HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES
INSTITUT INFORMATIK

15.09.2025

Klausur

Softwaretechnik

Sommersemester 2025

Studiengänge: AI, MTI, WI

Prüfer: Prof. Dr. Malte Weiß

Persönliche Angaben

Nachname	Vorname	Matrikelnummer

Bewertung

Frage	Punkte	Erreicht
1	9	
2	4	
3	4	
4	4	
5	6	
6	12	
7	10	

Frage	Punkte	Erreicht
8	10	
9	10	
10	10	
11	16	
12	5	
Gesamt:	100	

Ablauf der Prüfung

- **Zeit für die Lösung dieses Aufgabenblatts**: 120 Minuten.
- **Erlaubte Hilfsmittel**: Referenz am Ende der Aufgabenblätter (eine Seite). Handgeschriebener, ein- oder beidseitig beschriebener Notizzettel im DIN-A4-Format.
- **Nicht erlaubte Hilfsmittel**: Mobiltelefon, Kommunikation (E-Mail, Web, SMS, etc.). Die Verwendung eines nicht erlaubten Hilfsmittels wird als Täuschungsversuch gewertet.
- **Stift und Papier**: Schreiben Sie ausschließlich auf dem Papier der Aufgabenblätter. Sollten die Zwischenräume nicht ausreichen, können Sie zusätzliches Papier von der Aufsicht erhalten. Beschriften Sie dieses sofort mit ihrem Namen und Ihrer Matrikelnummer. Eigenes Papier ist nicht zulässig. Schreiben Sie ausschließlich mit dokumentenechten schwarzen oder blauen Stiften.
- **Vollständigkeit des Aufgabenblatts**: Die Klausur besteht aus den Aufgabenseiten 5 – 20 und einem Anhang auf Seite R1. Die Aufgabenblätter sind doppelseitig bedruckt. Prüfen Sie, ob alle Blätter vorhanden sind.
- **Fragen**: Melden Sie sich, wenn Sie eine Frage haben. Die Aufsicht kommt zu Ihnen, verlassen Sie nicht unaufgefordert Ihren Platz.
- **Zwischenschritte**: Geben Sie Zwischenschritte an. So können Sie selbst bei falschem Endergebnis einen Teil der Punkte erreichen.

Grundlagen

1. (a) Wie heißt das Software Design Pattern (Softwareentwurfsmuster), das sicherstellt, dass von einer Klasse nur eine einzige Instanz erstellt werden kann? (1½)

(b) Wie nennt man die Testart, bei der alle Anforderungen einmal vollständig durchgetestet werden, um die Erfüllung des Vertrags zu prüfen? (1½)

(c) Wie nennt man das Dokument, in dem alle ausgearbeiteten Anforderungen hinterlegt werden? (1½)

(d) Wie nennt man Software, die nach genauen Vorgaben für einen Kunden entwickelt wird? (1½)

(e) Wie nennt man die Architektur, bei der alle Komponenten aus vielen unabhängigen, skalierbaren Diensten bestehen, die miteinander kommunizieren? (1½)

(f) Wie nennt man die Methode zur Erhebung von Anforderungen, bei der man einen Nutzer bei seiner alltäglichen Arbeit beobachtet? (1½)

Anforderungen

2. Beschreiben Sie die zwei Transformationen, die auftreten können, wenn Anforderungen **sprachlich** erfasst werden. (4)

3. Erläutern Sie anhand eines **konkreten Beispiels**, wieso das Vergessen einer Nicht-Funktionalen Anforderung (NFA) oft problematischer ist als das Vergessen einer Funktionalen Anforderung (FA). (4)

4. Warum ist es aus wirtschaftlicher Sicht wichtig, dass die Erfüllung einer Anforderung **messbar** ist. (4)

5. Geben Sie für die folgenden Aussagen an, welche Rolle eines Scrum-Teams verantwortlich ist. Es kann nur eine Rolle genannt werden. Gibt es keine passende Rolle, schreiben Sie "niemand".

   (a) "Ich implementiere eine Java-Klasse." (1)

   (b) "Ich sortiere die Anforderungen nach Wichtigkeit." (1)

   (c) "Ich plane das Projekt von der Erhebung der Anforderungen bis zum Rollout." (1)

   (d) "Ich erstelle manuelle Testfälle." (1)

   (e) "Ich beseitige Hindernisse, die das Team an produktiver Arbeit hindern." (1)

   (f) "Ich erkläre dem Team, worauf es bei der Retrospektive ankommt." (1)

UML

6. Erstellen Sie ein Aktivitätsdiagramm für das folgende Szenario eines Restaurants: (12)

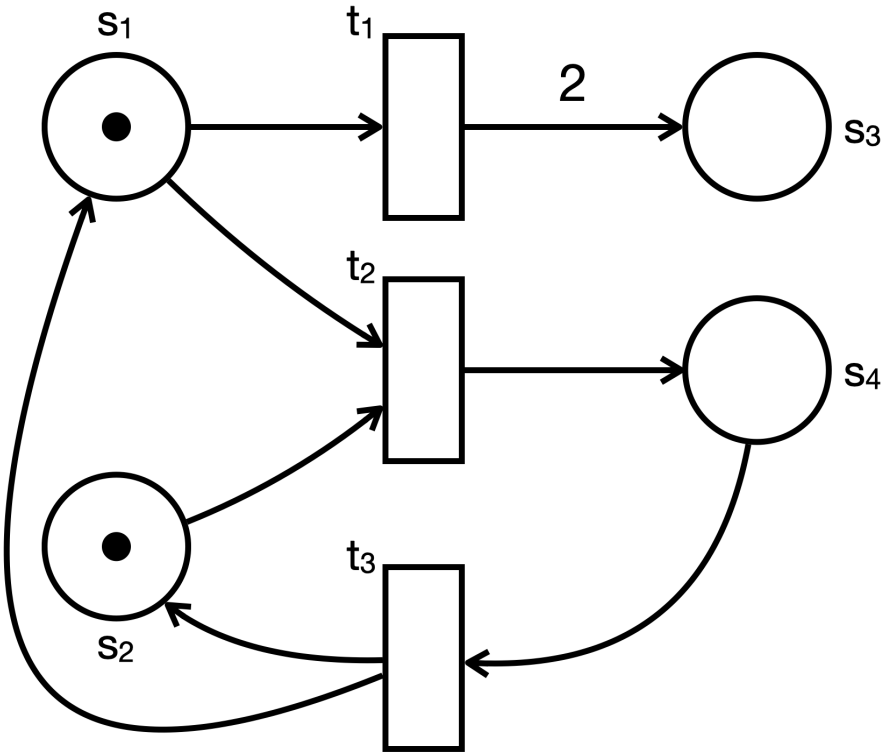
   > Besucher des Restaurants geben zunächst ihre Bestellung für Essen und Getränke auf. Danach wird das Essen zubereitet und anschließend zum Tisch gebracht. Während die Gäste darauf warten, werden bereits die Getränke geliefert. Nachdem das Essen verzehrt wurde, fordern die Gäste die Rechnung an, die ein:e Kellner:in dann zum Tisch bringt. Die Gäste bezahlen nun – wenn der Service gut war, schlagen sie etwas Trinkgeld auf den Rechnungsbetrag auf.

   Identifizieren Sie alle Aktionen, Parallelisierungen und Entscheidungen.

   Erstellen Sie dann auf der nächsten Seite ein Aktivitätsdiagramm.

Petrinetz

7. Betrachten Sie das folgende Petrinetz:



Lösen Sie die Aufgabe auf der nächsten Seite.

(a) Erstellen Sie für das gezeigte Petri-Netz den Erreichbarkeitgraphen. (4)

(b) Beantworten Sie anhand des Petri-Netzes **oder** anhand des Erreichbarkeitsgraphen, ob folgende Aussagen zutreffen: (6)

1. “Es gibt einen Deadlock.”
2. “Das Netz ist stark lebendig.”
3. “Das Netz ist schwach lebendig.”

Jede Aussage muss **separat** beantwortet werden.

Aufwandsschätzung

8. Berechnen Sie den Aufwand nach dem Application-Composition-Modell für das im Folgenden beschriebene Softwaresystem. Wie lange benötigt das Team für die Entwicklung des Softwaresystems? (10)

   Das Softwaresystem besteht grob aus folgenden Komponenten:

   - 4 Screens, davon 2 komplexe und 2 sehr komplexe
   - Insgesamt 32 Java-Klassen
   - Einen schwierig zu erzeugenden Bericht

   Das Team ist sehr erfahren und hat schon Jahre lang zusammengearbeitet. Außerdem kann die Hälfte des Codes aus dem vorherigen Projekt übernommen werden.

   Hinweis: Das Berechnungsmodell finden Sie im Anhang.

Objektorientierung

9. Betrachten Sie folgende Java-Klasse `Player`, die einen Spieler mit Namen, Geschlecht und Lieblingsspiel modelliert. (10)

```
public enum Gender {
    MALE, FEMALE
}

public class Player {
    public String name;
    public Gender gender;
    public String favoriteGame;

    public void print() {
        System.out.print(name + " (");
        if(gender == Gender.MALE) {
            System.out.print("male");
        } else {
            System.out.print("female");
        }
        System.out.print("), favorite game: " + favoriteGame);
    }
}
```

Der Entwickler hat hier gegen fundamentale Prinzipien der Objektorientierten Programmierung verstoßen. Erläutern Sie die Verstöße und wie man sie auflöst.

Lösen Sie die Aufgabe auf der nächsten Seite.

Software Design Patterns

10. Betrachten Sie Klasse `PartyPlanner`. Sie erlaubt es, die Personen zu erfassen, die zu einer Party eingeladen werden. (10)

```
public class PartyPlanner {
    private List<Person> persons = new ArrayList<>();

    public void addPerson(Person person) {
        persons.add(person);
    }

    public int getNumberOfPersons() {
        return persons.size();
    }
}
```

Die Klasse `Person` speichert Namen und Alter, auf die jeweils mit `getName()` und `getAge()` zugegriffen kann.

Implementieren Sie einen Security-Proxy `AdultPartyPlannerProxy` für die obigen Klasse. Der Proxy stellt sicher, dass **nur erwachsene Personen** zu einer Party eingeladen werden können.

Lösen Sie die Aufgabe auf der nächsten Seite.

11. Schreiben Sie JUnit-Tests für die folgende Java-Methoden.

(a) Die Java-Methode `getHighestNumber` liefert die größte von drei übergebenen Zahlen zurück. (6)

```
public static long getHighestNumber(int a, int b, int c) {
    if(a > b && a > c) {
        return a;
    }
    else if(b > a && b > c) {
        return b;
    }
    return c;
}
```

(b) Die Java-Methode `containsName` erhält ein Array von Strings und einen Suchstring. Sie gibt genau dann `true` zurück, wenn der Suchstring in einem der Namen vorkommt. Bei einem ungültigen Parameter wird eine Exception geworfen. (10)

```
public static boolean searchName
        (String[] names, String searchString) {
    if(names == null || searchString == null) {
        throw new IllegalArgumentException();
    }
    if(searchString.length() == 0) {
        return true;
    }
    for(String name : names) {
        if(name.contains(searchString)) {
            return true;
        }
    }
    return false;
}
```

12. Erklären Sie die Aufgaben eines Build-Systems wie Maven. (5)

Anhang

Application-Composition-Modell

- Ermittlung der Application Points (NAP)
  - Anzahl einzelner Screens oder Websites
    (einfach = 1 AP, komplexer = 2 AP, sehr komplex = 3 AP)
  - Anzahl der erstellen Berichte
    (einfach = 2 AP, komplexer = 5 AP, schwer erzeugbar = 8 AP)
  - Anzahl der Module in Programmiersprache 3. Generation (z.B. Java-Klassen)
    1 AP pro Modul
- Produktivität (→ PROD = NAP / Monat)

Erfahrung und Kompetenz des Entwicklers / Reife der Softwarewerkzeuge	sehr gering	gering	nominell	hoch	sehr hoch
PROD (NAP/Monat)	4	7	13	25	50

$$PM = (NAP \times (1 - \%reuse)) / PROD$$

PM	=	Aufwand in Personenmonaten
NAP	=	Gesamtzahl der Application-Points im fertigen System
%reuse	=	Schätzung der wiederverwendeten Codemenge (0-1)
PROD	=	Produktivität bei der Verarbeitung von Application-Pointe