# Beleg  Programmierung 3  WS 20/21

Entwicklung einer mehrschichtigen und getesteten Anwendung

## Allgemeine Anforderungen

- Java8 (language level)
- Trennung zwischen Test- und Produktiv-Code
- JUnit5 als Testframework
- Mockito als Mock/Spy-framework
- main-Methoden nur im default package
- Abgabe als zip-Datei, welche ein lauffähiges IntelliJ-IDEA-Projekt enthält
- 100% Testabdeckung (additiv) außerhalb des default packages

## Testanforderungen

- nicht leer
- nicht tautologisch
- deterministisch
- keine Systemanforderungen
- betriebssystemunabhängig
- immer nur eine Methode unter Test und nur eine Zusicherung, oder Abweichung ist mit Kommentar begründet
- Zusicherung richtig befüllt

## Geschäftslogik

Erstellen Sie eine Geschäftslogik zur Verwaltung von Mediadateien mit begrenzter Kapazität. Die Typen der Mediadateien (`Content`) sind bereits als Interfaces definiert. Neben der Verwaltung für die Mediadateien ist eine Verwaltung von Produzenten (`Uploader`) zu realisieren. Die Geschäftslogik ist nur für die Verwaltung zuständig, nicht das tatsächliche Hochladen oder Speichern von Dateien.

Die Geschäftslogik muss folgende Funktionalität realisieren:

- Anlegen von Produzenten; dabei muss sichergestellt sein, dass kein Name mehr als einmal vorkommt
- Hochladen von Mediadateien

    - unterstützt werden alle Typen die sowohl von `Uploadable` als auch `MediaContent` ableiten
    - es ist zu prüfen, dass die Media-Datei zu einem bereits existierenden Produzenten gehört
    - es ist zu prüfen, dass die Gesamtkapazität nicht überschritten wird, dafür ist die Dateigröße in `size` definiert
    - beim Hochladen wird eine Abrufadresse vergeben (`address`); zu keinem Zeitpunkt können mehrere Mediadateien innerhalb der Verwaltung die gleiche Abrufadresse haben
    - beim Hochladen wird ein Upload-Datum vergeben
- Abruf aller Produzenten mit der Anzahl der ihrer Mediadateien
- Abruf vorhandener Mediadateien; wird ein Typ angegeben werden nur Mediadateien von diesem Typ aufgelistet
- Abruf aller vergebenen bzw. nicht vergebenen Tags in der Verwaltung
- Hochsetzen der Abrufe (`accessCount`) gegen Übergabe der Instanz oder Adresse
- Löschen eines Produzenten gegen Übergabe der Instanz oder Name
- Löschen einer Mediadatei gegen Übergabe der Instanz oder Adresse

## CLI

Implementieren Sie eine Benutzeroberfläche. Die Kommunikation zwischen Oberfläche und Geschäftslogik soll dabei über events erfolgen.

Weiterhin sollen nach dem Beobachterentwurfsmuster 2 Beobachter realisiert werden: der Erste soll eine Meldung produzieren wenn 90% der Kapazität überschritten werden, der Zweite über Änderungen an den vorhandenen Tags informieren. Beachten Sie dass diese erweiterte Funktionalität nicht zur Geschäftslogik gehört.

Das UI soll als zustandsbasiertes (Einfüge-, Anzeige-, Lösch- und Änderungs-Modus, …) command-line interface (CLI) realisiert werden.

Stellen Sie sicher, dass Bedienfehler in der Eingabe keine unkontrollierten Zustände in der Applikation erzeugen.

Beim Starten der Anwendung sollen die Argumente ausgelesen werden. Ist eine Zahl angegeben ist dies die Kapazität. Ist `TCP` oder `UDP` angegeben ist die

Applikation als Client für das entsprechende Protokoll zu starten. Dabei kann davon ausgegangen werden, dass der jeweilige Server bereits läuft.

### Befehlssatz

- `:c` Wechsel in den Einfügemodus
- `:d` Wechsel in den Löschmodus
- `:r` Wechsel in den Anzeigemodus
- `:u` Wechsel in den Änderungsmodus
- `:p` Wechsel in den Persistenzmodus
- `:config` Wechsel in den Konfigurationsmodus

Einfügemodus:

- `[Produzentenname]` fügt einen Produzent ein
- `[Media-Typ] [Produzentenname] [kommaseparierte Tags, einzelnes Komma für keine] [Bitrate] [Länge] [[Encoding] [Höhe] [Breite] [Samplingrate] [Interaktionstyp] [Lizenzsgeber]]` fügt eine Mediadatei ein; Beispiele:
  - `InteractiveVideo Produzent1 Lifestyle,News 5000 3600 DWT 640 480 Abstimmung`
  - `LicensedAudioVideo Produzent1 , 8000 600 DCT 1400 900 44100 EdBangerRecords`

Anzeigemodus:

- `uploader` Anzeige der Produzenten mit der Anzahl der hochgeladenen Dateien
- `content [[Typ]]` Anzeige der Mediadateien - ggf. gefiltert nach Typ[1] - mit Abrufadresse, Upload-Datum und Anzahl der Abrufe
- `tag [enthalten(i)/nicht enthalten(e)]` Anzeige der vorhandenen bzw. nicht vorhandenen Tags

Löschmodus:

- `[Produzentenname]` löscht den Produzenten
- `[Abrufadresse]` löscht die Mediadatei

[1] Typ meint hier die vorgegebenen Interfaces, nicht die Eigenschaft in Interactive

Änderungsmodus:

- `[Abrufadresse]` erhöht den Zähler für die Abrufe um eins

Persistenzmodus:

- `saveJOS` speichert mittels JOS
- `loadJOS` lädt mittels JOS
- `saveJBP` speichert mittels JBP
- `loadJBP` lädt mittels JBP
- `save [Abrufadresse]` speichert eine einzelne Instanz in eine Datei für alle Instanzen, falls die Datei nicht existiert werden alle Instanzen in eine neue gespeichert
- `load [Abrufadresse]` lädt eine einzelne Instanz aus der Datei

Konfigurationsmodus:

- `add [Klassenname]` registriert einen benannten Beobachter bzw. listener
- `remove [Klassenname]` de-registriert einen benannten Beobachter bzw. listener

## Simulation

Stellen Sie sicher, dass die Geschäftslogik thread-sicher ist. Erstellen Sie dafür eine Simulation, die die Verwendung der Geschäftslogik im Produktivbetrieb testet. Die Abläufe in der Simulation sollen auf der Konsole dokumentiert werden. Jeder thread muss für jede ändernde Interaktion an der Geschäftslogik eine Ausgabe produzieren. Jede Änderung an der Geschäftslogik muss eine Ausgabe produzieren, sinnvollerweise über Beobachter.

Zur Entwicklung darf `Thread.sleep` o.ä. verwendet werden, in der Abgabe muss dies deaktiviert sein bzw. darf nur mit Null-Werten verwendet werden.

### Simulation 1

Erstellen Sie einen thread der kontinuierlich versucht eine zufällig erzeugte Mediadatei einzufügen. Erstellen Sie einen weiteren thread der kontinuierlich die Liste der enthaltenen Mediadateien abruft, daraus zufällig eine auswählt und löscht. Diese Simulation sollte nicht terminieren.

### Simulation 2

Erstellen Sie einen weiteren thread der kontinuierlich die Liste der enthaltenen Mediadateien abruft, daraus zufällig eine auswählt und den Abruf auslöst (`accessCount` inkrementiert). Modifizieren Sie den löschenden thread so, dass er die Mediadatei mit den wenigsten Abrufen löscht. Sorgen Sie mit wait/notify dafür, das Einfügen und Löschen miteinander synchronisiert sind. D.h. wenn das Einfügen wegen Kapazitätsmangel nicht möglich ist wird der löschende thread benachrichtigt und während dieser arbeitet wartet der einfügende thread. Umgekehrt arbeitet auch der löschende thread nicht während der Ausführung des Einfügens.

### Simulation 3

Modifizieren Sie den löschenden thread aus der zweiten Simulation so dass er eine zufällige Anzahl (inklusive 0) von Mediadateien löscht, wobei diese weiterhin das Kriterium zu den wenigsten Abrufen erfüllen. Starten Sie die Simulation mit mindesten je zwei einfügenden und löschenden threads.

## GUI

Realisieren Sie eine skalierbare graphische Oberfläche mit JavaFX für die Verwaltung. Sie soll den gleichen Funktionsumfang wie das CLI haben, abzüglich des Konfigurationsmodus und der Beobachter. Die Auflistung der Produzenten und Mediadateien soll immer sichtbar sein und nach Benutzeraktionen automatisch aktualisiert werden.

Die Auflistung der Mediadateien soll sortierbar nach Abrufadresse, Anzahl der Abrufe und Produzent sein.

Ermöglichen Sie die Änderung der Abrufadresse mittels drag&drop.

## I/O

Realisieren Sie die Funktionalität den Zustand der Geschäftslogik zu laden und zu speichern.

Der Anwender kann wählen ob die Persistierung mit JOS oder JBP erfolgt.

Implementieren Sie weiterhin die Möglichkeit einzelne Mediadateien wahlfrei (random access auf Dateisystemebene) zu Laden und zu speichern.

## net

Realisieren Sie die Verwendung von CLI und Geschäftslogik in verschiedenen Prozessen. Die Verbindung soll wahlweise über TCP oder UDP erfolgen.

Der Server wird mit 2 Argumenten gestartet: Protokoll und Lagerkapazität.

Ermöglichen Sie die Verwendung mehrerer Clients, die sich auf einen gemeinsamen Server verbinden für TCP oder UDP.

Die Berücksichtigung von Skalierbarkeit, Sicherheit und Transaktionskontrollen ist nicht gefordert. Auch die Beobachter und der Konfigurationsmodus müssen im Netzwerkmodus nicht unterstützt werden.

## Zusätzliche Anforderungen

Realisieren sie ein optionales, mehrsprachiges Logsystem.

Wird beim Starten der Applikation (CLI oder GUI) ein ISO 3166 Länderkürzel als Kommandozeilenparameter übergeben wird das Log verwendet. Ist kein Kürzel angegeben wird es nicht verwendet. In dem Log wird jede Benutzerinteraktion, die die Geschäftslogik erreicht, in einer Zeile dokumentiert. Außerdem werden alle Änderungen am Zustand der Geschäftslogik ebenfalls in einer Zeile dokumentiert.

Der Dateiname für das Log ist „log.txt“. Beachten Sie, dass das Log eine singuläre Ressource ist und geschützt werden muss. Auch gehört das Log nicht zur Geschäftslogik.

Implementieren Sie das das Log vollständig für DE und prototypisch mit vier übersetzten Log-Einträgen für EN. Berücksichtigen Sie, dass theoretisch beliebige Sprachen unterstützt werden sollen. Encoding und Schreibrichtung müssen nicht berücksichtigt werden.

## Dokumentation

- Architekturdiagramm mit allen Schichten und der Einordung aller packages
- ggf. Begründungen
- Quellennachweise (s.u.)

## Quellen

Zulässige Quellen sind suchmaschinen-indizierte Internetseiten. Werden mehr als drei zusammenhängende Anweisungen übernommen ist die Quelle in den

Kommentaren anzugeben. Ausgeschlossen sind Quellen, die auch als Beleg oder Übungsaufgabe abgegeben werden oder wurden. Zulässig sind außerdem die über moodle bereitgestellten Materialien, diese können für die Übungsaufgaben und Beleg ohne Quellenangabe verwendet werden.

## Bewertungsschema

| | | |
|---|---|---|
| Entwurf | Schichtenaufteilung* | 3 |
| | Architekturdiagramm | 1 |
| | Zuständigkeit | 2 |
| | Paketierung | 2 |
| | Benennung | 2 |
| | keine Duplikate | 1 |
| Tests | sichtbare & terminierende Methoden getestet* | 7 |
| | jede Anweisung getestet* | 7 |
| | Mockito richtig verwendet* | 5 |
| | Spytests (Verhalten) | 3 |
| | keine unbeabsichtigt fehlschlagenden Test | 1 |
| Fehler-freiheit* | kein konstruierter fehlschlagender Test | 5 |
| | keine Ablauffehler | 5 |
| Basis-funktionalität* | CRUD | 2 |
| | CLI | 2 |
| | Simulation 1 | 2 |
| | GUI | 2 |
| | I/O | 2 |
| | Net | 2 |
| Funktionalität | vollständige, threadsichere GL | 2 |
| | vollständiges CLI inkl. Laufzeitkonfiguration | 2 |
| | vollständiges GUI | 1 |
| | events (mindestens 3) | 2 |
| | observer ^ property change propagation | 2 |
| | angemessene Aufzählungstypen | 2 |
| | drag&drop | 1 |
| | Simulationen 2 & 3 | 2 |
| | data binding | 1 |
| | JBP und JOS | 2 |
| | random access | 1 |
| | mehrere Clients | 1 |
| extra Anforderung | Nachrichten an die GL | 2 |
| | Änderungen an der GL | 2 |
| | richtige Einordnung in der Schichtenarchitektur | 2 |
| | Behandlung als Ressource | 1 |
| | Mehrsprachigkeit | 1 |

gesamt: 83