

RUHR-UNIVERSITÄT BOCHUM

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

Onlineklausur

## Objektorientierte Modellierung

**22.09.2021**

Sommersemester 2021

Bearbeitungszeitraum: 10:00 – 14:00

Name:

Vorname:

Matrikelnummer:

Aufgabe	1	2	3	4	Gesamt
Punkte	20	25	30	25	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](https://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 vorgesehenen 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

**DONE**  
**real class** 也可以建立**constructor**  
**proxy**的用处, 就是减轻**real class**的负担, 并且  
 添加新的功能, 但是它做的还是**real class**的工作

## Aufgabe 1

Punkte	Erreicht
20	

Eine Mailanwendung soll erweitert werden, um nun auch verschlüsselte Mails zu unterstützen. Dabei ist zu beachten, dass nur der Inhalt einer Mail verschlüsselt bzw. entschlüsselt wird. Alle anderen Eigenschaften, wie z.B. Sender, Empfänger oder Betreff liegen zu jedem Zeitpunkt unverschlüsselt vor. Folgende Logik liegt bereits vor:

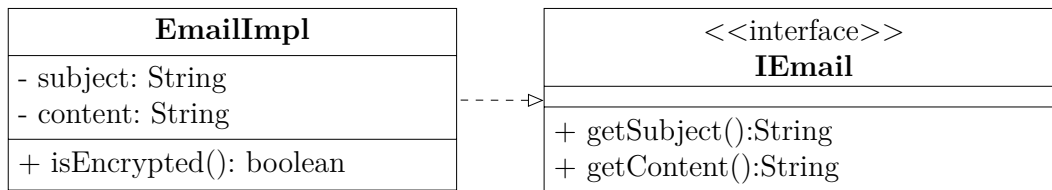


Abbildung 1: UML-Klassendiagramm der Mailanwendung

Die Klasse EmailImpl enthält immer die originale Nachricht, welche über das Interface IEmail z.B. der GUI zur Anzeige bereitgestellt wird. Verwenden Sie das Proxy-Muster, um eine entschlüsselte Version jeder Nachricht bereitstellen zu können. Über die Methode isEncrypted() der Klasse EmailImpl kann abgefragt werden, ob der Inhalt verschlüsselt ist. Über den Proxy muss in jedem Fall eine entschlüsselte Version bereitgestellt werden. Gehen Sie davon aus, dass Mails in diesem Fall mit Base64 verschlüsselt sind. Zum Entschlüsseln können Sie die Java-Klasse java.util.Base64 verwenden.

- Erweitern Sie das o.g. Klassendiagramm um ein Proxy-Muster. (5 Pkt)
- Implementieren Sie alle Klassen aus Aufgabenteil (a) und demonstrieren Sie die Funktionalität in der Main-Funktion. Erstellen Sie eine Mail mit dem verschlüsselten Inhalt "R2V0ZW1t" und entschlüsseln diese mit ihrem Proxy. (10 Pkt)
- Weiterhin sollen für die Verschlüsselung verschiedene Algorithmen bereitgestellt werden. Mails sollen entweder mit RSA oder mit einem Elliptic-Curve-Verfahren entschlüsselt werden können, wofür ein Passwort benötigt wird. Erweitern Sie ihr UML-Klassendiagramm entsprechend (eine Implementierung ist nicht gefordert). (5 Pkt)

DONE

注意 conc State里面, 过渡state的方法, 先更新信息, 再换state

## Aufgabe 2

Punkte	Erreicht
25	

Sie erhalten die Aufgabe, die Software für eine neue Kaffeemaschine zu entwickeln. Die Kaffeemaschine kann **normalen Kaffee und Milchkaffee ausgeben**. Außerdem lassen sich Brühzeit und Röststufe auf *HOCH*, *MITTEL*, und *NIEDRIG* einstellen. Das Gerät besitzt drei Knöpfe: *Start*, *Select*, und *Stop*. Im folgenden erhalten Sie ein UML-State-Diagramm, welches die Menüführung der Kaffeemaschine beschreibt.

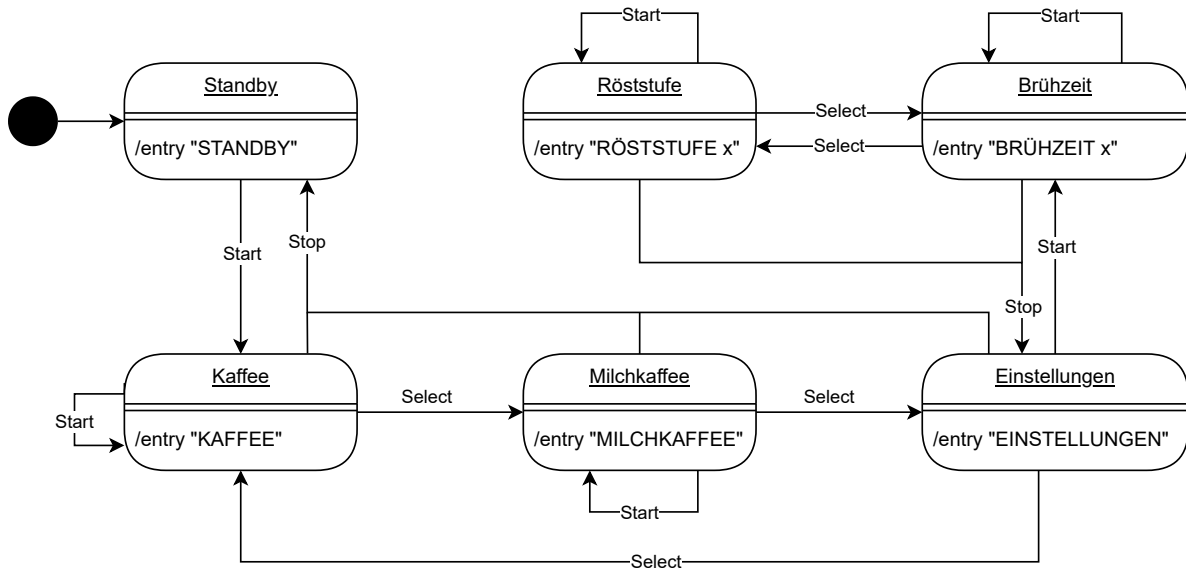


Abbildung 2: UML-State-Diagramm der Kaffeemaschine

Bei einem **Zustandswechsel** soll jeweils der **in den Zuständen angegebene Text in der Konsole erscheinen**. Wenn im Zustand *Brühzeit* oder *Röststufe* auf *Start* gedrückt wird, so soll zwischen den Einstellungen *HOCH*, *MITTEL*, und *NIEDRIG* gewechselt werden. Wenn im Zustand *Kaffee* oder *Milchkaffee* auf *Start* gedrückt wird, soll der gewählte Kaffee erstellt werden. **Simulieren Sie dies indem Sie einen entsprechenden Text in die Konsole ausgeben**. Geben Sie dabei auch die eingestellte Röststufe und Brühzeit aus.

- Setzen Sie das UML-State-Diagramm in Java-Programmcode um. Verwenden Sie dabei das **State-Muster**. (20 Pkt)
- Simulieren Sie einen Durchlauf ihres Programms indem Sie folgende Eingabefolge testen:  
*Start, Select, Select, Start, Start, Select, Stop, Select, Start, Stop* (5 Pkt)

10,00 → 10 GB

### Aufgabe 3

Punkte	Erreicht
30	

Ein Bildbearbeitungsprogramm möchte als neues Feature **Makros** hinzufügen, mit denen man **Filteroperationen** mehrmals auf ein Bild anwenden kann. Das Programm unterstützt bereits Filteroperationen für *Weichzeichnen* und *Rauschen hinzufügen*, welche mithilfe des Kommandomusters umgesetzt wurden. Mithilfe eines Makrobefehls soll eine Filteroperation drei mal oder fünf mal hintereinander ausgeführt werden (z.B. 3x Weichzeichnen oder 5x Rauschen hinzufügen).

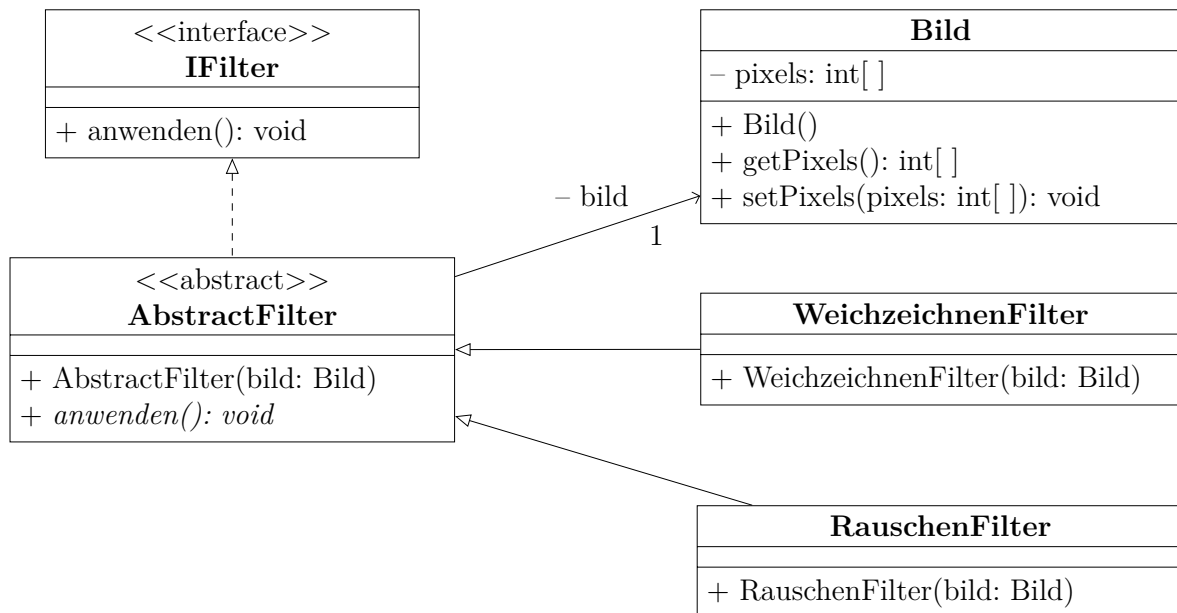


Abbildung 3: UML-Klassendiagramm der Filter-Commands

- Implementieren Sie das **Command-Muster** wie im UML-Diagramm gezeigt. Filteroperationen können dabei stellvertretend als **Konsolenausgabe** implementiert werden. (10 Pkt)
- Erweitern Sie das **UML-Klassendiagramm** um das beschriebene Makro-System, welches einen Befehl drei mal oder fünf mal hintereinander ausführen kann. Nutzen Sie dafür das **Decorator-Muster**. (10 Pkt)
- Erweitern Sie Ihr Programm aus Aufgabenteil (a), indem Sie Ihr UML-Klassendiagramm aus Aufgabenteil (b) implementieren. (10 Pkt)

## Aufgabe 4

Punkte	Erreicht
25	

Für ein Logistikunternehmen sollen Sie ein Programm schreiben, welches Lieferscheine für Pakete formatiert. Gegeben ist die Klasse `LieferscheinGUI.java`.

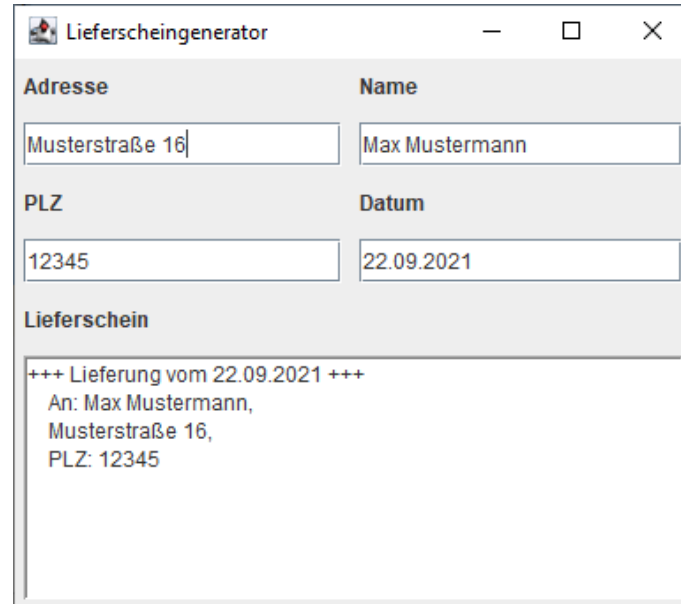


Abbildung 4: LieferscheinGUI.java

In der GUI sind die Werte Adresse, PLZ, Name, und Datum gegeben. Das Programm soll aufgrund der gegebenen Daten einen formatierten Lieferschein in das untere Textfeld ausgeben. Ein Lieferschein soll beispielsweise wie folgt formatiert werden:

```
+++ Lieferung vom 22.09.2021 +++  
  An: Max Mustermann,  
  Musterstraße 16,  
  PLZ: 12345
```

- (a) Implementieren Sie oben genannte Funktionalität und erweitern Sie die gegebene GUI-Klasse. Wird ein Parameter (Adresse, PLZ, Name, Datum) geändert, soll sofort der Lieferschein im Textfeld aktualisiert werden. Nutzen Sie dafür das **MVC-Muster**. (25 Pkt)

**Hinweis 1:** Die gegebene Klasse `LieferscheinGUI.java` dürfen Sie in Ihr Projekt importieren und verändern. Der Aufbau der Benutzeroberfläche soll jedoch nicht geändert werden.

**Hinweis 2:** Sie können auf Änderungen in einem `JTextField` mithilfe des Interfaces `DocumentListener` hören.