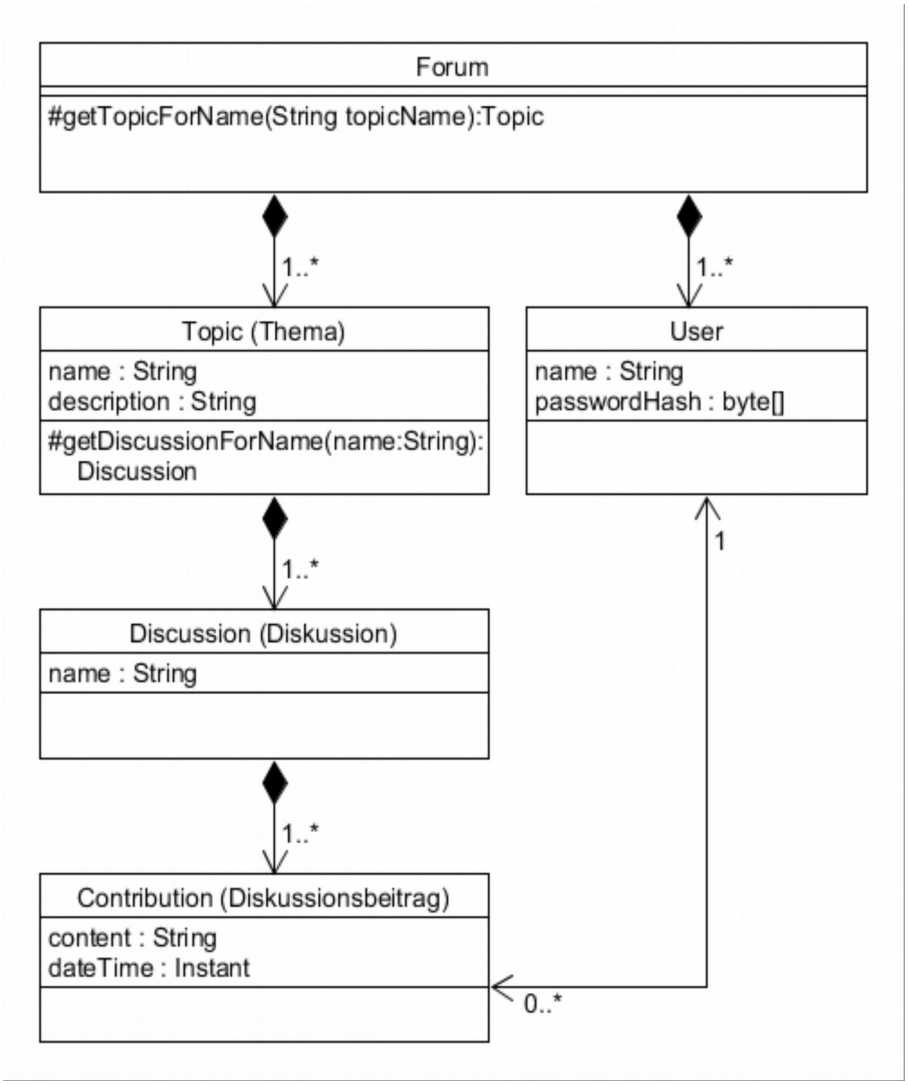
Praktikum GRASP

1 Problembeschreibung

Alle Teilaufgaben basieren auf dem folgenden Design-Klassendiagramm. Es stellt einen unvollständigen Design einer Anwendung zur Verwaltung eines Diskussionsforums dar. Ein Forum besteht aus einem oder mehreren Themen, und jedes Thema besteht aus einem oder mehreren Diskussionen. Eine Diskussion besteht aus einem oder mehreren Beiträgen.
Um irgendetwas verändern oder hinzufügen zu können, muss sich ein Benutzer mit seinem Namen und Passwort identifizieren.
Das Klassendiagramm verwendet spezielle Assoziationen, die wir noch nicht besprochen haben. Was könnte die Bedeutung der ausgefüllten Rauten sein?



2 Aufgaben

1. Welche Klasse ist der Fassaden-Controller?
2. Die Systemoperation „getNbrOfContributions()“ gibt die Anzahl Diskussionsbeiträge („Klasse Contribution“) im gesamten Forum zurück. Zeichnen Sie ein Kommunikationsdiagramm, das seinen Ablauf darstellt und ergänzen Sie das Klassendiagramm. Erklären Sie Ihre Design-Entscheide kurz, indem Sie auf Entwurfsmuster verweisen.
3. Die Systemoperation „addNewDiscussion(userName : String, passwordHash:byte[]; topicName : String; discussionName : String) fügt eine neue Diskussion einem bestehendem Topic hinzu. Es muss überprüft werden, ob so eine Diskussion bereits existiert und in diesem Fall muss eine Exception geworfen werden. Zeichnen Sie ein Sequenzdiagramm, das den Ablauf darstellt und ergänzen Sie das Klassendiagramm. Erklären Sie Ihre Design-Entscheide kurz, indem Sie auf Entwurfsmuster verweisen.
4. Vervollständigen Sie den Code und die Testfälle für die obigen Systemoperationen im Quelltext, den Sie auf OLAT (Forum-Projekt.zip) finden. Fügen Sie den zusätzlichen Quelltext der Domänenlogik und der Testfälle direkt in Ihr PDF-Dokument ein, das Sie abgeben. Sie müssen den Quelltext nicht in Textform abgeben aber allenfalls dem Dozierenden demonstrieren.

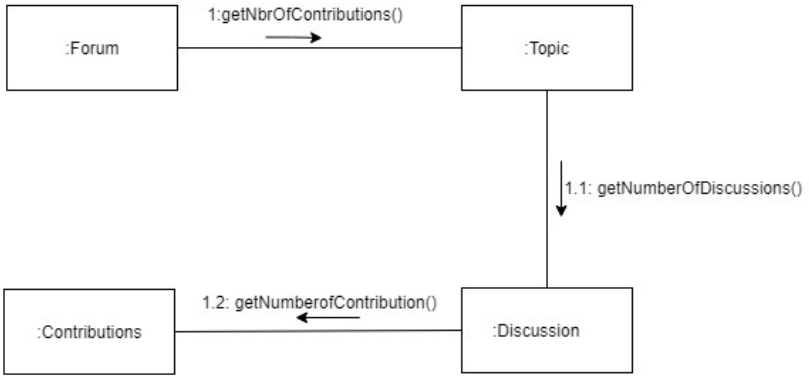
Bei den Aufgaben 2 und 3 brauchen Sie die Zugriffe auf Container Objekte (die Objekte, die die 1..* Beziehung realisieren) nicht im Detail zu modellieren. Wenn Sie dies so machen, sollten Sie aber Methoden einfügen, die diese Zugriffe abkapseln oder Sie vermerken über Notizen, wo Sie auf die Container Objekte zugreifen.

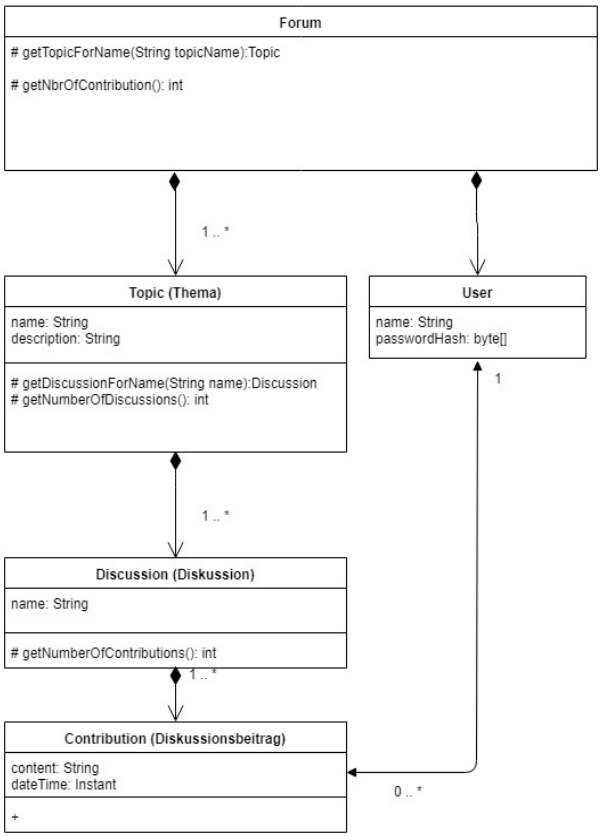
Antworten Pascal Brunner

1. Welche Klasse ist der Fassaden-Controller?

   Klasse: Forum - Denn der User nimmt die Interaktionen mit dieser Klasse vor

2. Die Systemoperation „getNbrOfContributions()“ gibt die Anzahl Diskussionsbeiträge („Klasse Contribution“) im gesamten Forum zurück. Zeichnen Sie ein Kommunikationsdiagramm, das seinen Ablauf darstellt und ergänzen Sie das Klassendiagramm. Erklären Sie Ihre Design-Entscheide kurz, indem Sie auf Entwurfsmuster verweisen.





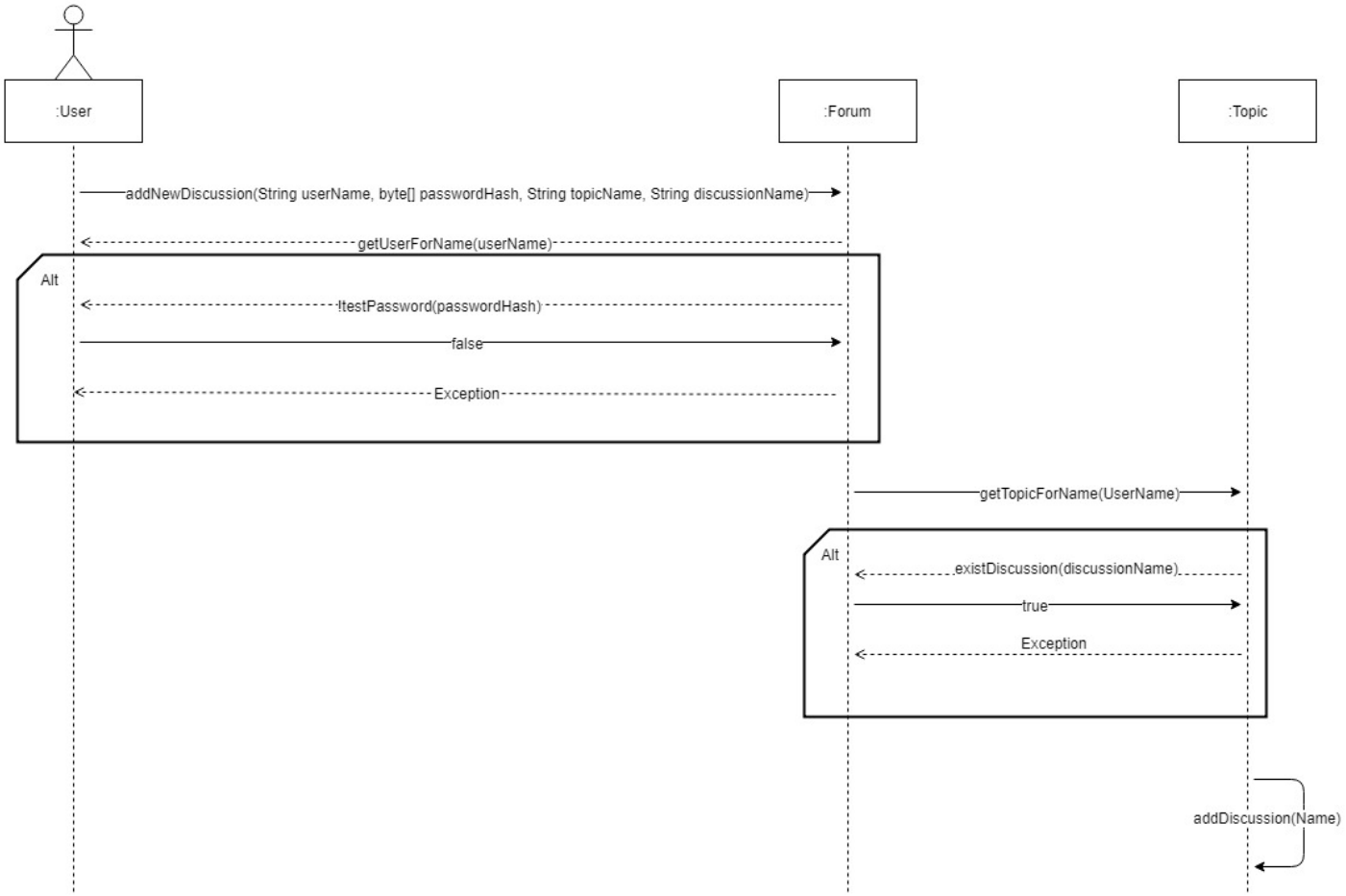
Aus dem Klassendiagramm ist ersichtlich, dass ein
- Topic ein oder mehrere Discussion hat
- Discussion ein oder mehrere Contributions hat

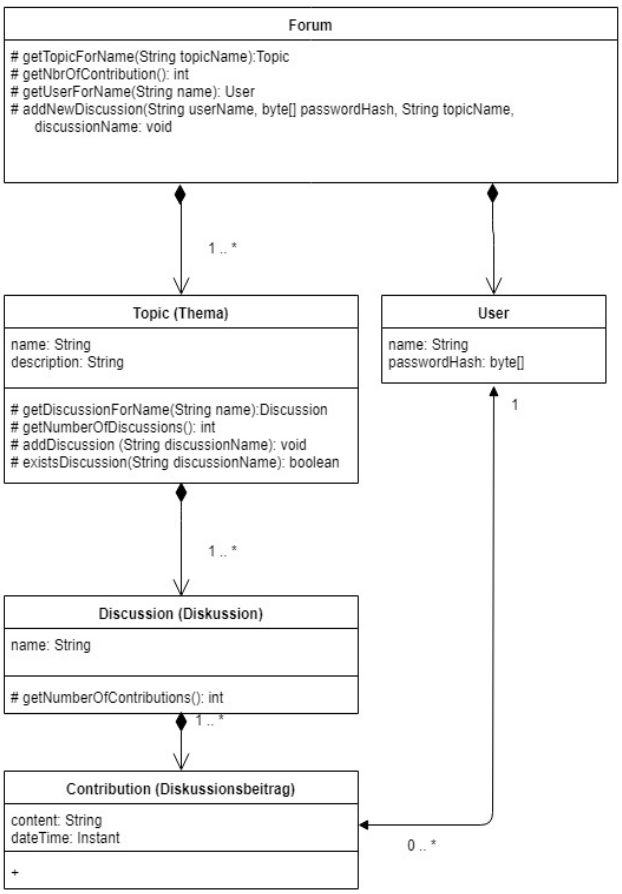
Damit man nun auf die gewünschte Summe aller Contributions kommt, muss man durch jedes Topic und innerhalb des Topics durch jede Discussion iterieren und auf dieser Ebene sämtliche Contributions zusammenzählen.

Ich empfand es als sinnvoll, dass jede Klasse "seine Aufgabe" übernimmt und die Iteration zu seiner darunterliegenden Klasse durchführt.

Aus diesem Grund iteriert getNbrContributions() durch alle Topics, jedes Topic iteriert (topic.getNumberOfDiscussions) durch sämtliche Discussion und die Discussion zählt sämtliche Contributions mit discussion.getNumberOfContributions()

3. Die Systemoperation „addNewDiscussion(userName : String, passwordHash:byte[]; topicName : String; discussionName : String) fügt eine neue Diskussion einem bestehendem Topic hinzu. Es muss überprüft werden, ob so eine Diskussion bereits existiert und in diesem Fall muss eine Exception geworfen werden. Zeichnen Sie ein Sequenzdiagramm, das den Ablauf darstellt und ergänzen Sie das Klassendiagramm. Erklären Sie Ihre Design-Entscheide kurz, indem Sie auf Entwurfsmuster verweisen.





Die Methode wurde in Forumsklasse hinzugefügt, da hier die Interaktion mit dem User stattfindet.

des Weiteren wurde im Topic die Methode addDiscussion hinzugefügt, sowie die die Methode existDiscussion, welche überprüft ob eine Diskussion bereits besteht. Der Ablauf ist im oberen Sequenzdiagram ersichtlich.

4. Vervollständigen Sie den Code und die Testfälle für die obigen Systemoperationen im Quelltext, den Sie auf OLAT (Forum-Projekt.zip) finden. Fügen Sie den zusätzlichen Quelltext der Domänenlogik und der Testfälle direkt in Ihr PDF-Dokument ein, das Sie abgeben. Sie müssen den Quelltext nicht in Textform abgeben aber allenfalls dem Dozierenden demonstrieren.

```java
/**
 * Represents the whole forum with its list of topics and users
 *
 */
public class Forum {
    private Clock clock;
    private List<User> users = new ArrayList<>();
    private List<Topic> topics = new ArrayList<>();

    public Forum(Clock clock) {
        super();
        this.clock = clock;
    }
```

```java
// Meine Änderungen
protected int getNbrOfContributions(){
    int counter = 0;
    List<Topic> topics = getTopics();
    // Iterate through every topic and sum the total number of Contributions in all discussions
    for(Topic topic : getTopics()) {
        counter += topic.getNumberOfDiscussions();
    }
    return counter;
}
```

```java
//meine Änderungen
public int getNumberOfDiscussions(){
    int counter = 0;
    List<Discussion> discussions = getDiscussions();
    for(Discussion discussion : discussions){
        counter += discussion.getNumberofContribution();
    }
    return counter;
}
```

```java
//meine Änderungen
public int getNumberofContribution(){
    return getContributions().size();
}
```

```java
//meine Änderungen - addNewDiscussion
protected void addNewDiscussion(String userName, byte[] passwordHash, String topicName, String discussionName) throws Exception {
    if(topicName == null || topicName == "" || discussionName == null || discussionName == ""){
        throw new Exception(); // Parameter, use proper Exception
    }
    User user = getUserForName(userName);
    if(user == null){
        throw new Exception(); // NotFound, use proper Exception
    }
    if(!user.testPassword(passwordHash)){
        throw new Exception(); // NotAuthenticated, use proper Exception
    }
    Topic topic = getTopicForName(topicName);
    if(topic == null){
        throw new Exception(); // NotFound, use proper Exception
    }
    if(topic.existsDiscussion(discussionName)){
        throw new Exception(); // Discussion Exists, use proper Exception
    }
    topic.addDiscussion(discussionName);
```

```java
//Meine Änderungen
public void addDiscussion(String discussionName){
    Discussion discussion = new Discussion(discussionName);
    discussions.add(discussion);
}
// meine Änderungen
public Boolean existsDiscussion(String discussionName){
    for (Discussion discussion : discussions) {
        if(discussion.getName() == discussionName){
            return true;
        }
    }
    return false;
}
```

```java
@Test
public void addNewDiscussionTest(){
    //Test if execptions get return
    try{
        forum.addNewDiscussion( userName: "", passwordHash: null, topicName: "", discussionName: "testName");

        fail("Expected exception");
    } catch(Exception ex){}

    try{
        forum.addNewDiscussion( userName: null, passwordHash: null, topicName: "", discussionName: "testName");

        fail("Expected exception");
    } catch(Exception ex){}

    try{
        forum.addNewDiscussion( userName: null, passwordHash: null, topicName: "", discussionName: "");

        fail("Expected exception");
    } catch(Exception ex){}

    try{
        forum.addNewDiscussion( userName: null, passwordHash: null, topicName: "testTopicName", discussionName: "testName");

        fail("Expected exception");
    } catch(Exception ex){}

    try{
        forum.addNewDiscussion(user.getName(), passwordHash: null, topicName: "testTopicName", discussionName: "testName");

        fail("Expected exception");
    } catch(Exception ex){}

    try{
        forum.addNewDiscussion(user.getName(),new byte[]{0}, topicName: "", discussionName: "");

        fail("Expected exception");
    } catch(Exception ex){}

    try{
        forum.addNewDiscussion(user.getName(), passwordHash: null, topicName: "", discussionName: "testName");

        fail("Expected exception");
    } catch(Exception ex){}

    try{
        forum.addNewDiscussion(user.getName(), passwordHash: null, topicName: "testTopicName", discussionName: "");

        fail("Expected exception");
    } catch(Exception ex){}

    //Test if new Discussion works
    try{
        Topic topic = new Topic( name: "testTopicName",  description: "Testeintrag");
        forum.getTopics().add(topic);
        forum.addNewDiscussion(user.getName(),new byte[]{0}, topicName: "testTopicName",  discussionName: "testDiscussionName");
        assertEquals(topic.getDiscussions().size(),  actual: 1);
        assertEquals(topic.getDiscussions().get(0).getName(),  actual: "testTopicName");
    } catch(Exception ex){
        fail("Expected no exception");
    }
```

```java
@Test
public void getNbrOfContributionsTest(){
    int counter = forum.getNbrOfContributions();
    assertEquals(counter,  actual: 0);

    Topic topic = new Topic( name: "testTopicName",  description: "Testeintrag");
    try{
        forum.getTopics().add(topic);
        forum.addNewDiscussion(user.getName(),new byte[]{0}, topicName: "testTopicName",  discussionName: "testDiscussionName");
    } catch(Exception ex){
        fail("Expected no exception");
    }
    counter = forum.getNbrOfContributions();
    assertEquals(counter,  actual: 0);

    topic.getDiscussions().get(0).getContributions().add(new Contribution(("TestContribution"), user, Instant.now()));
    counter = forum.getNbrOfContributions();
    assertEquals(counter,  actual: 1);
}
```