



**Klausur
Softwaretechnologie WS 2015/16**

Prof. Dr.rer.nat.habil.
Uwe Aßmann

Name:	
Vorname:	
Immatrikulationsnummer:	

Aufgabe	Maximale Punktzahl	Erreichte Punktzahl
1	37	
2.1	2	
2.2	6	
2.3	45	
Gesamt	90	

Hinweise:

- In der Klausur ist als Hilfsmittel lediglich ein **A4-Blatt, beidseitig beschrieben**, zugelassen.
- Die Klammerung der Aufgabenblätter darf **nicht** entfernt werden.
- Tragen Sie bitte die Lösungen auf den Aufgabenblättern ein!
- Verwenden Sie keine roten, grünen Stifte oder Bleistifte!
- Es ist kein eigenes Papier zu verwenden! Bei Bedarf ist zusätzliches Papier bei der Aufsicht erhältlich. Bitte jedes zusätzliche Blatt mit Name, Vorname und Immatriculationsnummer beschriften.
- Es sind alle Aufgabenblätter abzugeben!
- Ergänzen Sie das Deckblatt mit Name, Vorname und Immatriculationsnummer!
- Halten Sie Ihren Studentenausweis und einen Lichtbildausweis zur Identitätsprüfung bereit.
- **Achtung!** Das Zeichen ☞ heißt: **Hier ist Java-Text einzufügen!**

Aufgabe 1: Domänenmodell für Projektanträge (37 Punkte)

Das Erstellen von Projektanträgen, die durch Fördermittelgeber unterstützt werden sollen, ist eine aufwendige Angelegenheit. Zunächst müssen wir verstehen, welche grundlegenden Begriffe eine Rolle spielen. Dazu soll im Folgenden ein Domänenmodell erstellt werden.

Projektanträge werden in einem (Förder-)Programm gestellt. Ein solches Programm hat grundsätzlich einen Namen, eine Beschreibung und einen Fördersatz. Für jedes Programm gelten Regeln (minimal eine Regel), wobei Regeln typischerweise gleichzeitig für mehrere Programme gelten.

Wenn ein Projektantrag gestellt wird, müssen für diesen ein Name, eine Zielstellung und die Gesamtkosten angegeben werden. Zusätzlich werden alle Projektpartner mit ihren Namen, Adresse, Umsatzsteuernummer, einem Ansprechpartner und einem Vertretungsbefugten aufgelistet. Außerdem muss für jeden Projektpartner im Projektantrag sein Arbeitspaket mit Arbeitsinhalt, Zeitraum und dazu notwendigen Personenmonaten geplant werden.

Projektpartner können Unternehmen (mit angegebener Rechtsform), Startups oder Forschungseinrichtungen sein. Jeder Projektantrag muss genau einen Projektpartner als Antragsteller benennen. Falls es mehrere Projektpartner in einem Projektantrag gibt, handelt es sich um ein Kooperationsprojekt, für welches eine Kooperationsvereinbarung mit zugehörigen Verwertungsrechten abgeschlossen werden muss. Jeder Projektpartner kann an mehreren Projektanträgen beteiligt sein. Das Gleiche gilt auch für seine Rolle als Antragsteller.

Im Projektantrag müssen, aufgeschlüsselt für jeden Projektpartner, die erwarteten Wirkungen durch das Projekt aufgelistet werden. Das beinhaltet die Auswirkungen auf die Wettbewerbsfähigkeit, die technologische Basis und gegebenenfalls die erwarteten Umsätze.

Der Antragsteller muss sich zudem entscheiden, ob es sich bei dem Projektantrag um ein Forschungsprojekt oder ein Entwicklungsprojekt handelt. Im Fall eines Entwicklungsprojektes muss als Bestandteil des Projektantrages ein Markteinführungskonzept mit Businessplan eingereicht werden. Um den Aufwand der Erstellung eines vollständigen Projektantrages zu minimieren, kann zunächst eine Projektskizze eingereicht werden. Gegebenenfalls wird später kein Projektantrag erstellt.

Jeder Projektantrag wird von einem Fördermittelgeber begutachtet.

Modellieren Sie alle oben genannten domänenspezifischen Begriffe in einem UML-Analyseklassendiagramm!

Berücksichtigen Sie dabei

- **Klassen mit Attributen (sofern oben genannt) und OHNE Operationen**
- **Assoziationen/Aggregationen/Kompositionen einschließlich Multiplizitäten, Rollen- oder Assoziationsnamen (mit Leserichtung)**
- **Assoziationsklassen**
- **Vererbungsbeziehungen**

Aufgabe 2: Observer-Muster mit Änderungsmanager (53 Punkte)

Im Klassendiagramm auf Seite 6 werden Kalendereinträge als Anwendungsbeispiel für das **Observer-Muster** betrachtet. Wenn ein Datum angelegt oder geändert wird (`setDatum()`), werden alle Beobachter benachrichtigt. Ein Objekt der Klasse `Termine` kapselt alle Termine **eines** Tages. Jeder einzelne Termin dieses Tages wird implementiert durch eine textuelle Beschreibung (`termin`) und die Uhrzeit (`Zeit`) des Tages. Alle einzelnen Termine eines Tages sollen in einer `TreeMap` – zeitlich aufsteigend geordnet – gespeichert werden. Die Zeit wird in Stunde (`stunde`) und Minute (`min`) angegeben. Ein Objekt der Klasse `Aufgaben` kapselt alle zu erledigenden Aufgaben **eines** Tages. Jede einzelne Aufgabe wird durch eine textuelle Beschreibung bereitgestellt.

Die Aktualisierungssemantik wird gegenüber dem klassischen Observer-Muster in einer eigenen Klasse (`AenderungsManager`) gekapselt. Diese Klasse bildet das Subjekt auf seine Beobachter ab und bietet eine Schnittstelle zur Verwaltung dieser Abbildung. Bei Anforderung durch das Subjekt (`benachrichtige()`) werden durch den Änderungsmanager alle zugeordneten Beobachter aktualisiert. Die Aktualisierung der Beobachter erfolgt im Programm durch eine Ausgabe auf der Kommandozeile.

Teilaufgabe 2.1 (2 Punkte):

Um welche Variante des Observer-Musters handelt es sich und warum?

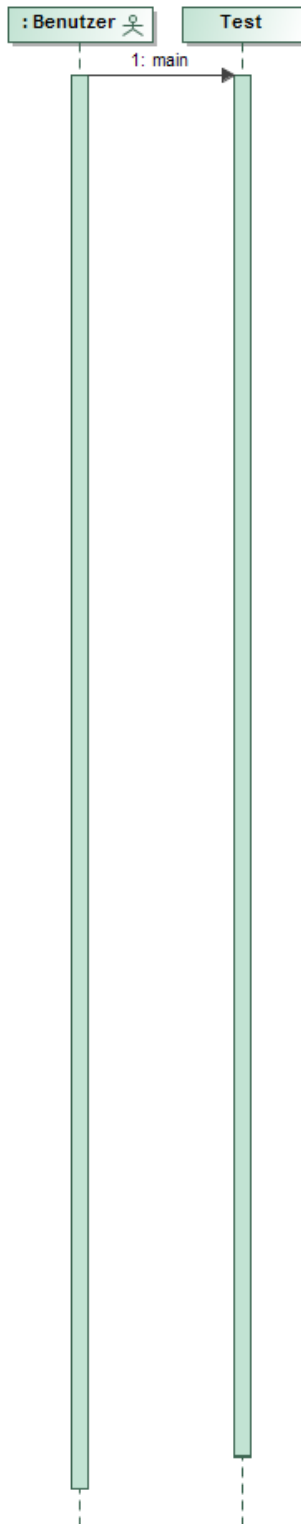
Teilaufgabe 2.2 (6 Punkte):

Betrachten Sie zur Illustration des Entwurfes die folgende `main()`-Methode der Klasse `Test`. Dieser Java-Code erzeugt eine Ausgabe auf der Kommandozeile, die auf Seite 10 unten angegeben ist.

```
public class Test {
    public static void main(String[] args) {
        AenderungsManager manager = new AenderungsManager();
        Datumsauswahl d = new Datumsauswahl (manager);
        Termine t = new Termine(d);
        Aufgaben a = new Aufgaben(d);
        d.meldeAn(t);
        d.meldeAn(a);
        d.setDatum(new Date(116,1,15)); // 15. Februar 2016
        t.addTermin(new Zeit(11,10), "ST-Klausur");
        t.addTermin(new Zeit(9,30), "ST-Meeting ");
        a.addAufgabe("Klausuraufsicht");
        a.addAufgabe("Klausurkorrektur planen");
        d.setDatum(new Date(116,1,18)); // 18. Februar 2016
    }
}
```

Vervollständigen Sie für die `main()`-Methode das UML-Sequenzdiagramm!

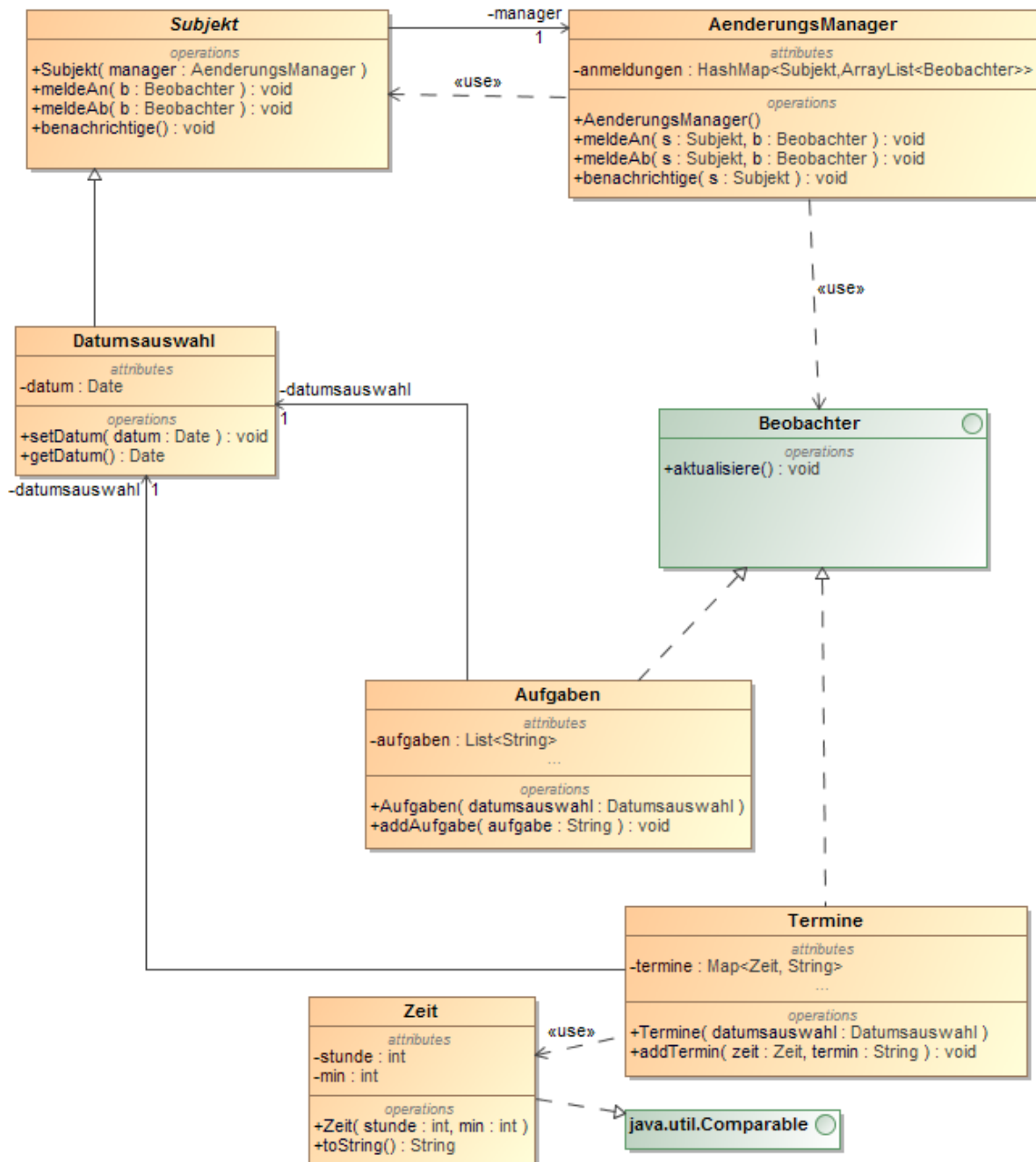
- Zeichnen Sie die Objekte und Methodenaufrufe ohne Parameter und Rückgabenachrichten ein!
- Beschränken Sie die Nachrichten im Sequenzdiagramm auf die Methodenaufrufe der `main()`-Methode (ohne dabei die Observer-Logik darzustellen).




Teilaufgabe 2.3 (45 Punkte):

Implementieren Sie den Beobachter mit Änderungsmanager, indem Sie den Java-Code auf den folgenden Seiten vervollständigen!

- Achten Sie auf Konsistenz des Java-Codes zu den Variablen und Methoden im Entwurfsklassendiagramm!
- Verzichten Sie der Einfachheit halber auf den Test von Parametern auf Nullwerte.
- Auf der Kommandozeile soll das Programm den Text auf Seite 10 erzeugen! Für die Ausgabe eines `TreeMap`-Objektes muss lediglich die Methode `toString()` der Klasse `TreeMap` aufgerufen werden.



```
import java.util.*;    // gilt für alle Klassen
import java.util.Date;
```

```
public abstract class Subjekt {
    
```

```
}
```

```
 public class Datumsauswahl {
```

```
}
```

```
public interface Beobachter {
    public void aktualisiere();
}
```

```
public class AenderungsManager {
```

```
    ↵
```

```
}
```



```
⌘ public class Termine {
```

```
}
```

```
⌘ public class Zeit {
```

```
}
```

```
public class Aufgaben {
```

```
}
```

Ausgabe von `main()` auf der Kommandozeile:

[Zeile 1]

Mon Feb 15 00:00:00 CET 2016 mit folgenden Terminen: {}

[Zeile 2]

Mon Feb 15 00:00:00 CET 2016 mit folgenden Aufgaben: []

[Zeile 3]

Thu Feb 18 00:00:00 CET 2016 mit folgenden Terminen: {9:30=ST-Meeting , 11:10=ST-Klausur}

[Zeile 4]

Thu Feb 18 00:00:00 CET 2016 mit folgenden Aufgaben: [Klausuraufsicht, Klausurkorrektur planen]
