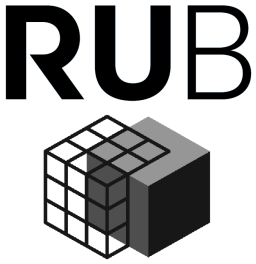

RUHR-UNIVERSITÄT BOCHUM

Fakultät für Bauingenieurwesen
Lehrstuhl für Informatik im Bauwesen
Prof. Dr.-Ing. M. König

Onlineklausur

Objektorientierte Modellierung

10.03.2022
Wintersemester 2021/22

Bearbeitungszeitraum: 10:00 – 14:00

Name:

Vorname:

Matrikelnummer:

Aufgabe	1	2	3	4	Gesamt
Punkte	25	30	25	20	100
Erreicht					

Regeln und Hinweise für die Onlineklausur

- Alle Aufgaben müssen eigenständig erarbeitet werden. Gruppenarbeit oder Anfragen von Hilfeleistungen (z.B. durch Forenposts) sind **nicht** erlaubt und werden als Täuschungsversuch geahndet.
- Erlaubte Hilfsmittel sind alle Kursunterlagen, Internetquellen, Grafikprogramme, IDEs und GUI-Builder (solange diese nur Pakete aus `java.*` oder `javax.*` verwenden).
- Benutzen Sie keine externen Bibliotheken. Benutzen Sie nur Klassen aus den Paketen `java.*` und `javax.*`, oder aus Paketen die Ihnen als Teil der Klausur zur Verfügung gestellt werden.
- Benutzen Sie keine Programme, die UML-Diagramme automatisiert aus Code erzeugen. Alle UML-Diagramme müssen selbst erstellt sein. Sie können dafür Programme wie *diagrams.net* benutzen, oder die Diagramme per Hand zeichnen und einscannen.
- Zur Abgabe, füllen Sie Name und Matrikelnummer in dieser PDF aus ODER erstellen Sie eine Textdatei in der Name und Matrikelnummer erhalten sind. Verpacken Sie ihr Projekt mit allen Quelldateien, UML-Diagrammen und der PDF / Textdatei in einer zip-Datei und laden diese in dem vorgesehenden Dateiupload in Moodle hoch.
- Fragen zu der Klausur können im gegebenen *Zoom-Raum* gestellt werden.
- Bei der Bewertung von Programmcode wird Wert auf Sauberkeit und Code-Qualität gesetzt. Verwenden Sie aussagekräftige Variablen- und Klassennamen und kommentieren Sie kritische Stellen in Ihrem Code.

Aufgabenstellung

Aufgabe 1

Punkte	Erreicht
25	

Für eine Social-Media-Plattform sollen Sie einen Link-Filter implementieren, welcher die Darstellung von gefährlichen Hyperlinks in Beiträgen blockiert. Sobald ein Beitrag einen Link zu einer schädlichen Webseite enthält, soll der Link durch den Text "[Link blockiert]" ersetzt werden. Der Rest des Beitrags soll jedoch weiterhin lesbar bleiben. Folgende Klassenstruktur wird Ihnen vorgegeben.

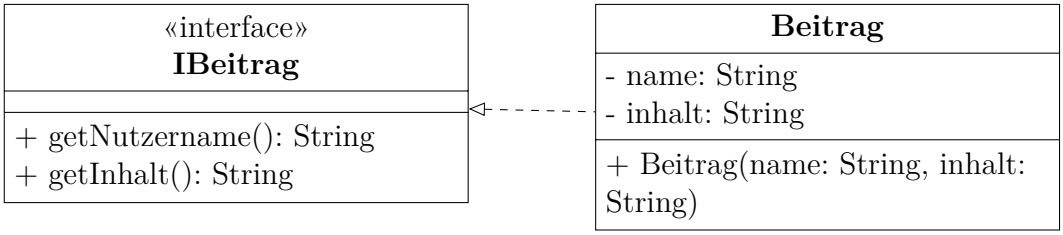

Abbildung 1: UML-Klassendiagramm der Plattform

Die Klasse `Beitrag` enthält immer den originalen Beitrag, welcher über das Interface `IBeitrag` z.B. der GUI zur Anzeige bereitgestellt wird. Verwenden Sie das **Proxy-Muster**, um eine zensierte Version eines Beitrags bereitstellen zu können. Blockiert werden sollen folgende Links:
`getmalware.test`, `freeram.example`, `notavirus.invalid`.

Achtung: Die Links können auch im Nutzernamen versteckt sein!

(a) Erweitern Sie das o.g. Klassendiagramm um ein Proxy-Muster. (5 Pkt)

(b) Implementieren Sie alle Klassen aus Aufgabenteil (a) und demonstrieren Sie die Funktionalität in der Main-Funktion. Erstellen Sie einen Beitrag in denen zwei Links auf der Blockliste vorkommen und lassen diesen durch den Proxy anzeigen. (10 Pkt)

(c) Ihre Firma hat eine andere Social-Media-Plattform gekauft, und will nun die Nutzerbasis vereinigen. Machen Sie die unten gezeigte Klasse `Posting` mit Ihrem System kompatibel, indem Sie einen Adapter entwerfen. Erweitern Sie ihr UML-Klassendiagramm aus Aufgabenteil (a) um entsprechende Klassen (eine Implementierung ist nicht gefordert). Sollten Sie Aufgabenteil (a) nicht gelöst haben, können Sie das Klassendiagramm aus Abbildung 1 stattdessen erweitern. (10 Pkt)

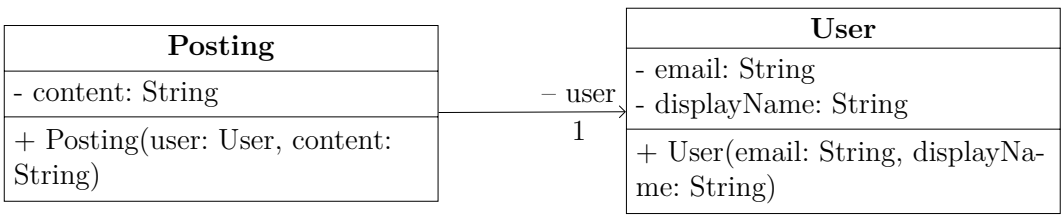

Abbildung 2: UML-Klassendiagramm der akquirierten Plattform

Aufgabe 2

Punkte	Erreicht
30	

Für ein Sicherheitssystem soll eine Software entwickelt werden, die eine Sicherheitstür kontrollieren kann. Die Tür kann offen (*open*), geschlossen (*closed*), und verschlossen (*locked*) sein. Abbildung 3 zeigt die vorgegebene GUI für das Sicherheitssystem. Oben wird der Status der Tür angezeigt, darunter zwei Knöpfe zum Bedienen der Tür. Basierend auf dem jetzigen Zustand sollen die beiden Knöpfe verschiedene Funktionen haben.

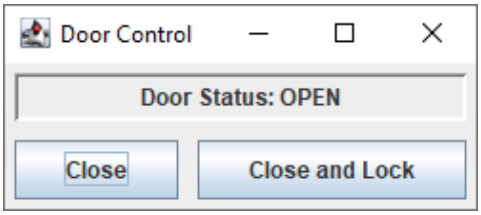

Abbildung 3: GUI der Türkontrolle

Abbildung 4 zeigt ein UML-Zustandsdiagramm für die Türkontrolle. Bei einem Zustandswechsel sollen sowohl der Text des Labels als auch der Text auf den Buttons entsprechend des Zustands gesetzt werden. Ist die Tür z. B. im Zustand *Closed*, so soll der linke Button *Lock* anzeigen, und der rechte Button *Open*. Wird ein Button gedrückt soll der Zustand entsprechend gewechselt werden.

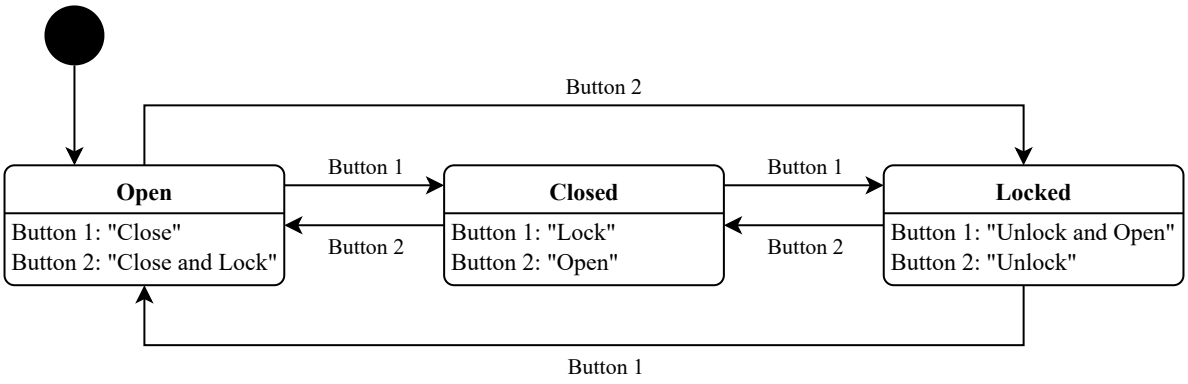

Abbildung 4: UML-State-Diagramm der Türkontrolle

(a) Implementieren Sie das State-Diagramm im Abbildung 4 als ein **State-Muster**. (10 Pkt)

(b) Verknüpfen Sie ihren Programmcode aus Aufgabenteil (a) mit der vorgegebenen GUI. Dazu können Sie die gegebene GUI-Klasse `DoorControlGUI.java` verwenden. Achten Sie dabei auf das **MVC-Prinzip**. (15 Pkt)

(c) Fügen Sie eine Undo-Funktionalität zu ihrem Programm hinzu, welche die letzte Eingabe rückgängig macht. Wählen Sie dafür ein passendes Muster. (5 Pkt)

Command-Muster

Aufgabe 3

Punkte	Erreicht
25	

Für eine Bauwerkverwaltungssoftware sollen die Bauteile eines Bauwerks besser organisiert werden. Einzelne Bauteile sollen unter Bauwerksgruppen zusammengefasst werden. Ihnen wird folgendes UML-Diagramm gestellt, welches Sie zu Java-Code umsetzen sollen.

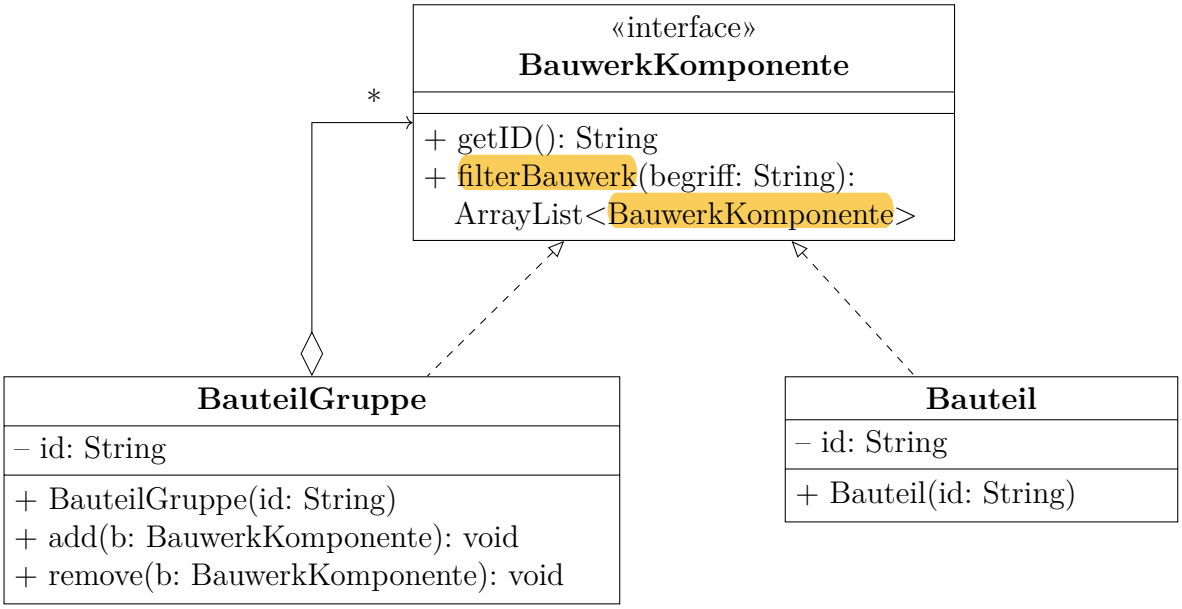

Abbildung 5: UML-Klassendiagramm der Bauverwaltungssoftware

(a) Implementieren Sie das in Abbildung 6 gezeigte **Composite-Muster**. (10 Pkt)

(b) Implementieren Sie die Funktion `filterBauteil(begriff: string)`, welche in der Bauwerkshierarchie nach Bauwerkskomponenten mit `begriff` in der ID sucht, und diese als Liste zurückgibt. (5 Pkt)

(c) Erstellen Sie ein UML-Sequenzdiagramm, dass einen beispielhaften Aufruf der Funktion `filterBauteil("BT1")` auf folgender Hierarchie zeigt: (10 Pkt)

- BT1Group
  - Pillar_BT1
  - Wall01

Aufgabe 4

Punkte	Erreicht
20	

Ein Labor möchte untersuchen, wie sich Mäuse in Labyrinthen orientieren. Sie werden beauftragt, ein Programm zum Erstellen und Verwalten dieser Labyrinthe zu entwerfen. Das Labor gibt Ihnen folgende Vorgaben:

- Labyrinthe basieren immer auf einem $n \times m$ großen Gitter aus Zellen.
- Zellen können durch Wände oder Vorhänge getrennt sein.
- Jedes Labyrinth hat genau einen Start- und einen Endpunkt.
- Zellen können markiert werden mit Farben, Mustern, oder Gerüchen.

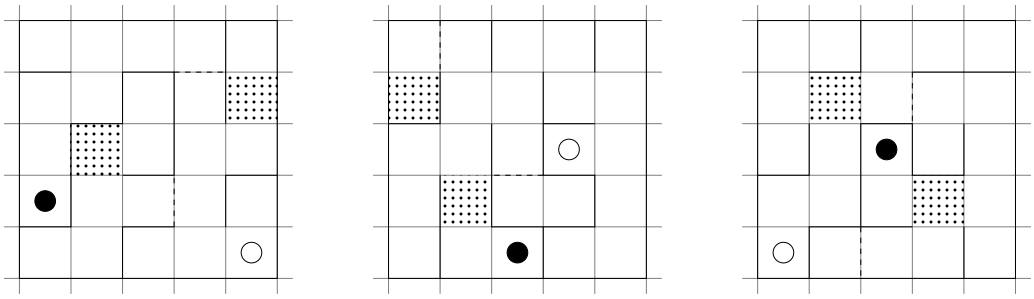

Abbildung 6: Beispiele für Labyrinthe: Start- und Endpunkte sind als Kreise markiert, Wände als durchgezogene Linien, Vorhänge als gestrichelte Linien. Zellen können z.B. mit Mustern (hier gepunktet) markiert werden.

Im folgenden soll das Programm nur als **UML-Klassendiagramm** entworfen werden.

(a) Entwerfen Sie entsprechende Klassen zur Repräsentation eines Labyrinths als UML-Klassendiagramm. Achten Sie dabei auf die Erweiterbarkeit des Systems und die Prinzipien der Objektorientierung. (10 Pkt)

(b) Das Labor möchte, nach bestimmten Vorgaben, zufällige Labyrinthe erstellen können. Der Nutzer soll entweder ein einfaches Standardlabyrinth mit Größe 5×5 erstellen können, oder ein schwieriges Labyrinth mit beliebiger Größe. Entwerfen Sie Klassen für die Datenhaltung und Erstellung von zufälligen Labyrinthen als UML-Klassendiagramm. Nutzen Sie dafür das Fabrikmuster. Ein konkreter Algorithmus zum Erstellen eines Labyrinths ist **nicht** gefordert. (10 Pkt)

Sollten Sie Aufgabenteil (a) nicht gelöst haben, nehmen Sie an, dass eine Klasse `Labyrinth` *mit leerem Konstruktor existiert.*