Software Engineering and Design - Team Red

Domain Model and Sequence Diagram

Task 4: Domain Model & Sequence Diagrams

Thomas Baumann Frédéric Lehmann Severin Thalmann
Ismael Riedo Fridolin Zurlinden Roland Roccaro
Tobias Weissert

April 27, 2018

Contents

1 Klassendiagramm

Klassen Diagramm in kompletter Form:

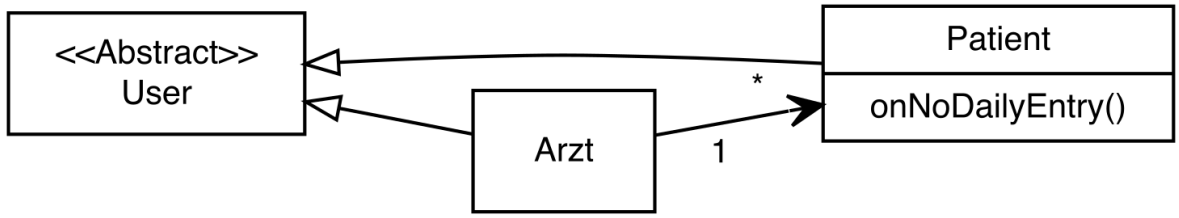

Figure 1.1: Klassen Diagramm Full

1.1 Users

In diesem Diagramm sehen wir das detaillierte Klassen Diagramm für die User unseres Tools:

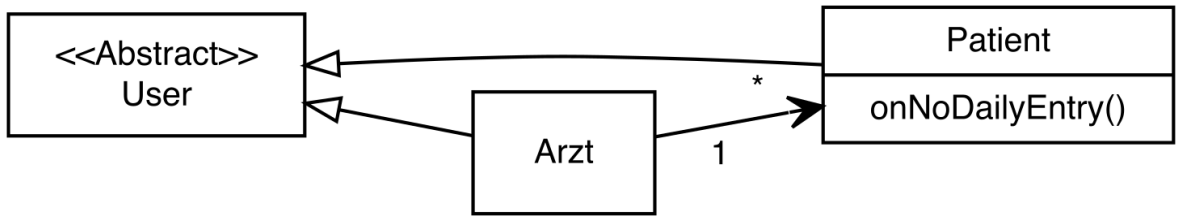

Figure 1.2: Klassen Diagramm Users

Wir haben für die Benuzter eine abstrakte Klasse ”User”, welche von den Klassen ”Patient” und ”Arzt” geerbt wird. Die Klasse Arzt hat zusätzlich eine Referenz auf die Patienten Klasse. Das Ziel dabei ist, dass Ärzte die Möglichkeit haben auf die Fortschritte ihrer Patienten zuzugreifen.

1.2 Journal

In diesem Diagramm sehen wir die Details der Journal Klasse:

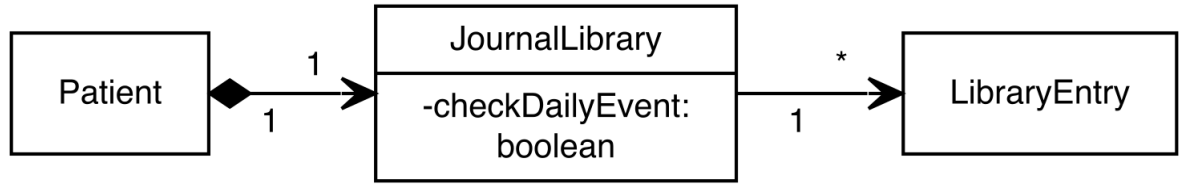

Figure 1.3: Klassen Diagramm Journal

Das Ziel des Journal ist es, dass Patienten eine Art Tagebuch führen können. Dafür haben wir eine Journal Library Klasse welche die Journal Einträge speichert und zusätzliche Funktionen bereitstellt.

1.3 Challenges

Hier sehen wir die Details der Klasse für die Challenge.

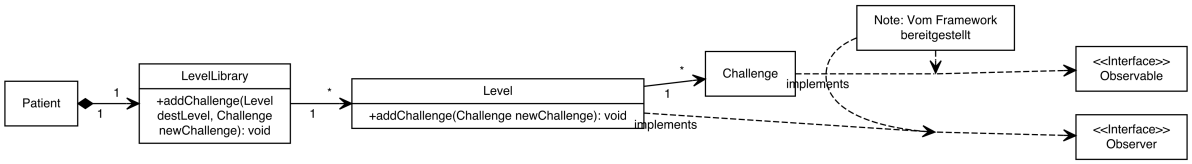

Figure 1.4: Klassen Diagramm Challenge

Auch für die Levels haben wir eine Level Library welche die einzelnen Levels speichert. Die Level Klasse soll die einzelnen Challenges speichern. Zusätzlich haben wir uns entschieden für die Level und Challenge Klassen ein Observable Pattern zu verwenden. Ziel ist die Umkehrung der Abhängigkeit zwischen Level und Challenge. Die Interfaces ”Observer” und ”Observable” werden durch das Java Framework bereitgestellt, diese müssen nur noch implementiert werden.

1.4 Data Layer Klassen

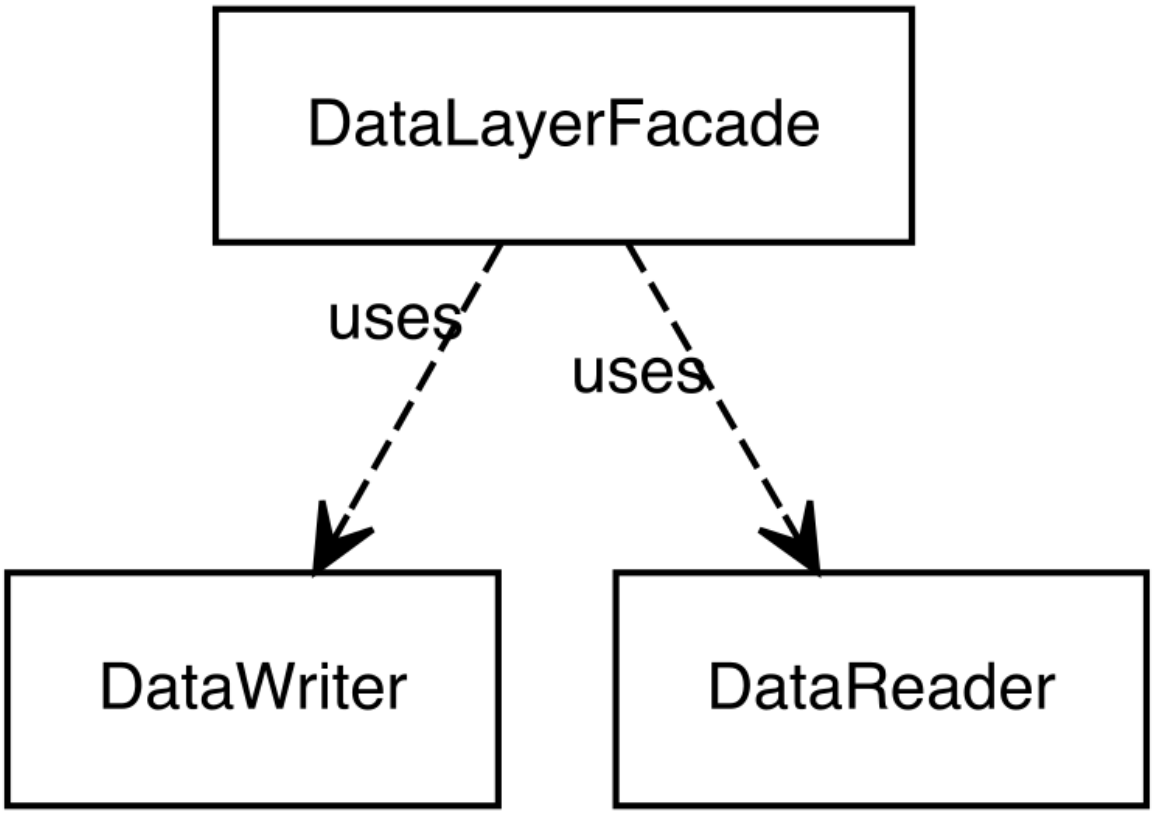

Figure 1.5: Klassen Diagramm Data Layer Klassen

2 Domain Diagramm

Domain Diagramm in kompletter Form:

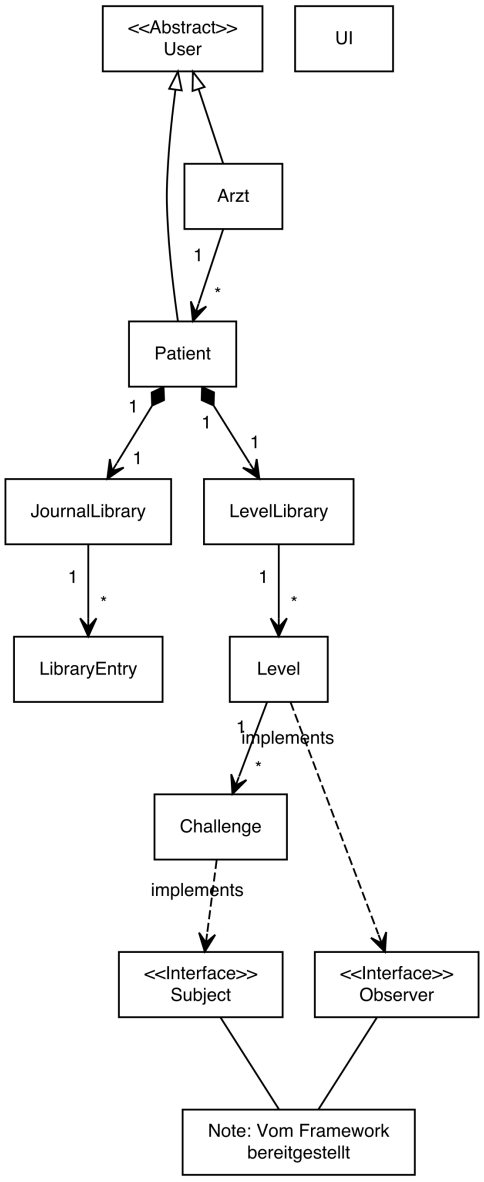

Figure 2.1: Domain Diagramm Full

2.1 Users

In diesem Diagramm sehen wir das detaillierte Klassen Diagramm für die User unseres Tools:

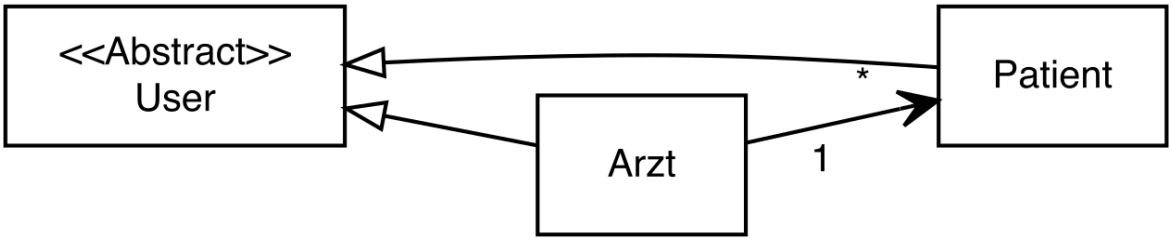

Figure 2.2: Domain Diagramm Users

Wir haben für die Benuzter eine abstrakte Klasse "User", welche von den Klassen "Patient" und "Arzt" geerbt wird. Die Klasse Arzt hat zusätzlich eine Referenz auf die Patienten Klasse. Das Ziel dabei ist, dass Ärzte die Möglichkeit haben auf die Fortschritte ihrer Patienten zuzugreifen.

2.2 Journal

In diesem Diagramm sehen wir die Details der Journal Klasse:

Figure 2.3: Domain Diagramm Journal

Das Ziel des Journal ist es, dass Patienten eine Art Tagebuch führen können. Dafür haben wir eine Journal Library Klasse welche die Journal Einträge speichert und zusätzliche Funktionen bereitstellt.

2.3 Challenges

Hier sehen wir die Details der Klasse für die Challenge.

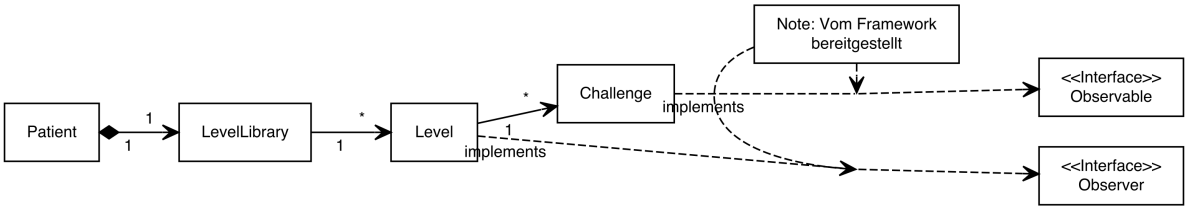

Figure 2.4: Domain Diagramm Challenge

Auch für die Levels haben wir eine Level Library welche die einzelnen Levels speichert. Die Level Klasse soll die einzelnen Challenges speichern. Zusätzlich haben wir uns entschieden für die Level und Challenge Klassen ein Observable Pattern zu verwenden. Ziel ist die Umkehrung der Abhängigkeit zwischen Level und Challenge. Die Interfaces ”Observer” und ”Observable” werden durch das Java Framework bereitgestellt, diese müssen nur noch implementiert werden.

2.4 Data Layer Klassen

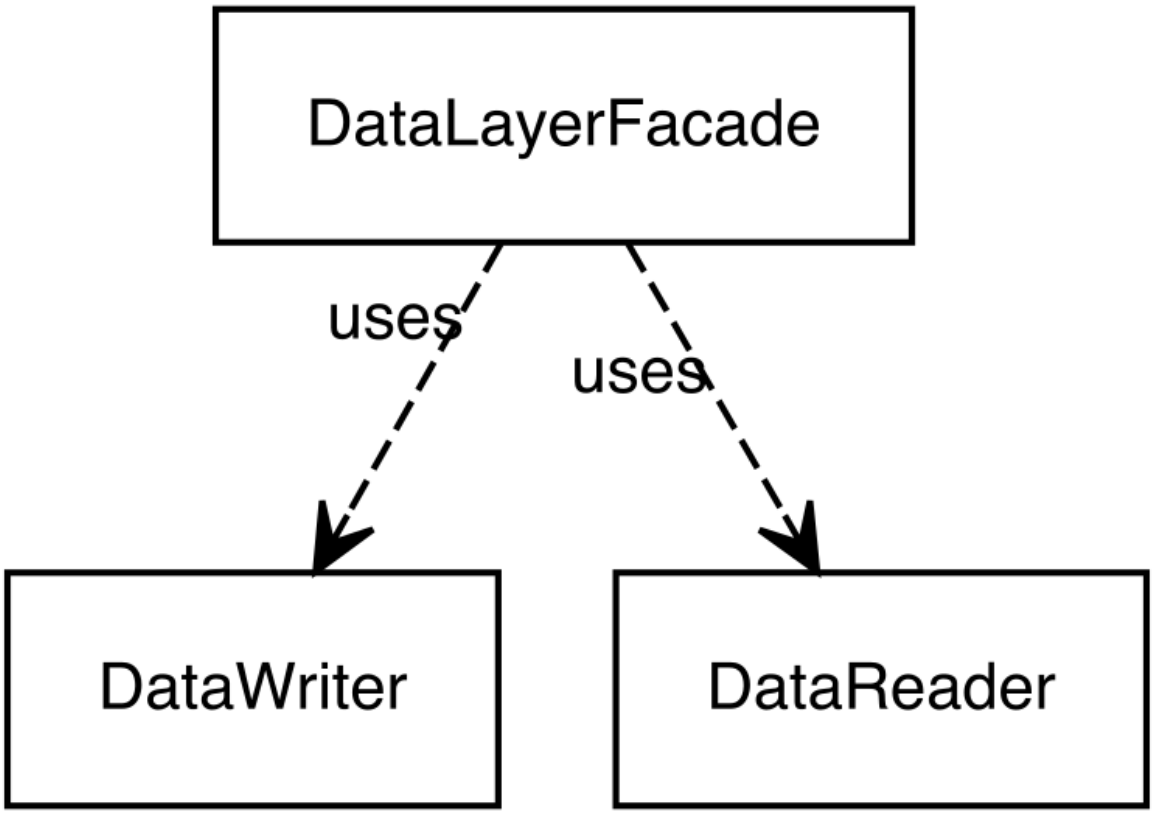

Figure 2.5: Domain Diagramm Data Layer Klassen

3 Package Diagramm

Wir haben uns ein Package Diagramm überlegt, welches uns ermöglichen soll möglichst flexibel Packages zu ersetzen/hinzufügen, ohne dass die Hauptlogik (enthalten im ”Business Layer” und im ”Common Layer”) verändert werden muss.

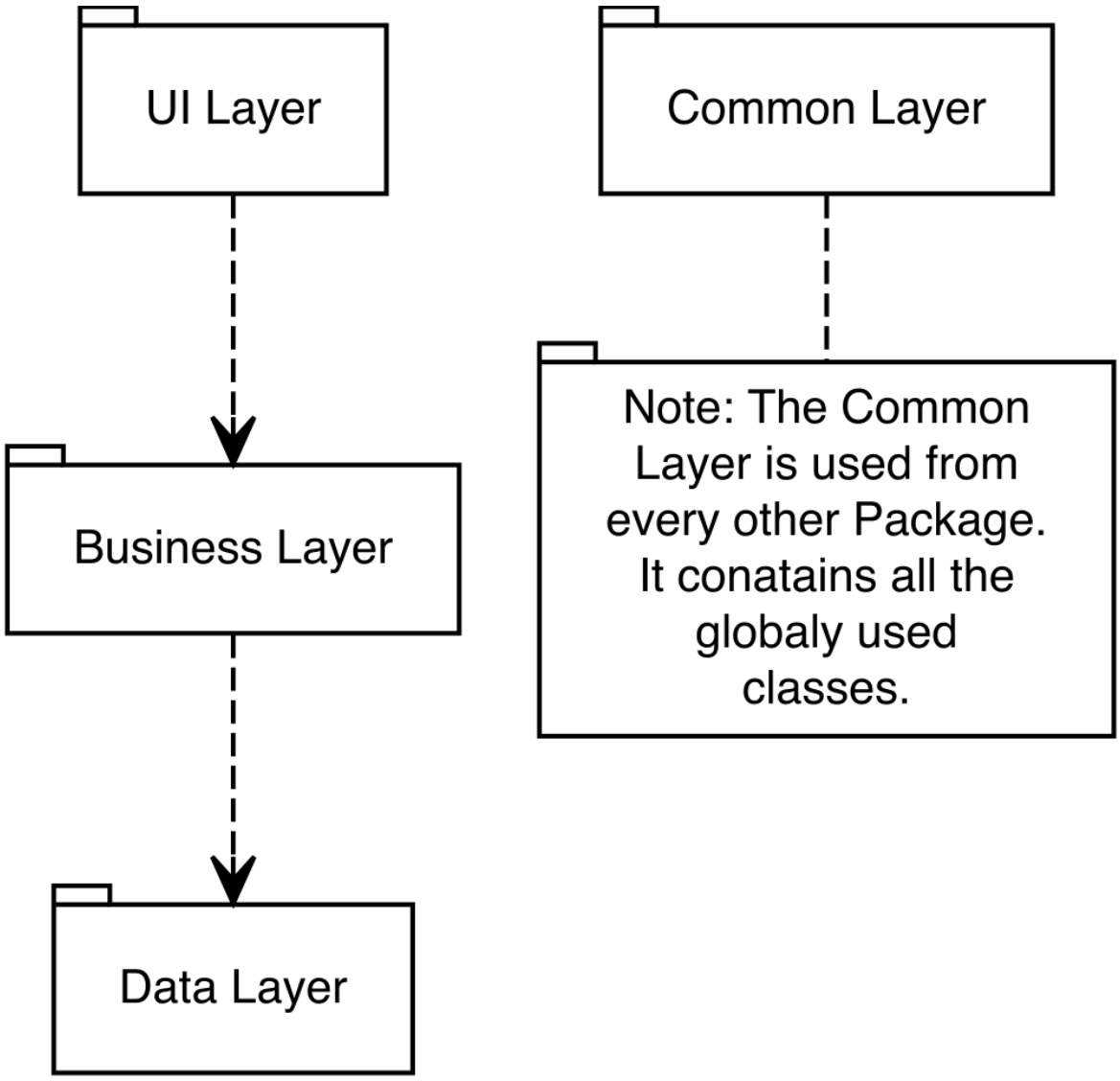

Figure 3.1: Package Diagramm

3.1 UI Layer

Im UI Layer wird die ganze UI Logik gehandelt, es wird das ”Business Layer” und das ”Common Layer” Package verwendet. Ziel dabei ist es, falls ein neues oder zusätzliches UI erstellt werden soll, dass das Package möglichst einfach an die Kern Logik angebunden werden kann.

3.2 Business Layer

Im Business Layer wird die gesammte Logik der Applikation abgebildet.

3.3 Data Layer

Der Data Layer ist für die Speicherung und für das Abrufen von Daten zuständig.

3.4 Common Layer

Im Common Layer sind die Klassen bzw. Models enthalten die in allen anderen Packages verwendet werden. Z.B: die User Klasse oder die Challenge Klasse.

4 Sequenz Diagramm

4.1 Erstellen einer neuen Challenge

In diesem Diagramm sehen wir das detaillierte Klassen Diagramm für die User unseres Tools:

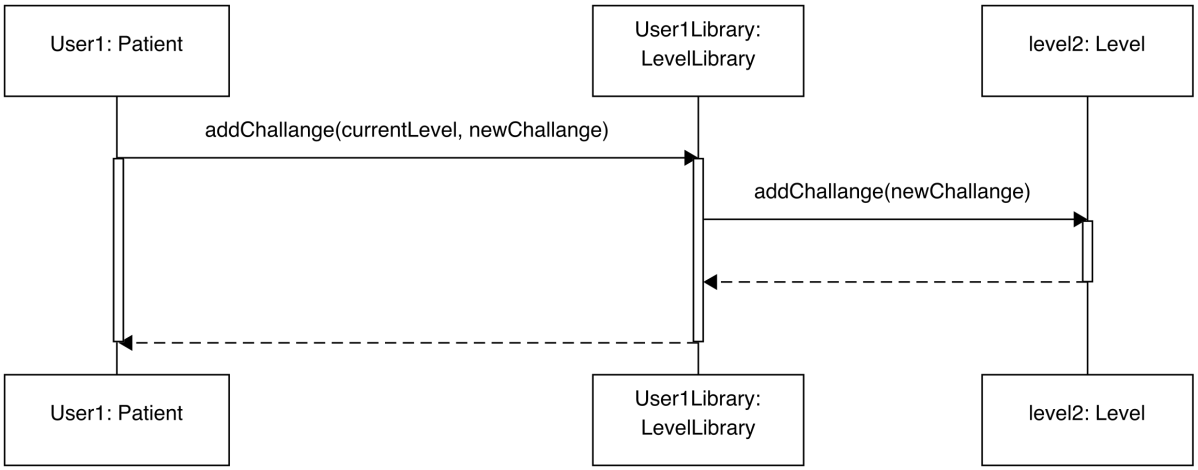

Figure 4.1: Sequenz Diagramm Challenge

Dieses Sequenz Diagramm beschreibt den Ablauf und die genutzten Klassen wenn der User eine neue Challenge erstellt.

4.2 Noch kein Journal Eintrag für den heutigen Tag

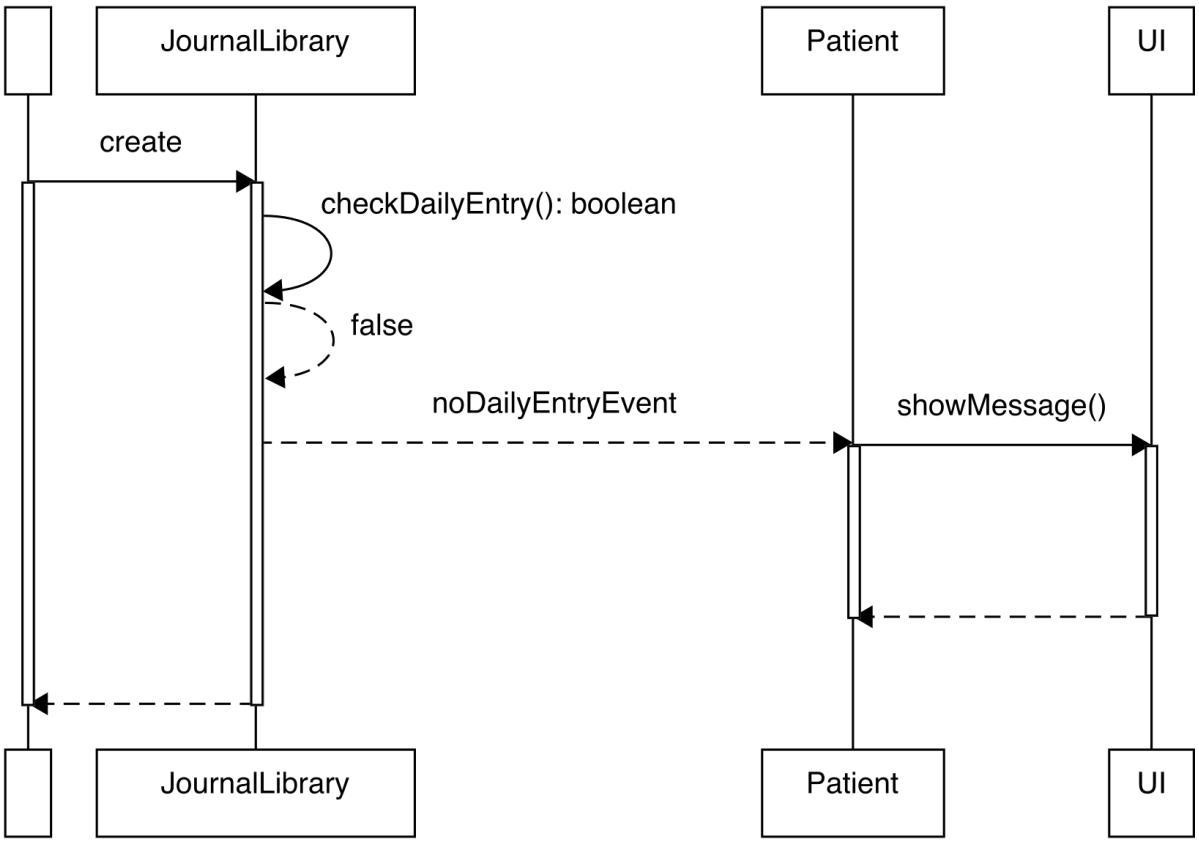

Figure 4.2: Sequenz Diagramm Journaleintrag

Dieses Sequenz Diagramm zeigt wie nach dem Login, beim erstellen der Journal Library, überprüft wird ob der User für den heutigen Tag bereits einen Journal Eintrag erstellt hat. Falls nicht wird auf dem UI eine Meldung angezeigt.

5 Index

List of Figures