Technische Hochschule Ingolstadt

Probeklausur Architektur und Entwurfsmuster

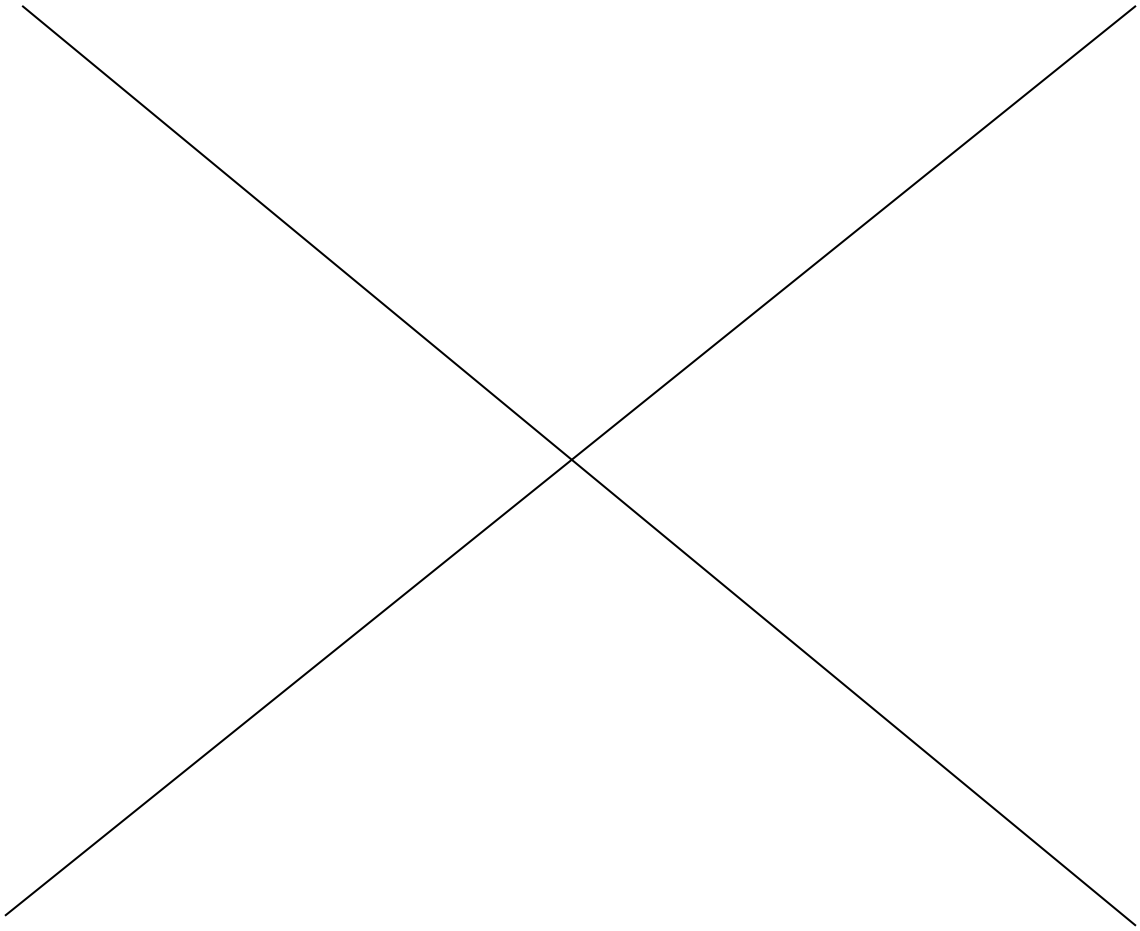
2018

Ausgangssituation

Als Mitarbeiter eines Projektteams werden Sie mit der Entwicklung eines Online-Systems für den Support von IT-Systemen beauftragt. Die Grundaufgabe des Systems besteht darin, Kunden bei auftretenden IT-Problemen zu unterstützen. Hierzu werden Störmeldungen (=Ticket) erfasst und von qualifizierten Supportmitarbeitern bearbeitet.

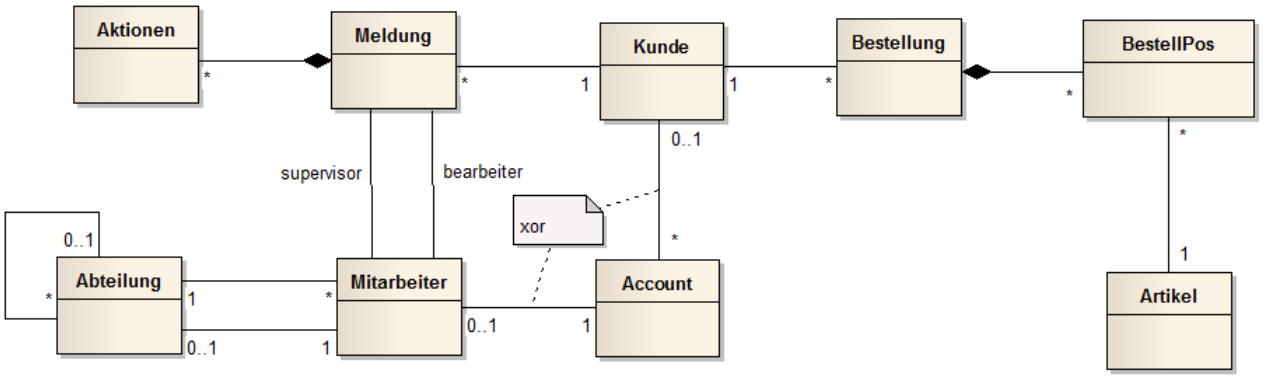
Produktvision:

- Ein Kunde soll für auftretende Probleme Störmeldungen erfassen können.
- Die Supportmitarbeiter bearbeiten die Meldungen gemäß einer festzulegenden Dringlichkeit.
- Die Kunden können zusätzlich über das System Artikel, wie z.B. Monitore und Drucker, bestellen.
- Damit ein Benutzer eine Funktion ausführen kann muss er sich am System anmelden.
- Das System soll sowohl über einen Web-Browser, als auch für gängige Smartphones angeboten werden.
- Bei der Umsetzung ist auf eine möglichst hohe Wiederverwendbarkeit zu achten.



1. Domain-driven Design

1a) Im Rahmen des Designs wenden Sie nun die Methode „Domain-driven Design“ auf das nachstehende Produktmodell an. Analysieren Sie das nachfolgende Diagramm und kennzeichnen Sie alle Klassen in diesem Diagramm. Stellen Sie folgende Kernelemente innerhalb des Diagramms heraus: Aggregate, Entity, Root-Entity, Value-Objects. Verwenden Sie falls notwendig Stereotypen.



1b) Wozu wird ein Bounded-Context sowie eine Context-Map benötigt? Begründen Sie warum diese beiden Konzepte sinnvoll sind und wie diese zusammenspielen.

1c) Erläutern Sie den Lebenszyklus eines Domain-Objekts und erläutern Sie die Auswirkung auf die Implementierung von Assoziationen. Wie findet sich dieses Konzept im Zusammenspiel mit Aggregaten wieder?

2. Software Architektur & Design Prinzipien

2a) Beschreiben Sie die Aufgabe des Konfigurationsmanagers innerhalb einer Komponentenarchitektur. Begründen Sie warum der Konfigurator essentiell für die Umsetzung der Losen Kopplung ist.

2b) Konfiguratoren können auf Basis des Prinzips „Dependency Injection“ implementiert werden. Erläutern Sie das Prinzip und begründen Sie warum dieses Prinzip für eine maximale Entkopplung der Software-Komponenten und dem Konfigurator sorgt.

2c) Bewerten Sie folgende Aussage: „Die Implementierung einer Komponenten-architektur ist äußerst aufwändig und kompliziert. Die direkte Kopplung ohne Konfigurationsmanager ist einfacher. Komponentenarchitektur hat keinen Nutzen“

2d) Was versteht man unter dem **Dependency-Inversion-Principle**? Beschreiben Sie das Prinzip und begründen Sie warum eine saubere Architektur durch dieses Prinzip unterstützt wird.

2e) Definieren Sie das **Liskovsches Substitutionsprinzip**. Erstellen Sie eine Skizze welches das Problem veranschaulicht.

3.) Software Architektur – Die Struktursicht

Im Rahmen einer Architekturbewertung sollen Sie nun Komponenten, sowie deren Schnittstellen bewerten. Hierzu wird ihnen für das Repository des Ticketsystems folgendes Quellcodefragment vorgelegt (Schnittstellenspezifikation):

```
public interface TechnicalLayer {
 // Studiengang laden
 java.jdom.JDomDocument loadMeldung( String meldungNr );
 // Studiengang in der Datenbank speichern
 void saveArtikel(jdbc.Connection conn, Artikel artikel );
 // Mail versenden
 void sendMail( java.mail.Adress recipient, String betreff, String body );
}
```

3a) Skizzieren Sie den Software-Kategoriegraphen unter der Annahme, dass die Anwendungskomponente „Ticket-Center“, sowie die Anwendungskomponente „Bestellung“ auf das TechnicalLayer zugreifen. Darüber hinaus gelten folgende Annahmen:

- Alle Datenbankspezifischen Klassen/Interfaces (jdbc) liegen in der Kategorie „JDBC“.
- Alle Email spezifischen Klassen/Interfaces (java.mail) liegen in der Kategorie „Mail“
- Alle XML/Dom spezifischen Klassen/Interfaces (java.jdom) liegen in der Kategorie „XML“
- Die Klassen des Produktmodells werden in der Kategorie „A“ (Anwendung) angesiedelt
- Collections liegen in Kategorie 0

3b) Begründen Sie, warum man gemäß Quasar A und T Software immer trennen sollte. Geben Sie ein negativ Beispiel und dessen Auswirkung an.

3c) Zeichnen Sie nun einen verbesserten Kategoriegraphen. Zerlegen Sie das Interface „TechnicalLayer“ in verschiedene Interfaces, um das Interface-Segregation-Prinzip zu gewährleisten.

3d) Definieren Sie das Konzept der R-Software. Welche besondere Einschränkung muss bei R-Software gelten. Welche Kategorien aus 3c entsprechen dieser R-Software

3e) Die Schnittstelle „TicketCenterIF“ wird in die Kategorie 0 gelegt. Welche Auswirkung hat diese Aussage auf die Parameter und Rückgabewerte der Interface-Methode(n). Wie können basierend auf dieser Entscheidung komplexe Objekte übergeben werden (2 Varianten)?

4.) Prozessicht & Skalierbarkeit

4a) Erläutern Sie die Grundidee des Session Context Manager-Patterns. Begründen Sie warum der Einsatz dieses Patterns die Skalierbarkeit der Software verbessert.

4b) Beschreiben Sie die Skalierungsstrategie **Scale-Out**. Sind Anpassung an der Software notwendig um diese Strategie nutzen zu können (Begründung)?

4c) Eine Maßnahme zur Verbesserung der Skalierung lautet „Asynchrone Kommunikation“. Was wird darunter verstanden und wie kann dadurch die horizontale Skalierbarkeit besser unterstützt werden? Welche Nachteile ergeben sich dadurch?

5) Die physische Sicht

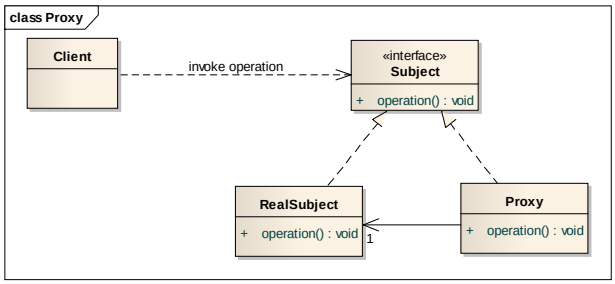
5a) Erläutern Sie die Grundidee eines Fail-Over-Clusters mit automatic fail-over. Skizzieren Sie eine Idee wie der Ausfall/Fehlfunktion eines Knoten erkannt werden kann.

5b) Analysieren sie das folgende Programmfragment in Hinblick auf den Einsatz in einem Application-Cluster. Identifizieren Sie die problematische(n) Stellen und geben Sie jeweils eine Begründung an. Skizzieren Sie für jedes der Probleme eine mögliche Lösung.

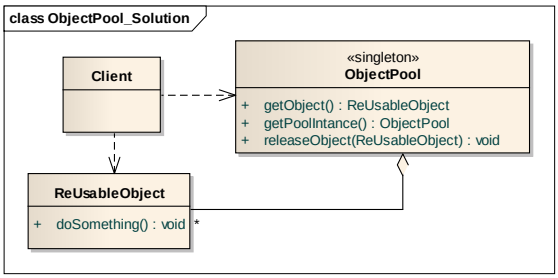
```
01 public class NumberServiceImpl implements NumberService {
02
03    Map<String,AtomicLong> numberMap = new HashMap<String,AtomicLong>();
04
05    public long getNextNumber( String category ) {
06        synchronized( this ) {// Enter this block exclusively
07            if( numberMap.containsKey( category ) == false ) {
08                numberMap.put( category, new AtomicLong( 1 ) );
09            }
10            return numberMap.get( category ).incrementAndGet();
11       }
12    }
13}
```

6 Design Pattern

6a) Erläutern Sie die Grundidee/Aufbau des Proxy-Patterns. Geben Sie ein sinnvolles Beispiel für die Verwendung dieses Patterns an. Wie unterscheidet sich dieses Pattern vom Decorator-Pattern?

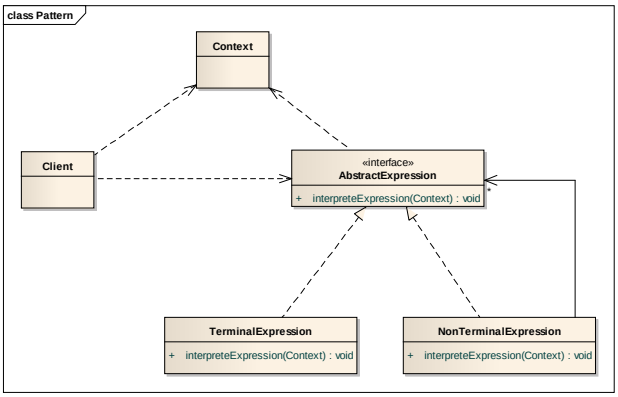


6b) Erläutern Sie die Grundidee/Aufbau des Objekt-Pool-Patterns. Wann wird dieses Pattern eingesetzt? Worauf muss besonders geachtet werden? Wo liegen mögliche Probleme? Geben Sie ein sinnvolles Anwendungsbeispiel an.



6 Design Pattern, Verhaltensmuster

6c) Erläutern Sie die Grundidee des Interpreter-Patterns. Wie unterscheidet sich dieses Pattern von dem Visitor-Pattern. Wo liegen die Grenzen des Interpreter-Patterns. Geben Sie ein sinnvolles Anwendungs-beispiel an.



6d) Wenden Sie nun das Interpreter-Muster auf die Berechnung von arithmetische Ausdrücke an. Skizzieren Sie ein Klassendiagramm mit welchem Ausdrücke mit folgenden Eingeschalten dargestellt werden können: Operatoren: *, /, Konstanten und Variablen. Erläutern sie wozu in diesem Fall der „Context“ benötigt wird.