Software Engineering I - Teil 02

Komplexaufgabe | Paketsortierung

Ein **Paket** ist charakterisiert durch [i] alphanumerische 6-stellige id aus dem Zeichenpool [a-z, 0-9], [ii] content (L: 25, B: 10, H: 10) mit einer Länge von zufällig gewählten 2500 aus dem Pool [a-z | . | : | - | !], [iii] sechsstelliger zip_code aus dem Bereich [01067-99998], [iv] type aus dem Bereich [NORMAL | EXPRESS | VALUE] und [v] weight aus dem Bereich [1.00 bis 5.00]. Die Verteilung für type ist [i] NORMAL: 80%, [ii] EXPRESS: 15% und [iii] VALUE: 5%. Es sind **24.000 Pakete** zu erzeugen und zu mischen (shuffle). Es ist einmalig eine **CSV-Datei base_package.csv im Format [id],[content],[zip_code],[type],[weight] mit 24.000 Zeilen** zu erstellen. In vier ausgewählten Paketen ist im content die Zeichenkette exp!os:ve an einer beliebigen Stelle zu integrieren.

Eine **Box** ist charakterisiert durch alphanumerische 5-stellige id aus dem Zeichenpool [a-z, 0-9] und hat eine Kapazität für 40 Pakete. Eine Box hat fünf Ebenen. Eine Ebene hat auf der linken und rechte Seite eine Kapazität für je vier (hintereinander angeordnete) Pakete. Die 24.000 Pakete sind sukzessive auf **600 Boxen** zu verteilen. Es ist einmalig eine **CSV-Datei base_box.csv im Format [box_id],[package_id] mit 600 Zeilen** zu erstellen.

Eine **Palette** ist charakterisiert durch eine fortlaufende nummerische id (beginnend bei 1) und hat 2x2 Positionen. Jede Position hat eine Kapazität für drei (aufeinandergestapelte) Boxen. Die 600 Boxen sind sukzessive auf **50 Paletten** zu verteilen. Es ist einmalig eine **CSV-Datei base_pallet.csv im Format [pallet_id],[position],[level],[box_id]** zu erstellen. Bsplw. bedeutet 1,2,2,23 dass die Box 23 auf der Palette 1 an der Position 2 und der Ebene 2 gelagert ist.

Ein LKW ist charakterisiert durch eine alphanumerische 4-stellige id aus dem Bereich [A-Z,0-9] und hat einen Anhänger. Der Anhänger hat auf der linken und rechten Seite eine Kapazität für je fünf (hintereinander angeordnete) Paletten. Zu Simulationszwecken werden sukzessive **fünf LKW mit je 10 Paletten** beladen. Es ist einmalig eine **CSV-Datei im Format [truck_id],[left | right], [position],[pallet_id]** zu erstellen.

Das **Paketsortierzentrum** verfügt über [i] eine zentrale Steuerungseinheit (ZS), [ii] sieben Zonen für die Entladung von LKW, [iii] eine Parkzone für fünf autonome Fahrzeuge für den Transport von Paletten sowie [iv] eine Sortieranlage.

Die **ZS verwaltet** den **EventBus** basierend auf Google Guava.

Bezüglich Zugriff und Berechtigungen für die ZS werden die **Rollen** [i] Supervisor, [ii] Administrator, [iii] Operator und [iv] Data Analyst unterschieden.

Supervisor, Administrator, Operator und Data Analyst sind Mitarbeiter. Mitarbeiter ist charakterisiert durch [i] ganzzahlige id und [ii] name. Supervisor hat zusätzlich das boolesche Attribut isSenior. Administrator hat zusätzlich das Attribut profile mit den zulässigen Werten A, B und C.

Jedem Mitarbeiter (**Employee**) ist eine **IDCard** zugeordnet. Der Mitarbeiter kennt die IDCard, die IDCard jedoch nicht den Mitarbeiter. Auf einem der IDCard zugeordneten Magnetstreifen (L: 100, B: 1) wird die Zeichenkette [id];[name];[role];[pin];[super pin] verschlüsselt gespeichert. Für die Verschlüsselung stehen die Strategien (**Strategy**) AES und DES zur Verfügung. In einer zentralen Configuration – realisiert als Enumeration – wird der angewandte Algorithmus und der Schlüssel dhbw definiert.

Das Terminal verfügt über ein Lesegerät für die IDCard. Der Mitarbeiter zieht seine IDCard durch das Lesegerät und wird danach aufgefordert seine PIN (nummerisch, 4-stellig) einzugeben. Die IDCard hat die Status Active, Locked und Invalid (**State**). Initial hat die IDCard den Status Active.

Bei dreimaliger Eingabe einer inkorrekten PIN wechselt die IDCard in den Status Locked und die Eingabe der Super-PIN (nummerisch, 6-stellig) ist erforderlich. Bei zweimaliger Eingabe einer inkorrekten SuperPIN wechselt die IDCard in den Status Invalid. Eine IDCard im Status Invalid wird vom Lesegerät abgewiesen und es ist keinerlei Eingabe von PIN/SuperPIN mehr möglich.

Über das Terminal mit einem zwischengelagerten **Proxy** auf die ZS wird die Sortieranlage mit den Kommandos (**Command**) [i] init, [ii] next, [iii] shutdown, [iv] lock, [v] unlock, [vi] show statistics und [vii] change search algorithm to [bm | rk] gesteuert. Folgende Berechtigungen sind auf dem Proxy konfiguriert [i] Supervisor | alle Kommandos, [ii] Administrator | shutdown u. show statistics, [iii] Operator | next und show statistics und [iv] Data Scientist | show statistics. Die Kommandos werden über ein Terminal mit einem TouchPad abgesetzt.

Mit **init** werden die fünf LKW mit je einem Anhänger auf Basis der CSV-Dateien erstellt und in einer Wartezone mit fünf Positionen nebeneinander geparkt. Mit **next** fährt der nächste LKW zu einer zufällig ausgewählten Zone für die Entladung. Mit **shutdown** werden die Sensoren an der Zone für die Entladung der LKW deaktiviert sowie die Komponenten bei den Scanner entladen. Mit lock und unlock wird die Sortieranlage ge-/entsperrt. Mit **show statistics** wird ein Bericht mit den Informationen erstellt (**Builder**) [i] Aktuelles Datum und Uhrzeit, [ii] Anzahl der aktuell abgefertigten LKW, [iii] Anzahl der aktuell gescannten Pakete gruppiert nach type und [iv] Pakete mit gefährlichem Gegenstand exp!os:ve. Dieser Bericht wird in eine Textdatei report.txt gespeichert.

In einer **Parkzone** befinden sich fünf autonome Fahrzeuge für den Transport von Paletten von der Zone für die Entladung der LKW zum vorgelagerten Roboter der Sortieranlage. Die autonomen Fahrzeuge sind über einen EventBus mit der ZS verbunden.

Jede **Zone** für die **Entladung** eines **LKW** ist mit einem Sensor (**Observer**) ausgestattet. Bei Ankunft eines LKW wird die ZS automatisch informiert. Die ZS sendet ein Event mit der ID der Zone an ein zufällig ausgewählte autonomes Fahrzeug.

Vor der **Sortieranlage** existiert ein **Bereich** für die **Zwischenlagerung von Paletten**. Dieser Bereich hat 5 Positionen. Jede Position hat eine Kapazität für maximal zwei aufeinandergestapelte Paletten. Das beauftragte autonome Fahrzeug entlädt vollständig den LKW und transportiert die Paletten zu dem Bereich für die Zwischenlagerung. Nach der Entladung fährt das autonome Fahrzeug zurück zum Parkplatz in der Parkzone. Nach Ankunft sendet das autonome Fahrzeug ein Event an die ZS, dass der LKW entladen wurde.

Die Sortieranlage verfügt über [i] einen vorgelagerten Roboter, [ii] einen Lagerplatz für leere Boxen, [iii] einen Lagerplatz für leere Paletten, [iv] acht Lagerbahnen mit einer Kapazität von je 600 Paketen, [v] einen Sensor für die Messung des Füllgrades einer Lagerbahn, [vi] drei Sortierbahnen mit den Zuständigkeiten Normal, Express und Value.

Die ZS sendet ein Event an den vorgelagerten Roboter der Sortieranlage. Der Roboter entnimmt sukzessive die Boxen von der Palette, entleert diese und füllt die acht Lagerbahnen sukzessive auf. Jede Lagerbahn verfügt über einen Sensor (**Observer**). Ist eine Lagerbahn befüllt, wird vom Sensor ein Event an die ZS kommuniziert.

Hat die zentrale Steuerungseinheit von allen acht Bahnen ein Event erhalten, sendet die ZS ein Event für die Sortierung nach Normal, Express und Value (**Chain of Responsibility**) in eine zuständige Sortierbahn. Im Rahmen der Sortierung werden die Sortierbahnen sukzessive geleert.

Jede Sortierbahn verfügt über einen Scanner bezüglich dem verbotenen Gegenstand exp!os:ve. Dem Scanner stehen zwei dynamisch austauschbare Komponenten BoyerMoore und RabinKarp zur Verfügung. Mit dem Kommando change search algorithm wird der angewandte Algorithmus für alle eingesetzten einheitlich geändert.

In die bestehende Architektur ist eine **DataAnalyticsEngine für** den **Data Scientist** zu integrieren. Jeder Scanner dokumentiert in einem **Logfile scanner_[type].log** die Informationen [type] | [start_timestamp] | [end_timestamp] | [weight] | algorithm [bm |rk] | found [none | explosive]. Das Attribut type kann die Werte [normal | express | value] annehmen. Das Attribut start_timestamp ist der Zeitstempel in Nanosekunden vor dem Scan des Paketes. Das Attribut end_timestamp ist der Zeitstempel in Nanosekunden nach dem Scan des Paketes. Das Attribut algorithm ist der Wert des für den Scan eingesetzten Algorithmus.

Für die statistische Auswertung stellt die DataAnalyticsEngine ein **RO-Interface** auf die Logfile-Informationen zur Verfügung.

Der Data Scientist setzt über den Proxy das Kommando show statistics ab.

Durch einen **Builder** und **Lamda/Streams** wird die nachfolgend spezifizierte statistische Auswertung erzeugt und in eine Textdatei statistical_analysis.txt geschrieben.

```
Zeitstempel // Format dd.MM.yyyy hh:mm:ss
# total trucks   | [total no trucks)
# total packages | [total no packages]
type  | [n] no_normal | [e] no_express | [v] no_value
average weight [normal) | [average weight for normal packages]
# packages with explosives | [no packages with explosives]
[package_id] | [type] | [weight] | algorithm [bm | kp]
```

Wichtige Hinweise

- **Bearbeitung im Team** durch zwei (im Fall von Team 36: drei) Studierende.
- Verwendung geeigneter **englische**r Begriffe für **Namen** und **Bezeichnungen**.
- Weitestgehende **Berücksichtigung** der **SOLID-Design Prinzipien**.
- Als **Entwicklung**sumgebung wird [i] **Java SE Development Kit 15.0.2**,
  [ii] **IntelliJ IDEA Community oder Ultimate 2020.3.2 und** [iii] **gradle** genutzt.
- **Modellierung Klassendiagramm** und **Komponentendiagramm** in Visual Paradigm.
  Bitte
  - achten Sie auf ein geordnetes Gesamtbild.
  - erstellen Sie ein **Unterverzeichnis diagram für** die **Speicherung** der **vpp-Datei**.
  - benennen Sie die **Datei** mit **komplexaufgabe_[team_id].vpp**.
  - benennen Sie das Klassendiagramm mit team_[team_id].
  - benennen Sie das Komponentendiagramm mit team_[team_id]
  - löschen Sie die *.bak-Dateien von Visual Paradigm.
- **Implementierung** einer technisch einwandfrei lauffähigen Applikation.
  Bitte
  - erstellen Sie ein **IntelliJ-Projekt im Unterverzeichnis implementation**.
  - nutzen Sie die **camelCase-Notation**, um die Lesbarkeit zu vereinfachen.
  - kommentieren Sie an geeigneten Stellen die Anwendung der SOLID-Design Prinzipien.
- Es sind **dedizierte Tests in JUnit für** folgende **Szenarien** zu realisieren.

  Setup des Paketsortierzentrums mit [i] einer zentralen Steuerungseinheit (ZS), [ii] sieben Zonen für die Entladung von LKW, [iii] einer Parkzone für fünf autonome Fahrzeuge für den Transport von Paletten sowie [iv] einer Sortieranlage.

  Das beauftragte autonome Fahrzeug entlädt vollständig den LKW und transportiert die Paletten zu dem Bereich für die Zwischenlagerung.
- **Clean-Up** und **Formatierung** des **Source Code**.
  (Code ► Reformat Code [Optimize imports, Rearrange entries, Cleanup code])
- Je Team wird eine unverschlüsselte **7-Zip-Datei komplexaufgabe_[team_id].7z**
  (Kompressionsstärke: Ultra) mit der **vpp-Datei**, dem vollständigen **IntelliJ-Projekt** und den **CSV-Datendateien in Moodle hoch**ge**laden**.
- **Zeitansatz:** 40 Stunden je Studierenden
- **Abgabetermin:** **Upload in Moodle bis** spätestens Sonntag, **04.04.2021**.
- **Bewertung**: **20 Punkte je Studierenden im Team**
  Modellierung 8 Punkte, Implementierung 10 Punkte, Test 2 Punkte.