diese Präferenzen können sich bei einem Kunden auch mal ändern. Zudem ist für die Zukunft nicht auszuschließen, dass neue Zustellmethoden und Verschlüsselungsmethoden hinzukommen.

Ein erster Entwurf sieht wie folgt aus.



a) Nennen Sie zwei (konzeptuell verschiedene) Schwachpunkte des Entwurfs hinsichtlich der gegebenen Anforderungen.

b) Zur Verbesserung des Entwurfs eignet sich das Strategy Pattern. Wenden Sie das Pattern zwei Mal an (auf zwei zu entkoppelnde Verhaltenskategorien), indem Sie das gegebene UML-Klassendiagramm anpassen und als vollständiges UML-Klassendiagramm neu zeichnen.

Korrektur

Diese Seite wird von den Korrektoren ausgefüllt.

Aufgabe	Erreichte Punkte	Mögliche Punkte
1 Softwarequalität		9
2 Vorgehensmodelle		8
3 Management		10
4 Planungsphase		9
5 Netzpläne		12
6 Sequenzdiagramme		9
7 Objektdiagramme		7
8 Lineare Temporallogik		13
9 Algebraische Spezifikation		13
10 Design Patterns		10
Gesamtpunktzahl		100

Note: _____

Anhang (zu Aufg. 4, darf aus Klausurheft herausgetrennt werden)

```
public class Main {
   public static void main(String[] args) {
      Zapfhahn fantaZapfhahn = new FantaZapfhahn();
      Zapfhahn pilsnerZapfhahn = new PilsnerZapfhahn();

      Kneipe kneipe = new Kneipe(fantaZapfhahn, pilsnerZapfhahn);

      Gast peter = new Gast();
      Gast maria = new Gast();

      Getraenk limo1 = kneipe.bestelle("grosse Limo");
      Getraenk limo2 = kneipe.bestelle("kleine Limo");
      Getraenk bier  = kneipe.bestelle("kleines Bier");

      peter.aktuellesGetraenk = limo1;
      peter.trinkeSchluck();
      peter.trinkeSchluck();
      peter.aktuellesGetraenk = limo2;

      maria.aktuellesGetraenk = bier;
      maria.trinkeSchluck();
   }
}



public abstract class Getraenk {
   protected int fuellstand;

   public Getraenk(int f) {
      fuellstand = f;
   }

   public void entnehme(int menge) {
      fuellstand = fuellstand - menge;
   }
}


public class Fanta extends Getraenk {
   public Fanta(int f) {
      super(f);
   }
}

public class Pilsner extends Getraenk {
   public Pilsner(int f) {
      super(f);
   }
}
```

```
public interface Zapfhahn {
   public Getraenk zapfeKleines();
   public Getraenk zapfeGrosses();
}

public class FantaZapfhahn implements Zapfhahn {
   public Getraenk zapfeKleines() {
      return new Fanta(330);
   }

   public Getraenk zapfeGrosses() {
      return new Fanta(500);
   }
}

public class PilsnerZapfhahn implements Zapfhahn {
   public Getraenk zapfeKleines() {
      return new Pilsner(250);
   }

   public Getraenk zapfeGrosses() {
      return new Pilsner(500);
   }
}

public class Kneipe {
   private Zapfhahn limoZapfhahn;
   private Zapfhahn bierZapfhahn;

   public Kneipe(Zapfhahn l, Zapfhahn b) {
      limoZapfhahn = l;
      bierZapfhahn = b;
   }

   public Getraenk bestelle(String wunsch) {
      if (wunsch.equals("kleine Limo"))
         return limoZapfhahn.zapfeKleines();
      else if (wunsch.equals("grosse Limo"))
         return limoZapfhahn.zapfeGrosses();
      else if (wunsch.equals("kleines Bier"))
         return bierZapfhahn.zapfeKleines();
      else if (wunsch.equals("grosses Bier"))
         return bierZapfhahn.zapfeGrosses();
      else return null;
   }
}

public class Gast {
   public Getraenk aktuellesGetraenk;

   public void trinkeSchluck() {
      aktuellesGetraenk.entnehme(20);
   }
}
```