Beleg Programmierung 3 WS 25/26

Entwicklung einer mehrschichtigen und getesteten Anwendung

Geschäftslogik

Erstellen Sie eine Geschäftslogik zur Verwaltung von Mediadateien mit begrenzter Kapazität. Die Typen der Mediadateien (`MediaContent`) sind bereits als Interfaces definiert im module `contract` im Projekt `belegProg3` definiert. Neben der Verwaltung für die Mediadateien ist eine Verwaltung von Produzent*innen (`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 Produzent*innen; dabei muss sichergestellt sein, dass kein Name mehr als einmal vorkommt
- Einfügen von Mediadateien:
  - unterstützt werden alle Typen die sowohl von `Uploadable` als auch `MediaContent` ableiten
  - es ist zu prüfen, dass die Media-Datei zu einer bereits existierenden Produzent*in 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 Produzent*innen mit der Anzahl der ihrer Mediadateien
- Abruf vorhandener Mediadateien; wird ein Typ (z.B. als Klassenname) angegeben werden nur Mediadateien von diesem Typ aufgelistet
- Abruf aller vergebenen und nicht vergebenen Tags in der Verwaltung
- Hochsetzen der Abrufe (`accessCount`)
- Löschen einer bestimmten Produzent*in
- Löschen einer bestimmten Mediadatei
- Abruf der vordefinierten Kapazität

Verwaltete Objekte

Alle von der Geschäftslogik verwalteten Mediadateien müssen mindestens folgende Eigenschaften haben:

- Abrufkosten (BigDecimal)
- Abrufzähler (int)
- Abrufadresse (string)
- Produzent*innen
- Tags (beliebige Anzahl)
- Größe (long)
- Verfügbarkeit (Duration), ergibt sich aus der Differenz zwischen dem aktuellen Datum und dem Upload-Datum

Es gibt 3 Arten von Mediadateien mit folgenden Eigenschaften:

- Audio
  - SamplingRate (int)
- Video
  - Resolution (int)
- AudioVideo
  - SamplingRate (int)
  - Resolution (int)

Es gibt 4 enum-kodierte Tags: *Animal*, *Review*, *Music*, *News*.

Produzent*innen besitzen mindestens eine Eigenschaft Name (String).

CLI

Implementieren Sie eine konsolenbasierte 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. Jeder Befehl wird mit einem Zeilenwechsel abgeschlossen. Eine Menüführung, insbesondere für Befehle mit mehreren Token, ist nicht gewünscht.

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

Beim Starten der Anwendung sollen die Argumente[1] 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 und an ihm die Kapazität gesetzt wurde.

Befehlssatz

- `:c` Wechsel in den Einfügemodus
- `:r` Wechsel in den Anzeigemodus
- `:u` Wechsel in den Änderungsmodus
- `:d` Wechsel in den Löschmodus
- `:p` Wechsel in den Persistenzmodus
- `:x` beendet die Anwendung

Einfügemodus:

- `[P-Name]` fügt eine Produzent*in ein
- `[Media-Typ] [P-Name] [kommaseparierte Tags, einzelnes Komma für keine] [Größe] [Abrufkosten] [[Optionale Parameter]]` Fügt eine Mediadatei ein, die optionalen Parameter sind abhängig vom Typ, ihre Reihenfolge wird durch den Typ-Namen bestimmt. Bei keiner Angabe sind default-Werte zu vergeben. Beispiele:
  - `AudioVideo RedLetterMedia Review 8700 3,60 44100 1080`
  - `Audio EdBangerRecords , 1200 2,35 44100`

Anzeigemodus:

- `uploader` Anzeige der Produzent*innen mit der Anzahl der Mediadateien

[1] https://www.jetbrains.com/help/idea/2023.1/run-debug-configuration.html

- `content [[Typ]]` Anzeige der Mediadateien - ggf. gefiltert nach Typ - mit Abrufadresse, Verfügbarkeit in Sekunden und Anzahl der Abrufe. Diese Darstellung muss nur so aktuell wie der letzte Abruf sein.
- `tag [enthalten(i)/nicht enthalten(e)]` Anzeige der vorhandenen bzw. nicht vorhandenen Tags

Löschmodus:

- `[P-Name]` löscht die Produzent*in
- `[Abrufadresse]` löscht die Mediadatei

Änderungsmodus:

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

Persistenzmodus:

- `save [JOS|JBP]` speichert den Zustand der Geschäftslogik mittels der angegebenen Technologie
- `load [JOS|JBP]` lädt den Zustand der Geschäftslogik mittels der angegebenen Technologie

alternatives CLI

Erstellen sie ein alternatives CLI mit eigener main-Methode in dem 2 Funktionalitäten (vorzugsweise Löschen von Produzent*innen und Auflisten der Tags) deaktiviert sind und nur ein Beobachter aktiv ist. Dieses CLI soll sich vom eigentlichen CLI nur durch die Konfiguration unterscheiden, nicht durch die Implementierung. D.h. der Unterschied besteht nur im setup in der main-Methode beim Einhängen der listener bzw. handler. Welche Funktionalitäten deaktiviert sind muss dokumentiert werden. Dieses CLI muss nicht über das Netzwerk funktionieren.

Simulation

Stellen Sie sicher, dass die Geschäftslogik thread-sicher ist. Erstellen Sie dafür Simulationen, die die Verwendung der Geschäftslogik im Produktivbetrieb testen. Die Abläufe in der Simulation sollen auf der Konsole dokumentiert werden.

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

Jede Simulation sollte eine eigene main-Methode haben und die Kapazität der GL per Kommandozeilenargument annehmen, wobei 0 ein zulässiger Wert ist.

Simulation 1

Erstellen Sie einen thread (Produzent) der kontinuierlich versucht eine zufällig erzeugte[2] Mediadatei in die Medienverwaltung einzufügen. Erstellen Sie einen weiteren thread (Konsument) der kontinuierlich die in der Verwaltung enthaltenen Mediadateien abruft, daraus zufällig eine auswählt und löscht. Diese Simulation sollte nicht terminieren und nicht wait/notify (bzw. await/signal) verwenden. Beide threads sollen für jede Interaktion an der Geschäftslogik eine Ausgabe produzieren. Außerdem soll jede Änderung an der Geschäftslogik eine Ausgabe produzieren, sinnvollerweise über Beobachter, ansonsten über den verursachenden thread.

Simulation 2

Erweitern Sie die Simulation 1 so, daß beim Start neben der Kapazität auch eine Anzahl *n* beim Start übergeben wird. In der Simulation sollen dann *n* Produzenten und *n* Konsumenten gestartet werden. Die Änderungen an der Geschäftslogik müssen jetzt durch einen Beobachter auf der Konsole ausgegeben werden.

Simulation 3

Erstellen Sie einen weiteren thread der kontinuierlich die Liste der enthaltenen Mediadateien abruft, daraus zufällig eine auswählt und den Abruf auslöst. Modifizieren Sie den Konsumenten so, dass er die Mediadatei mit dem höchsten Abrufzähler löscht. Sorgen Sie mit wait/notify (bzw. await/signal) dafür, das Einfügen und Löschen miteinander synchronisiert sind. D.h. wenn das Einfügen wegen Kapazitätsmangel nicht möglich ist werden die Konsumenten benachrichtigt und während diese arbeiten warten die Produzenten. Umgekehrt arbeiten auch die Produzenten nicht während der Ausführung des Löschens.

Beim Start der Simulation 3 werden drei Parameter übergeben, die Kapazität, die Anzahl der jeweils zu startenden threads (Einfügen, Löschen und Update) und ein Intervall in Millisekunden. Die Änderungen an der Geschäftslogik werden jetzt nicht mehr durch einen Beobachter ausgegeben, sondern dafür soll ein thread implementiert werden, der in regelmäßigen Abständen (übergebenes Intervall)

[2] zufällige Auswahl aus einer vordefinierten Liste ist auch zulässig

den Zustand der Geschäftslogik liest und ausgibt. Von diesem thread soll immer nur eine Instanz über den `ScheduledExecutorService` gestartet werden.

GUI

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

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

Ermöglichen Sie die Tausch der Abrufadresse mittels drag&drop.

Das Einfügen der Mediadateien sollte nicht sperrend sein, es sollte also möglich sein, während der Einfügens andere Funktionen oder ein weiteres Einfügen auszulösen. D.h. die Ausführung des Einfügens sollte nebenläufig erfolgen.

I/O

Realisieren Sie die Funktionalität den Zustand der Geschäftslogik zu laden und zu speichern. Der Dateinamen ist nicht vorgegeben, darf aber keine Pfade enthalten. (Damit wird die Datei im Ausführungsverzeichnis gespeichert.)

Die Beobachter gehören hierbei nicht zum Zustand der Geschäftslogik. Nach dem Laden der Geschäftslogik müssen sie nicht wieder eingehangen werden.

Die Anwender*innen können wählen ob die Persistierung mit JOS oder JBP erfolgt.

Net

Realisieren Sie die Verwendung von CLI und Geschäftslogik in verschiedenen Prozessen. Die Verbindung soll in Abhängigkeit vom übergebenen Kommandozeilenargument über TCP oder UDP erfolgen. Der Server wird mit 2 Argumenten gestartet: Protokoll und Kapazität.

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 müssen im Netzwerkmodus nicht unterstützt werden.

zusätzliche Anforderungen

werden im PZR bekanntgegeben

Anforderungen an Testqualität

- eine falsifizierbare Aussage zu einem Methodenaufruf
  - nicht leer (noch zu implementierende Test werden mit `fail` befüllt)
  - enthalten genau eine Zusicherung (ggf. Abweichung kommentieren)
  - Ergebnis ist vom Produktiv-Code abhängig
- ein Ablaufpfad
  - keine Schleifen, Verzweigungen
  - Abzweigungen terminieren mit `fail`
- überall schnell ausführbar
  - betriebssystemunabhängig
  - kein Zugriff auf Dateisysteme, Netzwerk, o.Ä.
  - deterministisch, d.h. keine Zufallsgeneratoren, keine threads
  - schnell, d.h. kein `sleep`, keine Lasttests mit hunderten von Einträgen
- gut lesbar
  - Tests in richtiger Testklasse
  - Testklassen im richtigen package und module
  - sprechender Testname
  - Zusicherung richtig befüllt
  - Duplikate sind zulässig
  - sparsame und angemessene Verwendung von `@BeforeEach` (Ist erfüllt, wenn jeder Test in der Testklasse vom gesamten setup abhängig ist.)

Dokumentation

- befüllte Checklist.md im root des Projektes
- 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.

Flüchtige Quellen, wie Sprachmodelle, sind per screen shot zu dokumentieren.

Bewertung

siehe Checklist.draft.md

Kapselung

überprüft anhand von Testfällen die Kapselung der Implementierung, z.B.:

1. erzeuge Verwaltung mit zwei Mediadateien
2. hole Aufzählung der Mediadateien
3. lösche diese Aufzählung (`clear()`)
4. hole Aufzählung der Mediadateien erneut
5. prüfe Inhalt dieser Aufzählung (sollte nicht leer sein)

oder:

1. erzeuge Verwaltung mit zwei Mediadateien
2. hole Aufzählung der Mediadateien
3. wenn an den Objekten aus der Aufzählung der setter für die Abrufadresse sichtbar ist, setze bei allen Objekten diese auf eine anderen Wert
4. hole Aufzählung der Mediadateien erneut
5. prüfe Inhalt dieser Aufzählung (die Abrufadressen sollten unverändert sein)

wenn das Anlegen und Einfügen von Objekten von außerhalb der GL möglich ist:

1. erzeuge Verwaltung
2. erzeuge Mediadatei
3. füge Mediadatei in Verwaltung ein
4. hole Aufzählung der Mediadateien

5. wenn an der in 2. erzeugten Mediadatei der setter für die Abrufadresse sichtbar ist, setze diese auf eine anderen Wert
6. hole Aufzählung der Mediadateien erneut
7. prüfe Inhalt dieser Aufzählung (die Abrufadresse sollte unverändert sein)

keine Ablauffehler

prüft anhand von use cases die Korrektheit der Implementierung, z.B.:

```
:c
Alice
Audio Alice Music 1200 2,05 44100
AudioVideo Alice Review,Music 7600 2,25
:r
content Audio
tag i
uploader
```